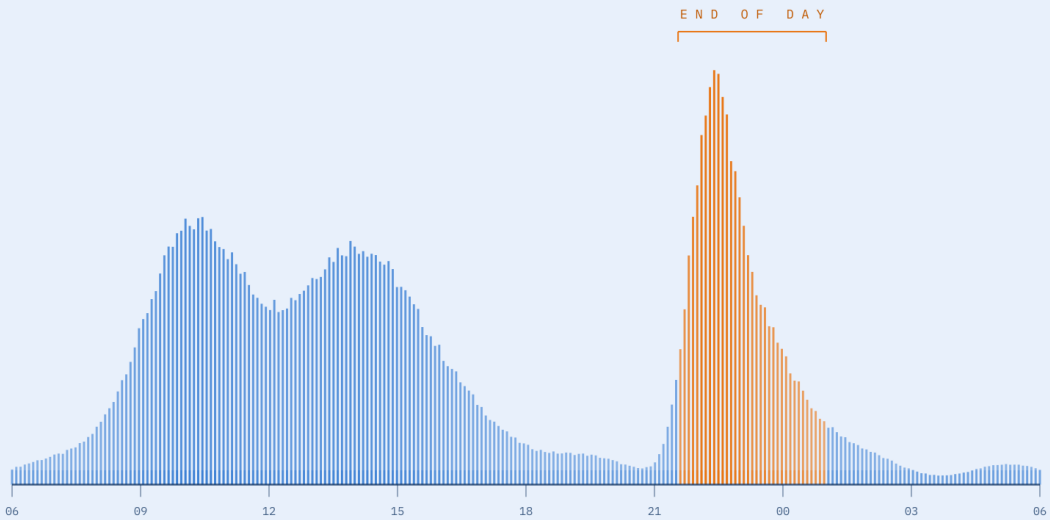
EDISI PERTAMA

IBM i 7.6 · POWER10

OBS. 001 — BEBAN TRANSAKSIONAL 24 JAM

SKALA RELATIF · SUMBU WAKTU 06:00–06:00



PANDUAN INSINYUR SISTEM

HANDBOOK OF SYSTEM ENGINEERING

IBM i · AS/400

Arsitektur · Operasi · Core Banking

Romi Nur Ismanto

rominur.com · hello@rominur.com

216 BAB · 45 RUNBOOK · 114 GAMBAR · 1827 ENTRI INDEKS

PEOPLE
PROCESS
DATA
A MORE
RESILIENT
WORLD
IBM i / AS400
TRUSTED
TODAY
READY
TOMORROW
PLATFORM SENYAP YANG MASIH MENJALANKAN BISNIS BESAR DUNIA
Stabil sejak generasi AS/400, berevolusi menjadi IBM i yang aman, terintegrasi, dan sangat andal untuk core business mission-critical.
Terintegrasi
Database, security, middleware, dan operasi dalam satu platform.
Tangguh
Dirancang untuk availability tinggi dan operasional jangka panjang.
Aman
Audit, otorisasi, dan kontrol sistem kelas enterprise.
Modern
Mendukung API, SQL, otomasi, virtualisasi, dan transformasi digital.
ENTERPRISE
WORKLOADS.
REAL BUSINESS
IMPACT.
LEGACY
STRENGTH
MODERN
OPPORTUNITY
IBM
Power Systems
STABLE
SECURE
INTEGRATED
SCALABLE
ALWAYS ON
DATA
APPLICATIONS
BUSINESS
SAME
PLATFORM.
A BRIGHTER
TOMORROW.
DATA
PEOPLE
APPLICATIONS
INNOVATION
CONTINUITY
APIs
Security
Automation
Virtualization
Integration
IBM i
Built for what's next
IBM
Power Systems
BUILT FOR
A SMARTER
PLANET
RELIABILITY TODAY • OPPORTUNITY TOMORROW
Banking
KEPERCAYAAN
SETIAP TRANSAKSI
Retail
MENGGERAKKAN
JUTAAN PELANGGAN
Manufacturing
EFISIENSI
TANPA HENTI
Logistics
RANTAI PASOKAN
LEBIH KUAT
Government
LAYANAN PUBLIK
LEBIH BAIK
AS/400
1988–2000
THE FOUNDATION
iSeries
2000–2008
EXPANDED POSSIBILITIES
System i
2008–2019
STRONGER TOGETHER
IBM i
2019–TODAY
BUILT FOR TOMORROW
Bukan sekadar legacy.
Ini adalah mesin bisnis yang terus berevolusi.
IBM
TECHNOLOGY
FOR A BRIGHTER
TOMORROW
SAME
PLATFORM
MORE
POSSIBILITIES

Daftar Isi

Tentang Buku Ini 12

Bagian I — Fondasi 13

Bab 1. Posisi IBM i dalam Arsitektur TI Modern 13

Bab 2. Evolusi Platform 15

Bab 3. Kerangka Rekayasa Sistem Buku Ini 16

Bab 4. Peran dan Kompetensi System Engineer IBM i 16

Bagian II — Arsitektur Platform 18

Bab 5. TIMI dan Arsitektur Berbasis Objek 18

Bab 6. Single-Level Storage 20

Bab 7. Objek, Library, dan Integrated File System 21

Bab 8. LPAR, PowerVM, dan VIOS 23

Bab 9. Db2 for i 24

Bab 10. Work Management 26

Bab 10A. Pemrograman CL untuk System Engineer 29

Bagian III — Perencanaan dan Implementasi 36

Bab 11. Requirement Engineering dan Sizing 36

Bab 12. Desain Topologi 38

Bab 13. Instalasi, Upgrade, dan Manajemen PTF 39

Bab 14. Konfigurasi Awal 41

Bab 15. Manajemen Konfigurasi dan Baseline 42

Bagian IV — Operasi Harian 44

Bab 16. Administrasi Pengguna dan Wewenang 44

Bab 17. Monitoring 47

Bab 18. Manajemen Penyimpanan dan Journaling 48

Bab 19. Backup dan Recovery 50

Bab 20. Penjadwalan dan Output Management 51

Bab 21. Runbook dan Integrasi ITIL 52

Bagian V — Kinerja, Keandalan, dan Keamanan 53

Bab 22. Performance Engineering 53

Bab 23. Capacity Planning 56

Bab 24. High Availability dan Disaster Recovery 58

Bab 25. Keamanan IBM i 60

Bab 26. Audit Journal 63

Bab 27. Kepatuhan 64

Bagian VI — Core Banking di atas IBM i 64

Bab 28. Lanskap Core Banking di IBM i 65

Bab 29. Arsitektur Lingkungan dan Partisi 66

Bab 30. Administrasi Core Banking bagi System Engineer 67

Bab 31. Batch Processing dan End of Day 70
Bab 32. Backup dan Recovery Lingkungan Core Banking 76
Bab 33. Debugging Program RPG 79
Bab 34. Kinerja Batch dan EOD 83
Bab 35. Tata Kelola dan Audit Core Banking 85

Bagian VII — Integrasi dan Modernisasi 86
Bab 36. Konektivitas dan API 86
Bab 37. Modernisasi Aplikasi 88
Bab 38. DevOps di IBM i 89
Bab 39. IBM i dan Hybrid Cloud 91
Bab 40. Manajemen Siklus Hidup Legacy 92

Bagian VIII — Tata Kelola dan Penutup 93
Bab 41. Manajemen Risiko dan Knowledge Transfer 93
Bab 42. Studi Kasus 93
Bab 43. Tren dan Arah Platform 94

Bagian IX — Katalog Runbook 95
RB-001: Disk ASP melewati ambang kritis 95
RB-002: Job berstatus MSGW di tengah EOD 96
RB-003: Record lock menahan job EOD 96
RB-004: EOD gagal di fase akrual 97
RB-005: GL tidak seimbang setelah EOD 98
RB-006: Replikasi ke DR tertinggal 98
RB-007: Kinerja menurun mendadak di jam layanan 99
RB-008: Backup semalam gagal 99
RB-009: Level check CPF4131 massal 100
RB-010: Profil pengguna terkunci massal 100
RB-011: IPL tidak selesai atau sistem tidak naik 101
RB-012: Kehabisan job number atau job tabel penuh 102
RB-013: Akses ODBC mencurigakan terdeteksi 102
RB-014: Tabel riwayat tumbuh tak terkendali 102
RB-015: Pemulihan setelah kehilangan partisi produksi 103

Bagian X — Katalog IBM i Services 103
10.1 Kenapa beralih ke SQL 104
10.2 Status sistem dan kapasitas 104
10.3 Job dan work management 104
10.4 Penyimpanan dan objek 105
10.5 Basis data 106
10.6 Keamanan 107
10.7 Audit journal 107
10.8 Jaringan dan komunikasi 108
10.9 PTF dan pemeliharaan 109

10.10 Journaling 109
10.11 Membangun pemantauan sendiri 109

Bagian XI — Primer RPG ILE untuk System Engineer 110
Bab 44. Anatomi Program RPG Modern 110
Bab 45. Membaca Kode RPG Warisan 119

Bagian XII — Katalog Message ID 121
Bab 46. Membaca Message ID 121
Bab 47. Katalog Pesan Operasional 121
Bab 48. Membangun Kamus Pesan Internal 124

Bagian XIII — Db2 for i Mendalam 126
Bab 49. Optimizer dan Jalur Akses 126
Bab 50. Commitment Control dan Integritas Transaksi 130
Bab 51. Fitur Db2 yang Sering Terlewat 132

Bagian XIV — Keamanan Mendalam 135
Bab 52. Exit Program 135
Bab 53. Enkripsi dan Sertifikat 138
Bab 54. Pengerasan Sistem 140

Bagian XV — High Availability dan Disaster Recovery Mendalam 143
Bab 55. Memilih Arsitektur Ketersediaan 143
Bab 56. Operasi Cluster 145
Bab 57. Merancang Latihan DR yang Bermakna 148

Bagian XVI — Modul Core Banking 149
Bab 58. Customer Information File 149
Bab 59. Modul Deposito 151
Bab 60. Modul Kredit 155
Bab 61. General Ledger 158
Bab 62. Remittance dan Sistem Pembayaran 160
Bab 63. Pelaporan Regulator 163
Bab 64. Interface Antar Sistem 165
Bab 65. Batch di Luar EOD 168

Bagian XVII — Jaringan dan TCP/IP 171
Bab 66. Fondasi TCP/IP 172
Bab 67. Konfigurasi Jaringan di IBM i 177
Bab 68. Diagnosis Jaringan 181
Bab 69. Layanan Jaringan 186
Bab 70. Pemantauan Jaringan Berkelanjutan 190

Bagian XVIII — Modernisasi Aplikasi 192
Bab 71. Merancang REST API di atas Core Banking 192
Bab 72. Open Source di IBM i 196
Bab 73. DevOps untuk IBM i 199

Bab 74. Mengukur Keberhasilan Modernisasi 204

Bagian XIX — Manajemen Proyek Teknis 205

Bab 75. Proyek Upgrade Rilis OS 205

Bab 76. Proyek Migrasi Perangkat Keras 207

Bab 77. Cutover 209

Bagian XX — Performance Engineering Mendalam 212

Bab 78. Wait Accounting 212

Bab 79. Menyusun Studi Kinerja 217

Bagian XXI — Operasi Tingkat Sistem 219

Bab 80. IPL 219

Bab 81. Restricted State 222

Bab 82. HMC dan PowerVM 223

Bab 83. Pembersihan Sistem Terjadwal 224

Bagian XXII — BRMS dan Manajemen Media 227

Bab 84. Konsep BRMS 227

Bab 85. Prosedur Pemulihan 231

Bagian XXIII — CCSID, Unicode, dan Pertukaran Data 234

Bab 86. Memahami CCSID 234

Bab 87. Manajemen Keluaran dan Cetak 238

Bab 88. Pertukaran Data dengan Sistem Lain 240

Bagian XXIV — Studi Kasus Troubleshooting 242

Bab 89. Kasus Kinerja 242

Bab 90. Kasus Data 243

Bab 91. Kasus Operasional 245

Bab 92. Pola yang Berulang 247

Bagian XXV — Operasi Tim 248

Bab 93. Jaga dan Eskalasi 248

Bab 94. Dokumentasi yang Bertahan 250

Bab 95. Membangun Kemampuan Tim 251

Bagian XXVI — DDS dan Antarmuka 5250 252

Bab 96. Struktur DDS 253

Bab 97. Display File 254

Bab 98. Printer File 258

Bagian XXVII — Respons Insiden Keamanan 259

Bab 99. Prinsip Penanganan 259

Bab 100. Penahanan dan Pemulihan 262

Bagian XXVIII — Migrasi dan Konversi Data 265

Bab 101. Merancang Migrasi Data 265

Bab 102. Konversi Objek Basis Data 270

Bagian XXIX — Integrasi Berbasis Pesan 272
Bab 103. Data Queue 272
Bab 104. MQ dan Middleware 274
Bagian XXX — Manajemen Vendor dan Dukungan 277
Bab 105. Dukungan IBM 277
Bab 106. Vendor Aplikasi 278
Bagian XXXI — Latihan dan Studi Mandiri 280
Bab 107. Soal Pemahaman Dasar 280
Bab 108. Soal Situasi Operasional 281
Bab 109. Soal Rancangan dan Analisis 281
Bab 110. Kunci Jawaban 282
Bab 111. Proyek Latihan 285
Bagian XXXII — Membangun Pemantauan Terpadu 286
Bab 112. Arsitektur Pemantauan 286
Bab 113. Integrasi dengan Sistem Luar 290
Bagian XXXIII — Lisensi, Biaya, dan Anggaran 292
Bab 114. Memahami Struktur Lisensi 292
Bab 115. Menyusun Anggaran 294
Bab 116. Total Biaya Kepemilikan 296
Bagian XXXIV — Referensi Perintah CL 297
Bab 117. Perintah per Kategori 297
Bab 118. Indeks IBM i Services 304
Bagian XXXV — Penutup 308
Bab 119. Peta Jalan Pengembangan Diri 308
Bab 120. Penutup 308
Bagian XXXVI — Studi Kasus Lanjutan 309
Bab 121. Kasus Penyimpanan dan Kapasitas 309
Bab 122. Kasus Aplikasi dan Data 311
Bab 123. Kasus Sistem dan Infrastruktur 313
Bab 124. Pola dari Dua Puluh Kasus 315
Bagian XXXVII — Pendalaman End of Day 316
Bab 125. Merancang Fase yang Aman Diulang 316
Bab 126. Skenario EOD Bermasalah 319
Bab 127. Mencegah EOD Bermasalah 321
Bagian XXXVIII — Pendalaman Debug dan Analisis Program 323
Bab 128. Membaca Kode yang Tidak Anda Tulis 323
Bab 129. Teknik Debug Lanjutan 326
Bagian XXXIX — Katalog Runbook Lanjutan 329
Bab 130. Runbook Basis Data 329

Bab 131. Runbook Aplikasi 331
Bab 132. Runbook Infrastruktur 333
Bab 133. Runbook Keamanan 335
Bab 134. Memelihara Katalog Runbook 338

Bagian XL — Pendalaman Prosedur Pemulihan Bencana 339
Bab 135. Prosedur Switchover Terencana 340
Bab 136. Prosedur Failover Tak Terencana 344
Bab 137. Dokumen Pemulihan yang Harus Dicetak 347

Bagian XLI — Algoritma Operasional 348
Bab 138. Algoritma Pemrosesan Batch 348
Bab 139. Algoritma Diagnosis 352
Bab 140. Algoritma Verifikasi 355

Bagian XLII — Pemrograman RPG dan COBOL 357
Bab 141. Referensi Bahasa RPG 358
Bab 142. ILE COBOL untuk IBM i 361
Bab 143. Interoperabilitas Antar Bahasa 367
Bab 144. Memilih dan Memelihara 370

Bagian XLIII — Praktikum 371
Bab 145. Praktikum Dasar 372
Bab 146. Praktikum Pemulihan 374
Bab 147. Praktikum Kinerja 375
Bab 148. Praktikum End of Day 377
Bab 149. Praktikum Jaringan dan Keamanan 378

Bagian XLIV — Latihan Soal dan Pekerjaan Rumah 380
Bab 150. Soal Latihan Lanjutan 380
Bab 151. Pekerjaan Rumah 381

Bagian XLV — Proyek Studi Mandiri 383
Bab 152. Proyek Tingkat Menengah 383
Bab 153. Proyek Tingkat Lanjut 384

Bagian XLVI — Glosarium 385

Bagian XLVII — Anatomi Insiden 392
Bab 154. Narasi Pertama: Malam Ketika Akrual Hampir Berjalan Dua Kali 392
Bab 155. Narasi Kedua: Cabang yang Lambat Selama Tiga Pekan 397
Bab 156. Narasi Ketiga: Backup yang Enam Bulan Tidak Lengkap 399
Bab 157. Narasi Keempat: Uji Pemulihan yang Gagal Sebagian 402
Bab 158. Benang Merah Empat Narasi 406

Bagian XLVIII — Esai Teknis 407
Bab 159. Mengapa Integritas Dibangun ke Dalam, Bukan Ditambahkan 407
Bab 160. Ekonomi Keandalan 409

Bab 161. Tentang Dokumentasi yang Sesungguhnya Dipakai 411

Bagian XLIX — Setahun dalam Hidup System Engineer 413

Bab 162. Kuartal Pertama 413

Bab 163. Kuartal Kedua 415

Bab 164. Kuartal Ketiga 416

Bab 165. Kuartal Keempat 417

Bab 166. Irama Mingguan dan Harian 419

Bagian L — Pendalaman Pemrograman CL 420

Bab 167. Membuat Perintah Sendiri 421

Bab 168. Memanggil API Sistem dari CL 424

Bab 169. Membangun Pustaka Utilitas CL 427

Bagian LI — Pendalaman Terapan 430

Bab 170. Pendalaman Exit Program 430

Bab 171. Pendalaman TLS dan Sertifikat 432

Bab 172. Pendalaman Pertukaran Data dan CCSID 434

Bab 173. Pendalaman Arsitektur Pesan 435

Bab 174. Pendalaman Hybrid Cloud 437

Bab 175. Pendalaman Migrasi Data 438

Bab 176. Pendalaman Cutover 440

Bagian LII — Pendalaman Db2 for i Lanjutan 441

Bab 177. Partisi Tabel 441

Bab 178. Materialized Query Table 443

Bab 179. Bekerja dengan Objek Besar 445

Bab 180. Penulisan Ulang Kueri oleh Optimizer 446

Bab 181. Kendali Sumber Daya Kueri 448

Bagian LIII — Pendalaman Identitas dan Akses 449

Bab 182. Mengelola Identitas dalam Skala Besar 449

Bab 183. Single Sign-On dan Kerberos 452

Bab 184. Manajemen Kata Sandi dan Kredensial 453

Bab 185. Tinjauan Akses Berkala 455

Lampiran 456

Lampiran A. Perintah CL Esensial 456

Lampiran B. System Values Kritikal 457

Lampiran C. Templat Runbook 457

Lampiran D. Checklist Harian System Engineer 458

Lampiran E. Checklist EOD 458

Lampiran F. Glosarium 459

Daftar Pustaka 460

Standar dan kerangka kerja 460

Arsitektur platform 460

Pemrograman 460
Keamanan 460
Dokumentasi IBM 460
Core banking 461
Regulasi Indonesia 461
Berkala dan komunitas 461
Catatan penyuntingan 461

Bagian LIV — Pendalaman Analisis Kinerja 461
Bab 186. Collection Services Secara Mendalam 461
Bab 187. Wait Accounting Terapan 465
Bab 188. Job Watcher dan Analisis Mendalam 468
Bab 189. Menyusun Studi Kinerja yang Meyakinkan 469

Bagian LV — Pendalaman Work Management 472
Bab 190. Merancang Subsystem dari Awal 472
Bab 191. Prestart Job dan Server Job 477
Bab 192. Penjadwalan Lanjutan 479

Bagian LVI — Pendalaman Ketersediaan Tinggi 481
Bab 193. Arsitektur PowerHA Secara Rinci 481
Bab 194. Db2 Mirror 485
Bab 195. Menguji Ketersediaan Secara Berkala 487

Bagian LVII — Pendalaman RPG Lanjutan 490
Bab 196. Prosedur dan Modularitas 490
Bab 197. Service Program dan Pengelolaan Versi 493
Bab 198. Penanganan Kesalahan yang Matang 495

Bagian LVIII — Tata Kelola dan Audit Lanjutan 498
Bab 199. Membangun Kerangka Kendali 498
Bab 200. Menghadapi Audit 504
Bab 201. Manajemen Risiko Teknologi 508

Bagian LIX — Katalog Runbook Tambahan 511
Bab 202. Runbook Kinerja 511
Bab 203. Runbook Integrasi 514
Bab 204. Runbook Data 516
Bab 205. Runbook Operasional 518
Bab 206. Indeks Runbook Lengkap

Bagian LX — Pendalaman DDS dan Layar 5250 529
Bab 207. Anatomi Display File 529
Bab 208. Subfile Secara Mendalam 537
Bab 209. Mengubah Layar Produksi dengan Aman 541
Bab 210. Printer File dan Laporan 543
Bab 211. Modernisasi Tampilan Tanpa Menulis Ulang 546

Bagian LXI — Anatomi Insiden Lanjutan 549
Bab 212. Malam Ketika Angka Cocok Tetapi Salah 549
Bab 213. Tiga Hari Mencari Sesuatu yang Tidak Ada 552
Bab 214. Backup yang Berhasil Setiap Malam Selama Delapan Bulan 554
Bab 215. Perbaikan yang Memperburuk Keadaan 557
Bab 216. Pola yang Muncul dari Delapan Narasi 559
Indeks Subjek 561

Handbook of System Engineering for IBM i (AS/400)

🏦 Arsitektur, Operasi, dan Core Banking

Romi Nur Ismanto

rominur.com · hello@rominur.com

Edisi Pertama · 18 September 2026

Tentang Buku Ini

Buku ini ditulis untuk satu peran spesifik: **System Engineer yang bertanggung jawab atas platform IBM i (AS/400) di lingkungan produksi kritikal**, khususnya perbankan. Bukan untuk programmer yang ingin belajar RPG, dan bukan untuk manajer yang ingin gambaran umum.

Sasaran pembaca

Pembaca	Bagian yang paling relevan
System Engineer baru di IBM i	I, II, III, IV
Operator senior yang naik ke peran engineer	IV, V, Lampiran
Application Support core banking	VI, terutama bab 31 dan 33
Arsitek dan technical lead	II, V, VII
Auditor TI dan risk officer	V bab 27, VI bab 35

Konvensi penulisan

- Perintah CL ditulis huruf kapital: `WRKACTJOB`, `DSPJOBLOG`
- Nama objek contoh memakai awalan netral: `APPLIB`, `DTALIB`, `JOBQ01`
- Placeholder ditulis dalam kurung siku: `WRKJOB JOB([nomor]/[user]/[nama])`
- Blok kode diberi label bahasa: CL, RPGLE, SQL, atau Bash
- Angka versi mengacu pada IBM i 7.6 kecuali disebutkan lain

Peringatan penting soal kerahasiaan

Seluruh contoh yang menyangkut core banking di Bagian VI bersifat **generik dan teranonimkan**. Tidak ada nama library, nama job, struktur file, atau parameter dari sistem produksi bank mana pun. Dokumentasi vendor core banking terikat perjanjian kerahasiaan, dan detail operasional harian sebuah bank termasuk informasi rahasia yang dilindungi ketentuan regulator.

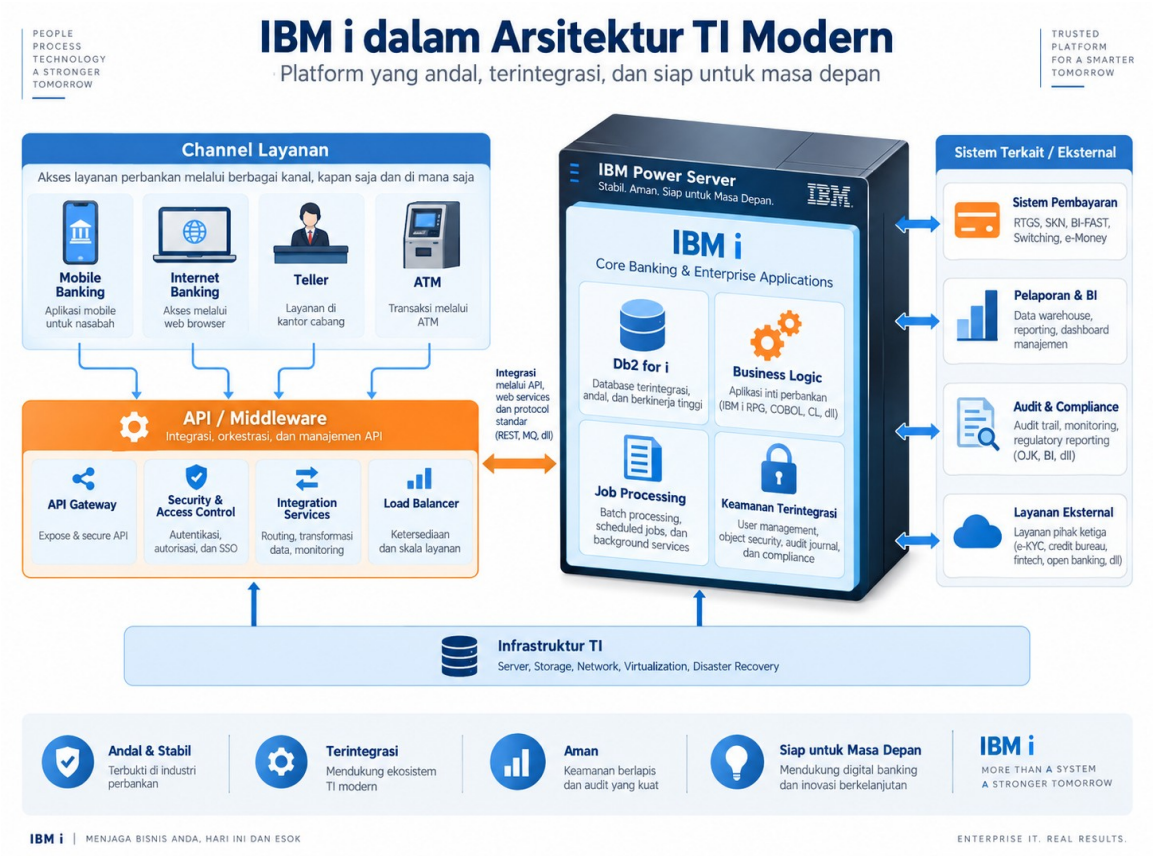
Jika Anda mengembangkan buku ini untuk publikasi, pastikan setiap contoh melewati tinjauan legal dan compliance institusi Anda terlebih dahulu.

Cara membaca

Pembaca baru sebaiknya berurutan dari Bagian I sampai IV. Pembaca berpengalaman bisa langsung ke Bagian V atau VI, tetapi Bab 10 (work management) adalah prasyarat untuk memahami hampir semua pembahasan operasional setelahnya.

🏗 Bagian I — Fondasi

🏗 Bab 1. Posisi IBM i dalam Arsitektur TI Modern



Gambar 1 — IBM i dalam arsitektur TI modern

IBM i bertahan bukan karena inersia, tetapi karena tiga sifat yang sulit ditiru platform lain: integritas transaksi bawaan, integrasi basis data di level sistem operasi, dan kompatibilitas biner lintas generasi perangkat keras selama lebih dari tiga dekade.

1.1 Salah paham yang paling umum

Anggapan	Kenyataan
"AS/400 itu mainframe"	Bukan. Ini midrange, turunan System/38, arsitektur sama sekali berbeda dari z/OS
"Sudah tidak didukung"	IBM i 7.6 rilis April 2025, hanya jalan di Power10 ke atas
"Hanya bisa layar hijau"	Ada REST API, Node.js, Python, Git, VS Code
"Tidak ada SQL"	Db2 for i adalah SQL engine penuh dengan optimizer modern
"Tidak bisa cloud"	Tersedia di IBM Power Virtual Server dan penyedia hosting Power

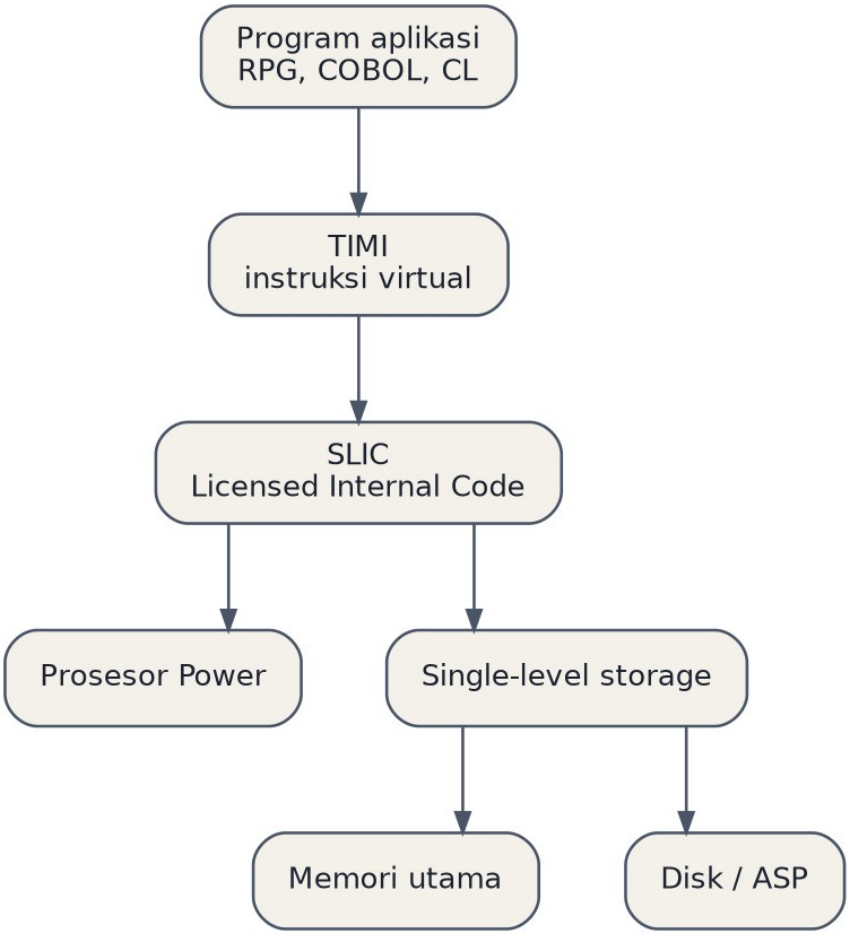
1.2 Tiga sifat pembeda

Object-based architecture. Di IBM i, semuanya adalah objek bertipe. Sebuah file adalah `*FILE`, program

adalah `*PGM`, antrean pekerjaan adalah `*JOBQ`. Sistem operasi menolak operasi yang tidak sah untuk tipe tersebut. Ini menutup seluruh kelas serangan yang lazim di platform berbasis file byte-stream.

Single-level storage. Memori utama dan disk adalah satu ruang alamat tunggal. Program tidak pernah tahu apakah datanya di RAM atau di piringan. Konsekuensi praktisnya: tidak ada manajemen memori manual, dan alamat objek stabil selamanya.

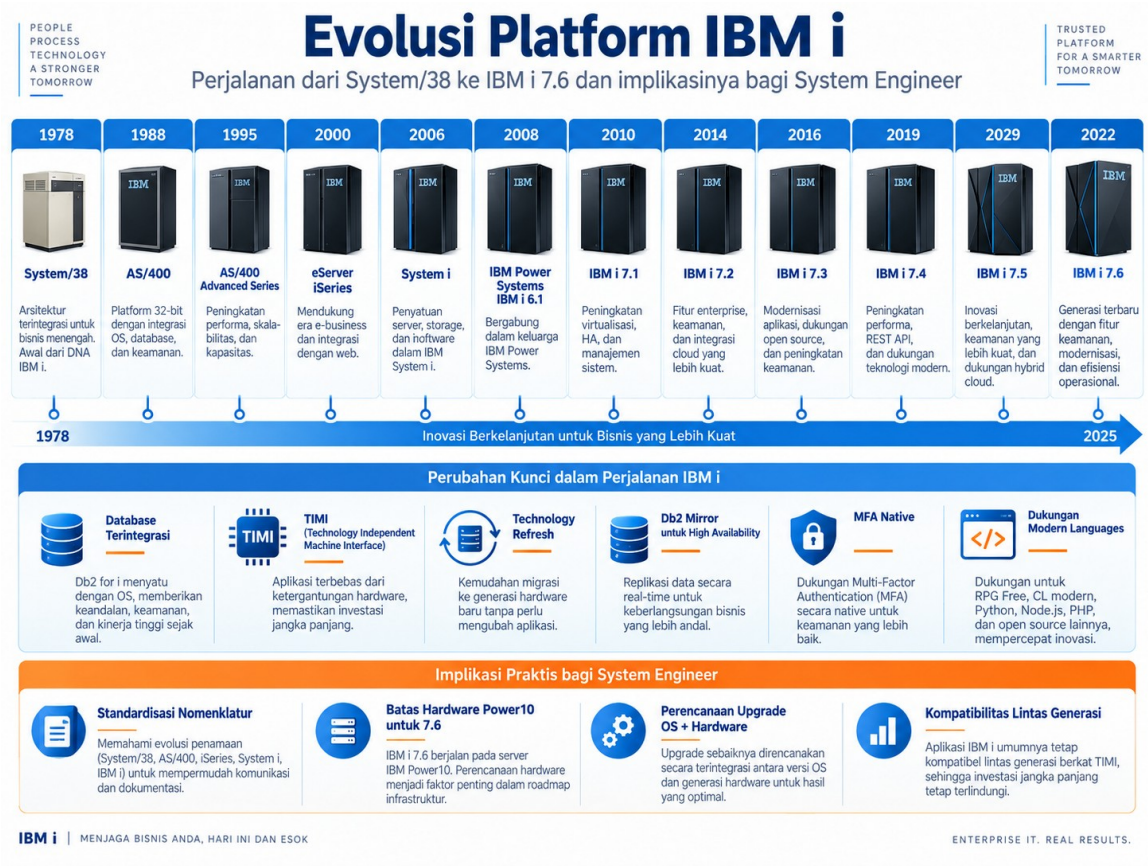
Technology Independent Machine Interface (TIMI). Program dikompilasi ke instruksi virtual, bukan instruksi prosesor. Saat perangkat keras berganti dari CISC ke PowerPC ke Power10, program lama tetap jalan tanpa kompilasi ulang.



Gambar 2

Lapisan TIMI adalah alasan program tahun 1990 masih berjalan di Power11 tanpa sentuhan.

🏗️ Bab 2. Evolusi Platform



Gambar 3 — Evolusi platform dari System/38 sampai IBM i 7.6

Tahun	Nama	Perubahan kunci
1978	System/38	Single-level storage, database terintegrasi, TIMI
1988	AS/400	Peluncuran, OS/400 V1
1995	AS/400 Advanced Series	Transisi CISC ke PowerPC 64-bit
2000	eServer iSeries	OS/400 V5, dukungan Linux
2006	System i	i5/OS V5R4
2008	IBM Power Systems	Nama OS jadi IBM i 6.1, konsolidasi dengan AIX
2010	IBM i 7.1	Technology Refresh diperkenalkan
2014	IBM i 7.2	Row and Column Access Control
2016	IBM i 7.3	Temporal table, Db2 Mirror dirintis
2019	IBM i 7.4	Db2 Mirror for i
2022	IBM i 7.5	Penghapusan dukungan program non-RISC lama
2025	IBM i 7.6	MFA native, hanya Power10 ke atas

Dua implikasi praktis untuk System Engineer:

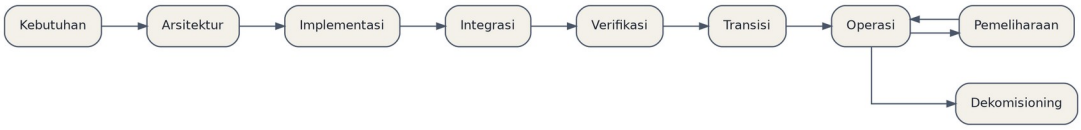
1. **Nomenklatur.** Dokumen internal Anda kemungkinan mencampur istilah AS/400, iSeries, dan IBM i untuk mesin yang sama. Standarkan satu istilah di seluruh dokumentasi.

2. **Batas perangkat keras.** IBM i 7.6 menuntut Power10. Rencana upgrade OS tidak bisa dipisahkan dari rencana refresh perangkat keras.

⚙️ Bab 3. Kerangka Rekayasa Sistem Buku Ini

Buku ini memetakan proses siklus hidup ISO/IEC/IEEE 15288:2023 ke pekerjaan nyata seorang System Engineer IBM i.

Proses 15288	Padanan di dunia IBM i	Bab
Stakeholder needs & requirements	Sizing, SLA, RTO/RPO	11
Architecture definition	Desain LPAR, ASP, topologi	12
Implementation	Instalasi OS, konfigurasi awal	13, 14
Integration	Koneksi kanal, API, middleware	36
Verification	UAT, uji beban, uji restore	32, 34
Transition	Cutover, migrasi data	40
Operation	Monitoring, EOD, incident	17, 31
Maintenance	PTF, TR, tuning	13, 22
Disposal	Dekomisioning, retensi arsip	40



Gambar 4

Siklus operasi dan pemeliharaan berputar bertahun-tahun; di situlah 90% waktu seorang System Engineer IBM i dihabiskan.

Bab 4. Peran dan Kompetensi System Engineer IBM i

4.1 Batas tanggung jawab

Aktivitas	SysEng	AppSupport	DBA	Operator	Security
Instalasi PTF	R/A	C	C	I	I
Tuning memory pool	R/A	C	C	I	—
Membuat user profile	R	I	—	—	A
Analisis EOD gagal	C	R/A	C	I	—
Restore library	R/A	C	C	I	I
Debug program RPG	I	R/A	C	—	—
Konfigurasi journaling	R/A	C	C	—	I
Uji DR tahunan	R/A	C	C	R	C

R = pelaksana, A = penanggung jawab, C = dikonsultasi, I = diberi tahu.

4.2 Kompetensi inti

1. **Wajib kuasai**
    - CL programming tingkat menengah, cukup untuk menulis utilitas operasional
    - Work management: subsystem, job queue, memory pool, routing entry
    - Backup dan recovery termasuk journaling dan BRMS
    - Membaca job log dan menelusuri message ID sampai akar masalah
    - SQL untuk Db2 for i, termasuk IBM i Services
2. **Sangat disarankan**
    - Dasar RPG ILE, cukup untuk membaca kode dan melakukan debug
    - PowerHA atau replikasi logis
    - Keamanan IBM i sampai level exit point
3. **Nilai tambah**
    - PowerVM, VIOS, dan storage SAN
    - Open source di IBM i: Node.js, Python, Git

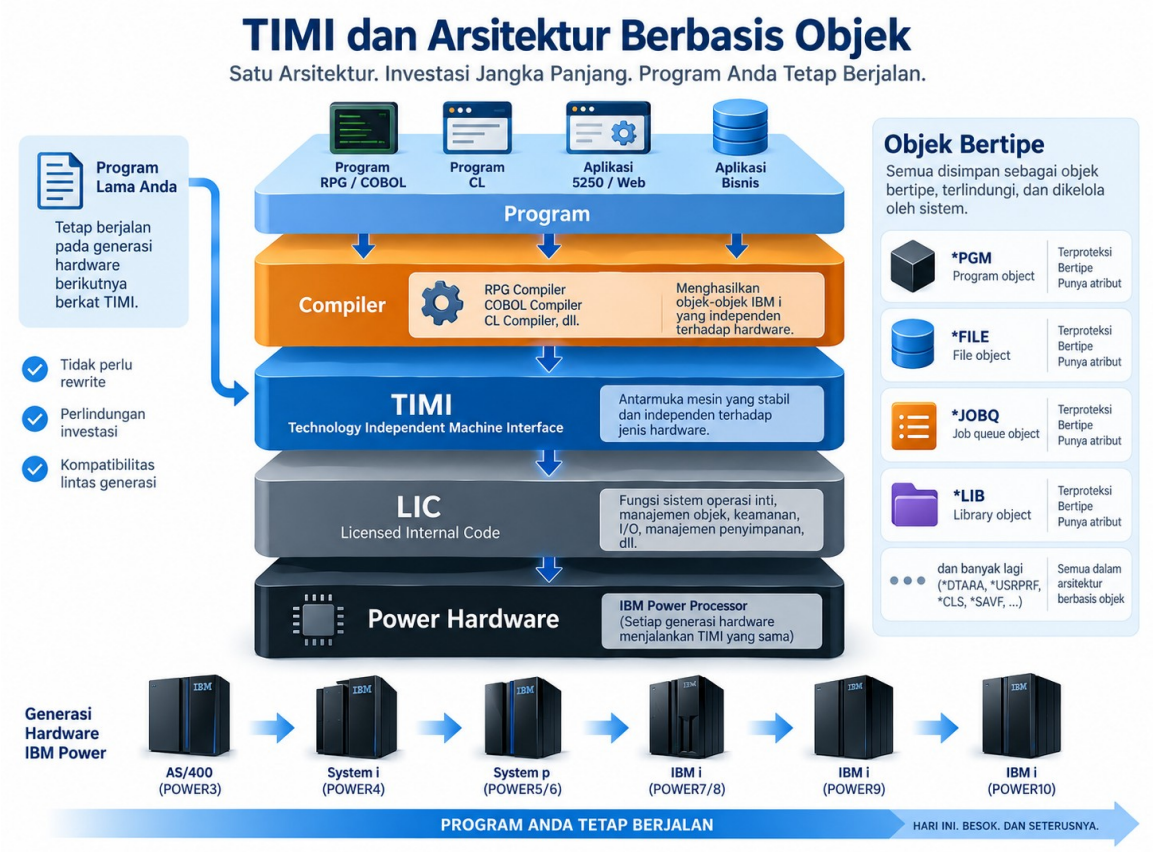
4.3 Masalah regenerasi SDM

Rata-rata usia praktisi IBM i di Indonesia tinggi, dan jalur masuk formal hampir tidak ada di kurikulum kampus. Konsekuensi untuk perencanaan Anda:

- Dokumentasikan setiap prosedur yang hanya ada di kepala satu orang
- Bangun pasangan senior-junior untuk setiap area kritikal
- Rekam sesi troubleshooting nyata sebagai materi pelatihan internal
- Rekrut dari latar belakang non-IBM i dan latih secara internal; pengalaman platform lain bukan penghalang

🏗️ Bagian II — Arsitektur Platform

👥 Bab 5. TIMI dan Arsitektur Berbasis Objek



Gambar 5 — TIMI dan arsitektur berbasis objek

Setiap entitas di IBM i adalah objek dengan tipe, dan tipe itu menentukan operasi apa yang sah. Tidak ada cara mengeksekusi sebuah file data sebagai program, karena sistem operasi menolaknya di level di bawah OS.

5.1 Tipe objek yang paling sering ditemui

Tipe	Isi	Perintah utama
`*LIB`	Wadah objek lain	`CRTLIB`, `DSPLIB`
`*FILE`	Tabel, file fisik/logis, printer file, display file	`DSPFD`, `DSPFFD`
`*PGM`	Program terkompilasi	`CRTBNDRPG`, `CALL`
`*SRVPGM`	Service program ILE	`CRTSRVPGM`, `DSPSRVPGM`
`*MODULE`	Modul ILE sebelum di-bind	`CRTRPGMOD`
`*JOBQ`	Antrean pekerjaan batch	`WRKJOBQ`
`*OUTQ`	Antrean keluaran cetak	`WRKOUTQ`
`*SBSD`	Deskripsi subsystem	`WRKSBS`, `STRSBS`
`*USRPRF`	Profil pengguna	`CRTUSRPRF`, `DSPUSRPRF`
`*JRN` / `*JRNRCV`	Journal dan receiver	`WRKJRN`, `DSPJRN`

Tipe	Isi	Perintah utama
*DTAARA	Area data	CRTDTAARA, DSPDTAARA
*MSGQ	Antrean pesan	WRKMSGQ, DSPMSG

Memeriksa isi library dan tipe objeknya:

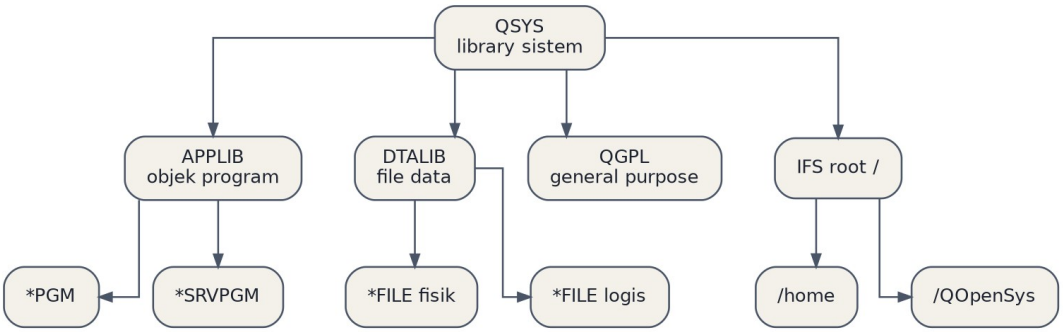
```
/* Semua objek di APPLIB */
DSPOBJD OBJ(APPLIB/*ALL) OBJTYPE(*ALL) DETAIL(*FULL)

/* Hanya program, keluarkan ke outfile untuk dianalisis dengan SQL */
DSPOBJD OBJ(APPLIB/*ALL) OBJTYPE(*PGM) DETAIL(*FULL) +
        OUTPUT(*OUTFILE) OUTFILE(QTEMP/OBJLIST)
```

Lalu analisis dengan SQL:

```
SELECT ODOBNM AS nama_objek,
       ODOBTP AS tipe,
       ODOBSZ AS ukuran_byte,
       ODCDAT AS tgl_buat,
       ODLUSD AS terakhir_dipakai
  FROM QTEMP.OBJLIST
 WHERE ODLUSD < CURRENT DATE - 2 YEARS
 ORDER BY ODOBSZ DESC;
```

Query ini menemukan program yang tidak pernah dipanggil dua tahun terakhir — titik awal yang baik untuk pembersihan objek warisan.

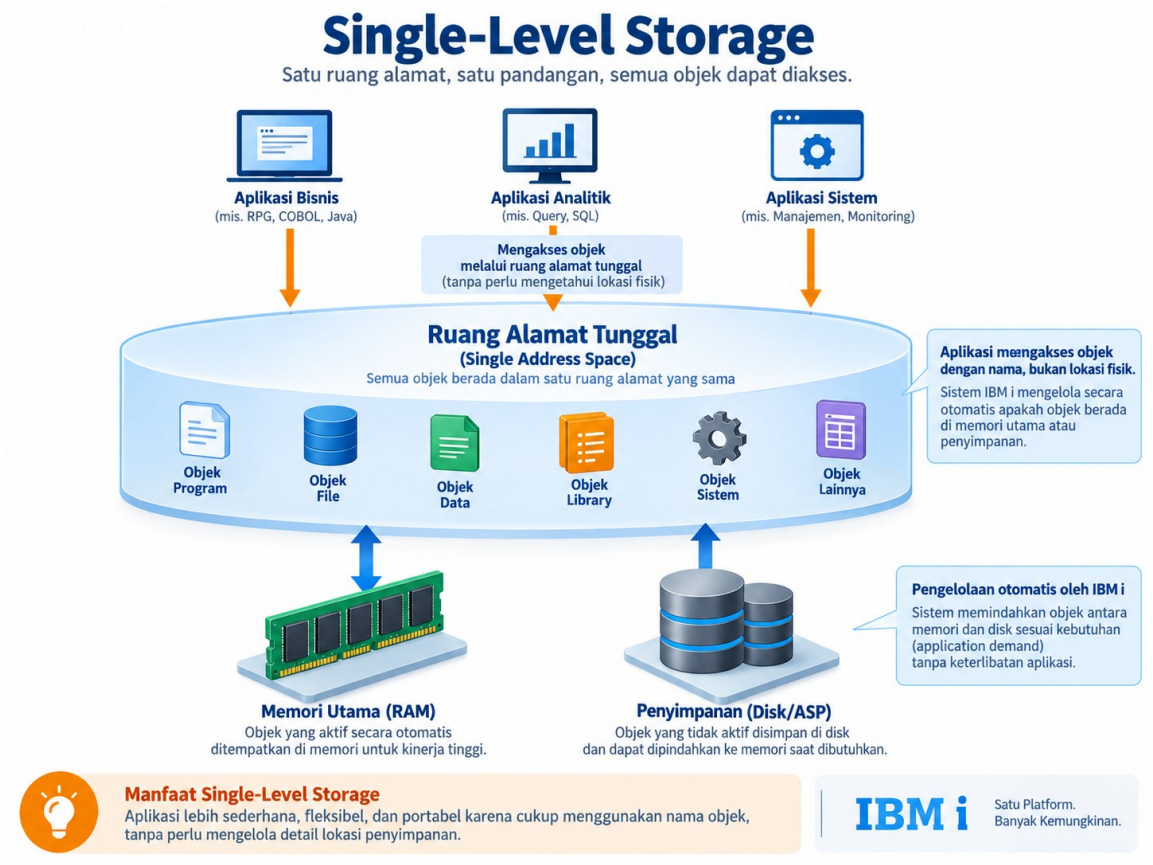
5.2 Struktur hierarki penamaan



Gambar 6

QSYS adalah satu-satunya library yang bisa memuat library lain. Struktur library flat, hanya satu tingkat — berbeda dari direktori bersarang di IFS.

Bab 6. Single-Level Storage



Gambar 7 — Single-level storage: satu ruang alamat tunggal

Semua disk dalam satu ASP diperlakukan sebagai satu kolam penyimpanan tunggal. Sistem menyebarkan data ke seluruh disk unit secara otomatis untuk memaksimalkan paralelisme I/O.

6.1 Konsekuensi operasional

1. **Anda tidak memilih disk mana yang dipakai objek.** Sistem yang memutuskan. Ini menghilangkan seluruh kategori pekerjaan tuning manual yang lazim di platform lain.
2. **Menambah disk unit langsung menambah throughput.** Semakin banyak arm, semakin banyak operasi I/O paralel.
3. **Kehilangan satu disk berpotensi merusak seluruh ASP.** Inilah alasan RAID dan mirroring bukan opsional.
4. **Persentase pemakaian disk adalah metrik kesehatan kritikal.** Di atas 80% kinerja mulai turun; di atas 90% sistem berisiko.

Memeriksa status disk:

```
WRKDSKSTS
WRKSYSSTS
DSPSYSSTS OUTPUT(*PRINT)
```

Atau lewat SQL, yang lebih mudah dijadwalkan:

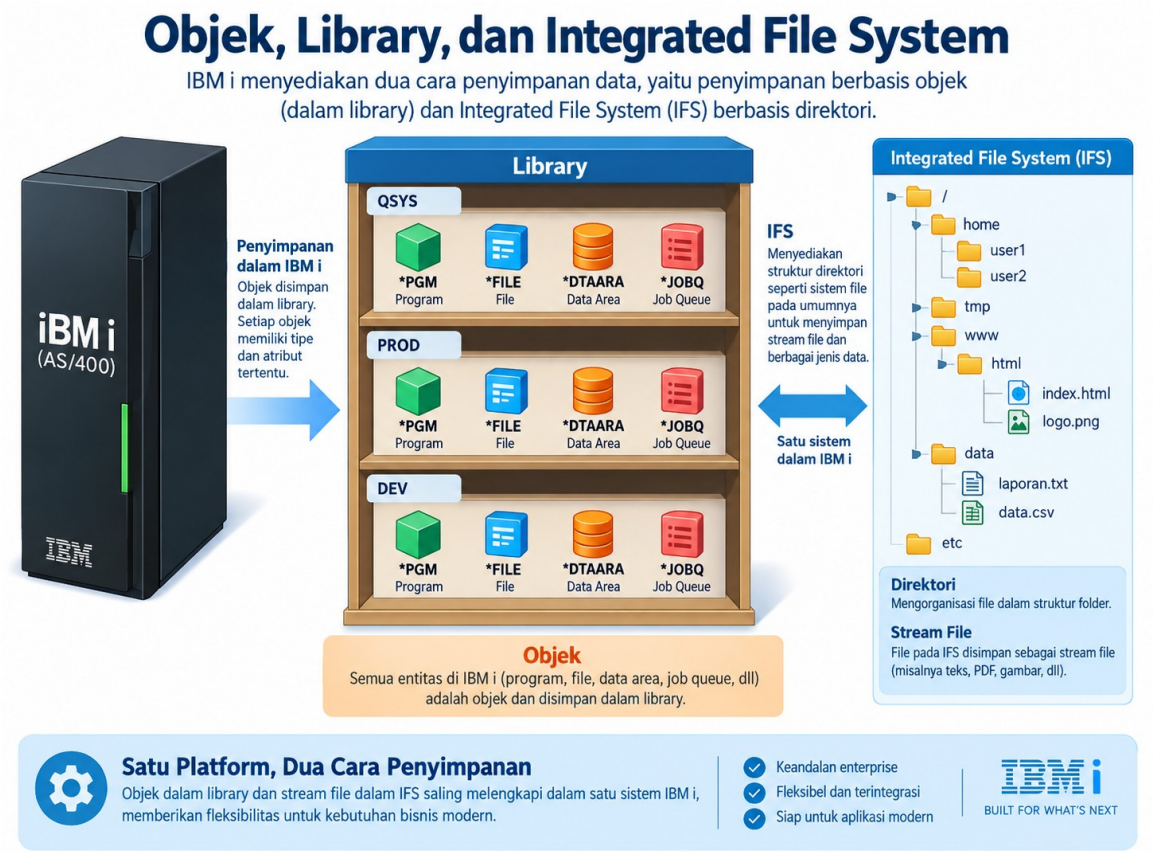
```
SELECT UNIT_NUMBER,
       UNIT_TYPE,
       UNIT_STORAGE_CAPACITY / 1024 / 1024 / 1024 AS kapasitas_gb,
       PERCENT_USED,
       UNIT_MEDIA_TYPE
  FROM QSYS2.SYSDISKSTAT
 ORDER BY PERCENT_USED DESC;
```

6.2 Ambang batas yang disarankan

Pemakaian ASP	Status	Tindakan
Di bawah 70%	Normal	Pantau tren bulanan
70–80%	Perhatian	Rencanakan penambahan kapasitas
80–85%	Peringatan	Bersihkan objek, jalankan RCLSTG terjadwal
85–90%	Kritis	Eskalasi, tambah disk segera
Di atas 90%	Darurat	Risiko kegagalan job, sistem melambat drastis

Untuk lingkungan core banking, saya sarankan ambang peringatan diturunkan ke 75%, karena lonjakan konsumsi saat End of Month bisa mencapai belasan persen dalam satu malam.

Bab 7. Objek, Library, dan Integrated File System



Gambar 8 — Objek, library, dan Integrated File System

IBM i punya dua sistem penyimpanan yang hidup berdampingan: library-based (QSYS.LIB) dan direktori

bergaya Unix (IFS).

7.1 Kapan pakai yang mana

Kebutuhan	Tempat
Tabel database, program, file cetak	Library QSYS.LIB
File CSV pertukaran data	IFS, misal `/interface/out`
Aplikasi Node.js atau Python	IFS `/QOpenSys`
Log aplikasi modern	IFS
Sertifikat SSL	IFS `/QIBM/UserData/ICSS`
File yang dibaca RPG lawas	Library

7.2 Library list — sumber kesalahan paling sering

Library list menentukan urutan pencarian objek. Kesalahan urutan library adalah penyebab nomor satu insiden "program jalan di UAT tapi gagal di produksi".

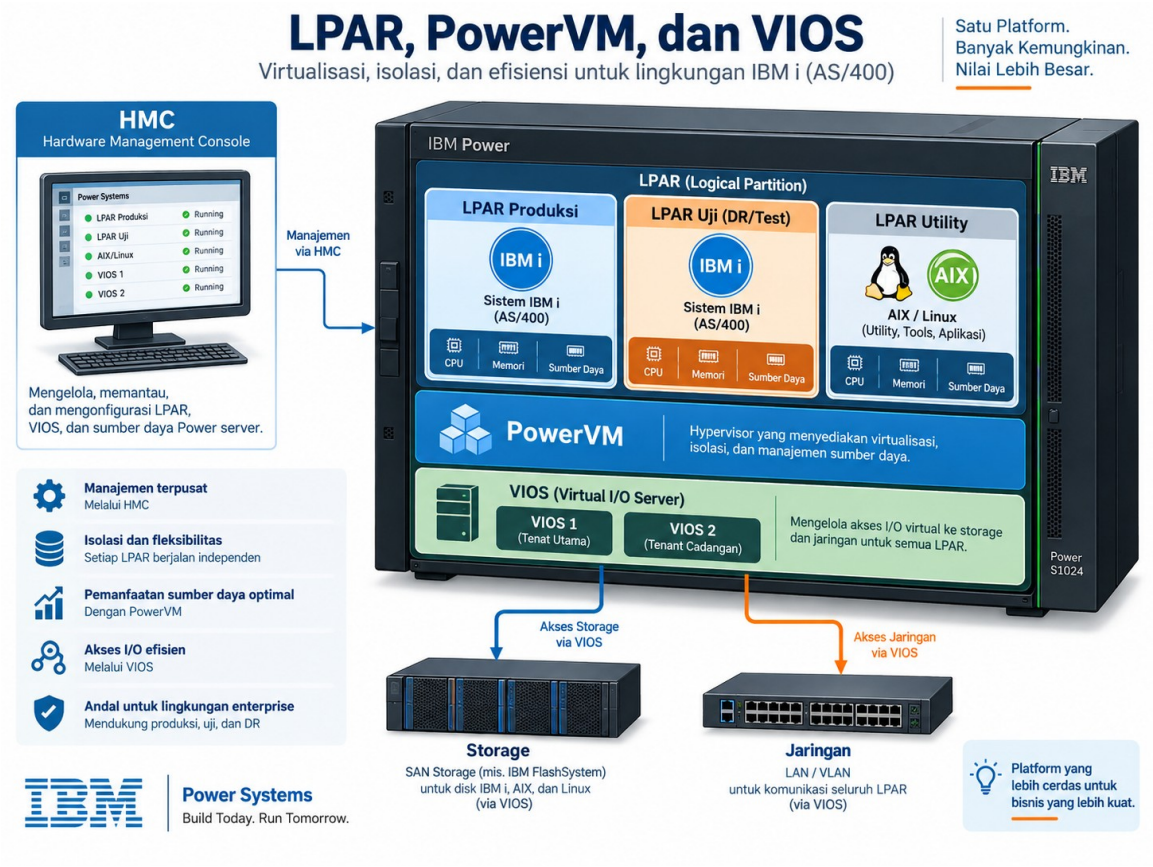
```
/* Melihat library list job saat ini */
DSPLIBL

/* Mengubah user portion */
CHGLIBL LIBL(APPLIB DTALIB QGPL QTEMP)

/* Menambah satu library di posisi tertentu */
ADDLIBLE LIB(TESTLIB) POSITION(*FIRST)
```

Struktur library list terdiri dari empat bagian, dicari berurutan:

Gambar 9

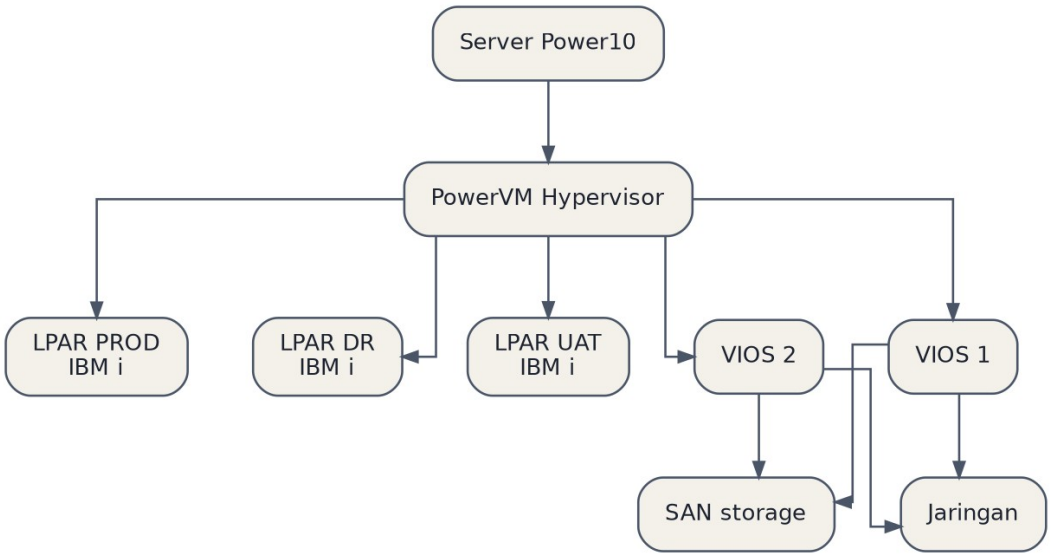
Praktik yang aman: jangan pernah mengandalkan library list untuk objek kritikal. Gunakan kualifikasi eksplisit `APPLIB/PROGRAM` di program produksi.

Bab 8. LPAR, PowerVM, dan VIOS



Gambar 10 — LPAR, PowerVM, dan VIOS

8.1 Model partisi



Gambar 11

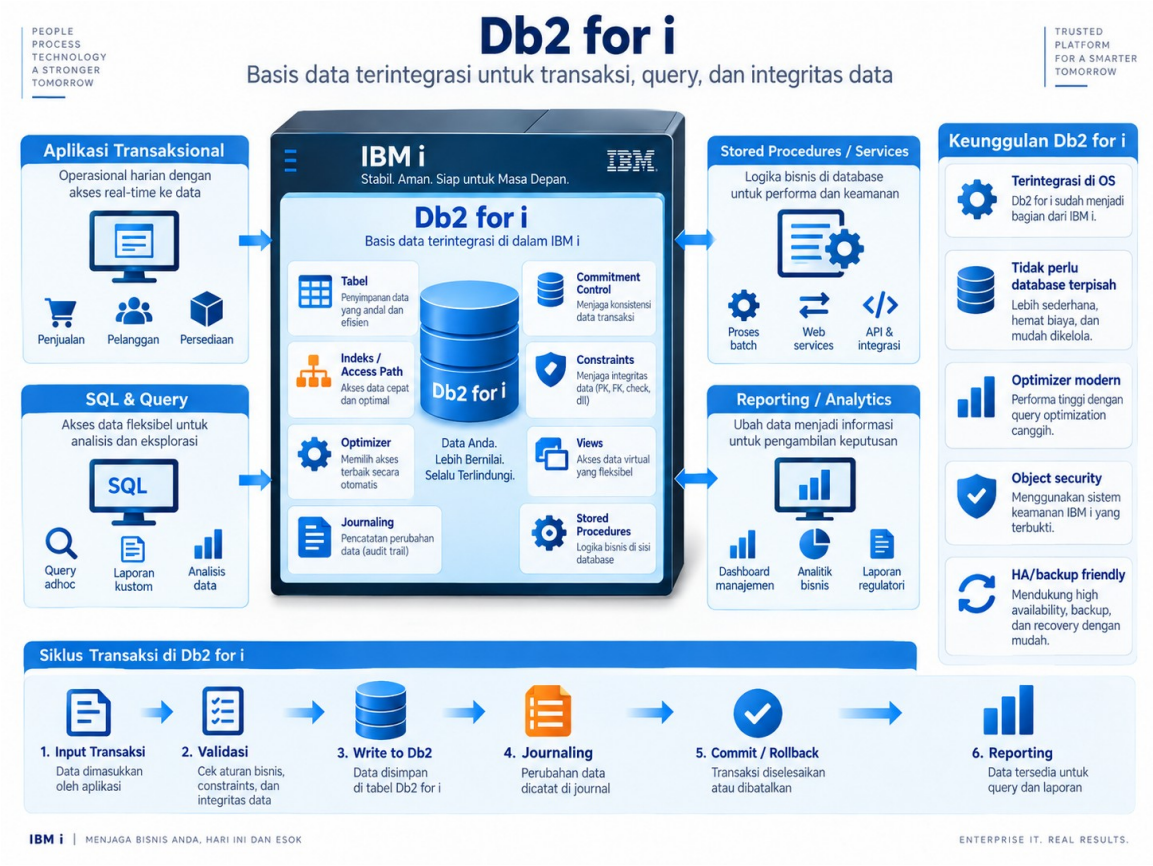
Dua VIOS dipasang berpasangan agar tidak ada titik kegagalan tunggal pada jalur I/O.

8.2 Parameter partisi yang perlu dipahami

Parameter	Arti	Kesalahan umum
Desired processing units	Alokasi CPU dalam pecahan core	Diberi terlalu kecil untuk beban batch
Virtual processors	Jumlah core logis yang terlihat OS	Rasio terhadap processing units terlalu tinggi
Capped / uncapped	Boleh meminjam CPU idle atau tidak	Produksi dibuat capped tanpa alasan
Weight	Prioritas saat berebut CPU	Semua partisi diberi weight sama
Memory	Alokasi RAM	Tidak menyisakan ruang untuk hypervisor

Untuk core banking, partisi produksi sebaiknya uncapped dengan weight tertinggi, agar bisa menyerap kapasitas idle partisi lain saat End of Day.

🗄️ Bab 9. Db2 for i



Gambar 12 — Db2 for i sebagai basis data terintegrasi

Basis data bukan produk terpisah yang dipasang, melainkan bagian dari sistem operasi. Tidak ada instans untuk dikelola, tidak ada listener untuk dinyalakan.

9.1 Dua wajah yang sama

Satu tabel bisa diakses dengan dua paradigma sekaligus:

```
-- Wajah SQL
CREATE TABLE DTALIB.REKENING (
    NO_REK      CHAR(16)      NOT NULL,
    NAMA        VARCHAR(60)   NOT NULL,
    KD_PRODUK   CHAR(4)       NOT NULL,
    SALDO       DECIMAL(17,2) NOT NULL DEFAULT 0,
    STATUS      CHAR(1)       NOT NULL DEFAULT 'A',
    TGL_BUKA    DATE          NOT NULL,
    PRIMARY KEY (NO_REK)
);

CREATE INDEX DTALIB.REKENING_IX1
    ON DTALIB.REKENING (KD_PRODUK, STATUS);
```

```
/* Wajah native: tabel yang sama terlihat sebagai *FILE */
DSPFD FILE(DTALIB/REKENING)
DSPFFD FILE(DTALIB/REKENING)
RUNQRY QRYFILE((DTALIB/REKENING))
```

Program RPG lawas membacanya dengan native I/O; program baru memakai embedded SQL. Keduanya menulis ke data yang sama, dengan penguncian record yang dikoordinasikan sistem.

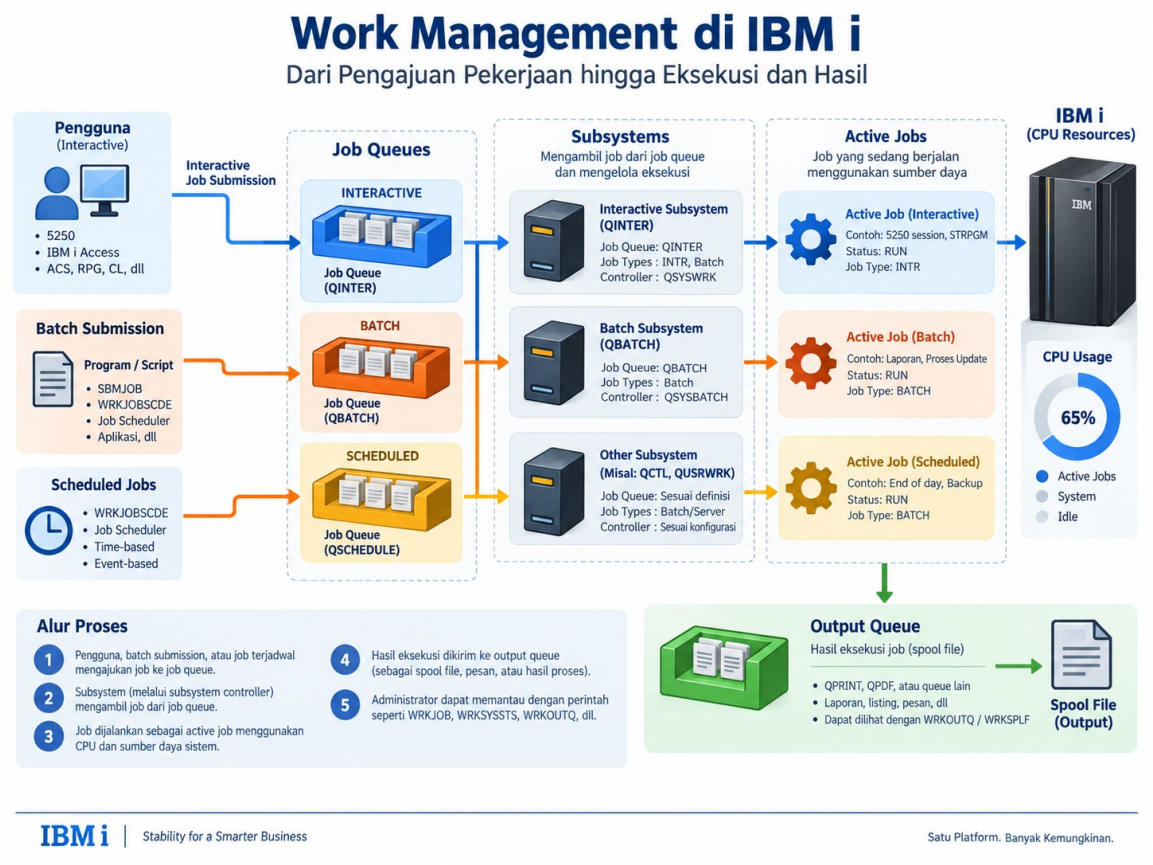
9.2 IBM i Services — administrasi lewat SQL

Ini pergeseran terpenting satu dekade terakhir bagi System Engineer. Hampir semua informasi sistem kini bisa dikueri sebagai tabel.

```
-- Job aktif dengan CPU tertinggi
SELECT JOB_NAME, AUTHORIZATION_NAME, SUBSYSTEM,
       ELAPSED_CPU_PERCENTAGE, ELAPSED_TOTAL_DISK_IO_COUNT
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
           DETAILED_INFO => 'ALL'))
 WHERE ELAPSED_CPU_PERCENTAGE > 1
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC
 FETCH FIRST 20 ROWS ONLY;

-- Profil pengguna dengan wewenang khusus berbahaya
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       LAST_USED_TIMESTAMP
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%'
   AND STATUS = '*ENABLED'
 ORDER BY LAST_USED_TIMESTAMP;

-- PTF group yang tertinggal dari level terbaru
SELECT PTF_GROUP_NAME, PTF_GROUP_LEVEL,
       PTF_GROUP_STATUS, PTF_GROUP_DESCRIPTION
  FROM QSYS2.GROUP_PTF_INFO
 WHERE PTF_GROUP_STATUS <> 'INSTALLED';
```

Simpan kueri semacam ini sebagai skrip yang dijadwalkan harian, keluarkan ke tabel riwayat, dan Anda punya basis monitoring tanpa perangkat tambahan.

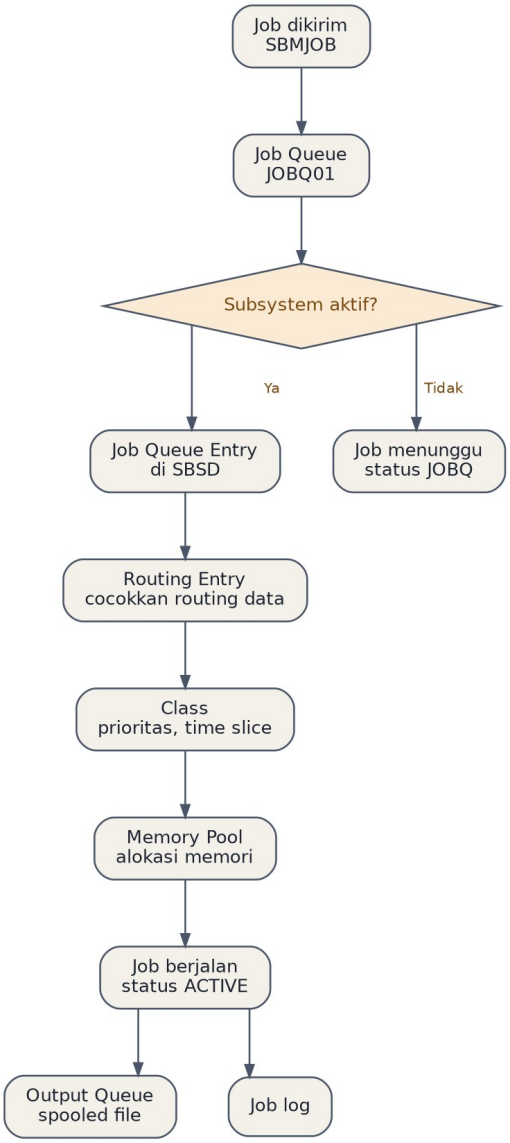
Bab 10. Work Management



Gambar 13 — Alur work management dari pengajuan job sampai keluaran

Ini bab prasyarat. Tanpa memahami work management, seluruh pembahasan kinerja dan End of Day di bagian berikutnya tidak akan masuk akal.

10.1 Alur sebuah job dari lahir sampai selesai



Gambar 14

10.2 Komponen dan perannya

Komponen	Peran	Perintah
Subsystem (`*SBSD`)	Wadah tempat job berjalan	`WRKSBS`, `STRSBS`, `ENDSBS`
Job queue (`*JOBQ`)	Antrean job batch menunggu giliran	`WRKJOBQ`, `HLDJOBQ`
Job queue entry	Menghubungkan JOBQ ke subsystem, atur MAXACT	`ADDJOBQE`, `CHGJOBQE`
Routing entry	Menentukan program awal dan class	`ADDRTGE`
Class (`*CLS`)	Prioritas run, time slice, default wait	`CRTCLS`, `CHGCLS`
Memory pool	Kolam memori tempat job dijalankan	`WRKSYSSTS`, `CHGSHRPOOL`

10.3 Subsystem standar dan peruntukannya

Subsystem	Peruntukan
`QINTER`	Sesi interaktif 5250
`QBATCH`	Job batch umum
`QCMN`	Komunikasi
`QSERVER`	File server, database server
`QUSRWRK`	Job server pengguna
`QSYSWRK`	Job sistem
`QHTTPSVR`	Web server

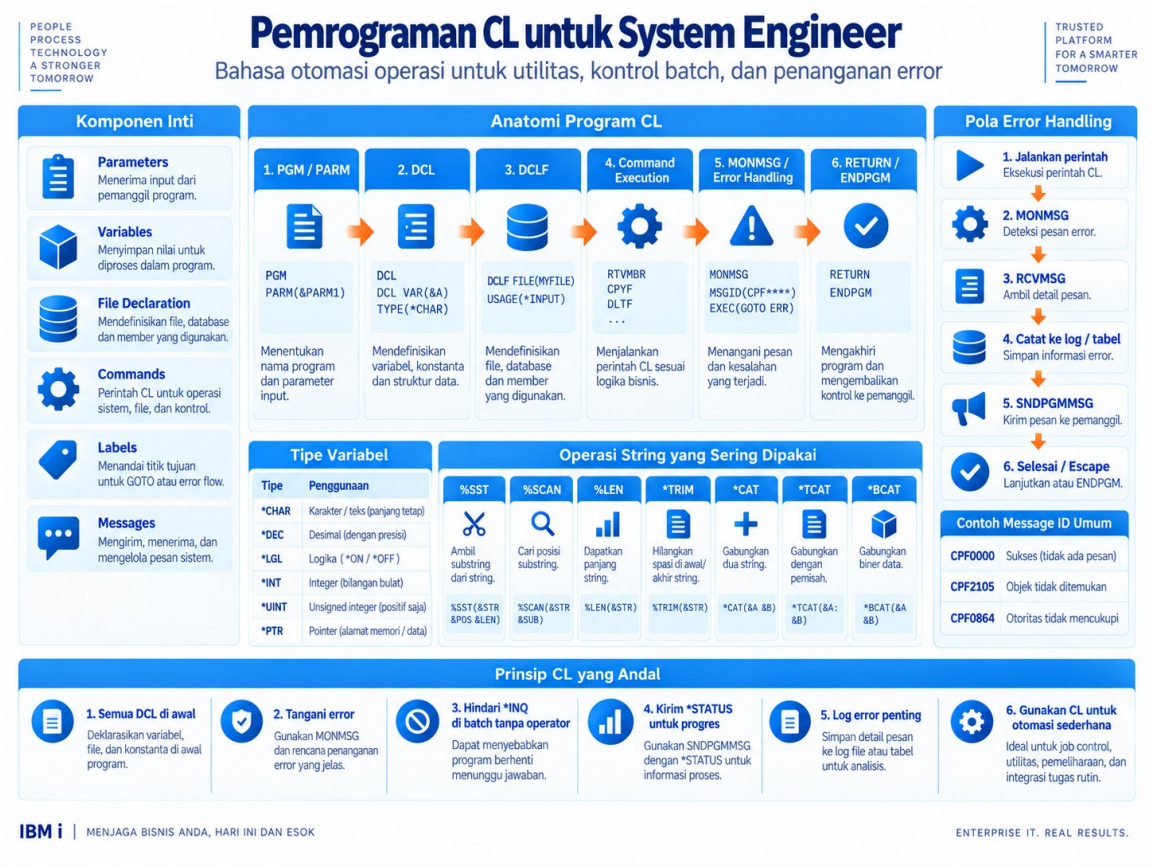
Di lingkungan core banking, praktik yang baik adalah membuat subsystem terpisah untuk beban End of Day, misalnya `EODSBS`, dengan memory pool khusus. Alasannya dibahas di Bab 34.

10.4 Contoh: membuat subsystem batch terpisah

```
/* 1. Buat deskripsi subsystem */
CRTSBSD SBSD(QGPL/EODSBS) POOLS((1 *BASE) (2 *SHRPOOL2)) +
        TEXT('Subsystem khusus End of Day')

/* 2. Buat class dengan prioritas lebih tinggi dari batch biasa */
CRTCLS CLS(QGPL/EODCLS) RUNPTY(30) TIMESLICE(5000) +
       DFTWAIT(60) TEXT('Class untuk job EOD')

/* 3. Buat job queue */
CRTJOBQ JOBQ(QGPL/EODJOBQ) TEXT('Antrean job EOD')

/* 4. Hubungkan job queue ke subsystem, maksimal 4 job serentak */
ADDJOBQE SBSD(QGPL/EODSBS) JOBQ(QGPL/EODJOBQ) MAXACT(4) SEQNBR(10)

/* 5. Tambahkan routing entry */
ADDRTGE SBSD(QGPL/EODSBS) SEQNBR(10) CMPVAL(*ANY) +
        PGM(QSYS/QCMD) CLS(QGPL/EODCLS) POOLID(2)

/* 6. Alokasikan memori ke shared pool 2 */
CHGSHRPOOL POOL(*SHRPOOL2) SIZE(8000000) ACTLVL(12)

/* 7. Jalankan */
STRSBS SBSD(QGPL/EODSBS)
```

Perhatikan `MAXACT(4)`: ini membatasi berapa job EOD berjalan bersamaan. Nilai terlalu tinggi menyebabkan job saling berebut memori dan justru memperlambat keseluruhan batch window.

💻 Bab 10A. Pemrograman CL untuk System Engineer



Gambar 15 — Anatomi dan pola program CL

CL bukan bahasa aplikasi. Ia bahasa otomasi operasi, dan seorang System Engineer yang tidak bisa menulis CL akan selamanya bergantung pada orang lain untuk setiap tugas berulang.

10A.1 Struktur program CL

```
PGM PARM(&PARM1 &PARM2)

  /* 1. Deklarasi variabel - harus di awal, sebelum perintah apa pun */
  DCL VAR(&PARM1)  TYPE(*CHAR) LEN(10)
  DCL VAR(&PARM2)  TYPE(*DEC)  LEN(5 0)
  DCL VAR(&HITUNG) TYPE(*DEC)  LEN(7 0) VALUE(0)
  DCL VAR(&PESAN)  TYPE(*CHAR) LEN(132)

  /* 2. Deklarasi file, maksimal 5 per program */
  DCLF FILE(DTALIB/REKENING)

  /* 3. Badan program */
  CHGVAR VAR(&HITUNG) VALUE(&HITUNG + 1)

  /* 4. Penanganan error global */
  MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(GAGAL))

  RETURN

GAGAL:
  CHGVAR VAR(&PESAN) VALUE('Program gagal')
```

```
   SNDPGMMSG MSG(&PESAN) MSGTYPE(*ESCAPE)

ENDPGM
```

Urutan ini mengikat. Semua `DCL` harus mendahului semua `DCLF`, dan keduanya harus mendahului perintah eksekusi pertama. Kompiler menolak urutan lain.

10A.2 Tipe variabel

Tipe	Panjang	Kegunaan
`*CHAR`	1–32767	Teks, nama objek, pesan
`*DEC`	(digit desimal)	Angka, penghitung
`*LGL`	1	Benar `'1'` atau salah `'0'`
`*INT`	2, 4, 8	Bilangan bulat, lebih efisien
`*UINT`	2, 4, 8	Bilangan bulat tak bertanda
`*PTR`	—	Penunjuk, untuk API tingkat rendah

10A.3 Operasi string yang paling sering dipakai

```
PGM
   DCL VAR(&TEKS)   TYPE(*CHAR) LEN(50) VALUE('PRDIBMI-EODPOST-001')
   DCL VAR(&BAGIAN) TYPE(*CHAR) LEN(10)
   DCL VAR(&GABUNG) TYPE(*CHAR) LEN(60)
   DCL VAR(&POSISI) TYPE(*DEC) LEN(3 0)
   DCL VAR(&PANJANG) TYPE(*DEC) LEN(3 0)

   /* Substring: %SST(variabel posisi panjang) */
   CHGVAR VAR(&BAGIAN) VALUE(%SST(&TEKS 9 7))       /* EODPOST */

   /* Penggabungan */
   /* *CAT  = sambung apa adanya, termasuk spasi */
   /* *TCAT = potong spasi kiri sebelum menyambung */
   /* *BCAT = potong spasi lalu sisipkan satu spasi */
   CHGVAR VAR(&GABUNG) VALUE('Job: ' *CAT &BAGIAN)
   CHGVAR VAR(&GABUNG) VALUE(&GABUNG *BCAT 'selesai')

   /* Mencari posisi karakter */
   CHGVAR VAR(&POSISI) VALUE(%SCAN('-' &TEKS))      /* 8 */

   /* Panjang efektif tanpa spasi kanan */
   CHGVAR VAR(&PANJANG) VALUE(%LEN(%TRIM(&TEKS)))

   /* Konversi huruf */
   CHGVAR VAR(&BAGIAN) VALUE(%UPPER(&BAGIAN))
   CHGVAR VAR(&BAGIAN) VALUE(%LOWER(&BAGIAN))
ENDPGM
```

Perbedaan `*CAT`, `*TCAT`, dan `*BCAT` adalah sumber bug paling sering bagi pemula. Ingat dengan kalimat ini: T memotong, B memotong lalu memberi spasi.

10A.4 Penanganan pesan: inti dari CL yang andal

Inilah yang membedakan program CL amatir dari program CL produksi.

```
PGM
   DCL VAR(&MSGID)  TYPE(*CHAR) LEN(7)
   DCL VAR(&MSGDTA) TYPE(*CHAR) LEN(512)
```

```
   DCL VAR(&MSGF)   TYPE(*CHAR) LEN(10)
   DCL VAR(&MSGFLIB) TYPE(*CHAR) LEN(10)
   DCL VAR(&MSGTXT) TYPE(*CHAR) LEN(512)

   /* Perintah yang mungkin gagal */
   DLTF FILE(QTEMP/KERJA)
   MONMSG MSGID(CPF2105) EXEC(DO)
      /* File memang belum ada, ini normal */
   ENDDO

   /* Operasi utama dengan penangkapan lengkap */
   CALL PGM(APPPGM/PROSES)
   MONMSG MSGID(CPF0000 MCH0000 RNX0000) EXEC(DO)

      /* Ambil detail exception yang barusan terjadi */
      RCVMSG MSGTYPE(*EXCP) RMV(*NO) +
             MSGID(&MSGID) MSGDTA(&MSGDTA) +
             MSGF(&MSGF) MSGFLIB(&MSGFLIB) +
             MSG(&MSGTXT)

      /* Catat ke tabel log, bukan hanya ke job log */
      RUNSQL SQL('INSERT INTO APPWRK.ERRLOG +
                  VALUES(CURRENT TIMESTAMP, ''' *CAT &MSGID *TCAT ''', +
                  ''' *CAT %SST(&MSGTXT 1 200) *TCAT ''')') +
             COMMIT(*NONE)

      /* Teruskan sebagai escape agar pemanggil tahu */
      SNDPGMMSG MSGID(&MSGID) MSGF(&MSGFLIB/&MSGF) +
                MSGDTA(&MSGDTA) MSGTYPE(*ESCAPE)
   ENDDO
ENDPGM
```

10A.5 Empat jenis pesan dan kapan memakainya

Tipe	Perilaku	Kapan dipakai
`*INFO`	Informasi, program lanjut	Progres normal, tidak butuh tindakan
`*COMP`	Penyelesaian, program lanjut	Tanda tugas selesai
`*DIAG`	Diagnostik, program lanjut	Masalah yang tidak fatal
`*ESCAPE`	Menghentikan program pemanggil	Kegagalan yang harus menghentikan proses
`*STATUS`	Ditampilkan di baris status	Progres job batch panjang
`*INQ`	Menunggu jawaban manusia	Keputusan yang butuh operator

Aturan praktis: **program batch produksi tidak boleh mengirim `*INQ` kecuali memang menuntut keputusan manusia.** Satu `*INQ` yang tidak sengaja akan menggantung EOD sampai pagi.

```
/* Progres job panjang, terlihat di WRKACTJOB kolom Function */
SNDPGMMSG MSG('Memproses record ' *CAT &HITUNG) +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)

/* Kegagalan yang harus menghentikan rantai */
SNDPGMMSG MSG('Validasi gagal, EOD dihentikan') +
          MSGTYPE(*ESCAPE)

/* Pemberitahuan tanpa menghentikan */
SNDPGMMSG MSG('Ada 12 record dilewati') +
          TOPGMQ(*PRV) MSGTYPE(*DIAG)
```

10A.6 Membaca file dalam CL

CL hanya bisa membaca berurutan, satu record pada satu waktu, dan hanya satu format record. Untuk pemrosesan serius pakai RPG atau SQL, tetapi untuk utilitas operasional ini memadai.

```
PGM
   DCLF FILE(QTEMP/OBJLIST)
   DCL VAR(&JUMLAH) TYPE(*DEC) LEN(7 0) VALUE(0)

   /* Bangun daftar objek besar */
   DSPOBJD OBJ(APPLIB/*ALL) OBJTYPE(*ALL) DETAIL(*FULL) +
           OUTPUT(*OUTFILE) OUTFILE(QTEMP/OBJLIST)

 BACA:
   RCVF
   MONMSG MSGID(CPF0864) EXEC(GOTO CMDLBL(SELESAI))

   IF COND(&ODOBSZ *GT 100000000) THEN(DO)
      SNDMSG MSG('Objek besar: ' *CAT &ODOBNM *BCAT +
                 %CHAR(&ODOBSZ)) TOUSR(SYSENG)
      CHGVAR VAR(&JUMLAH) VALUE(&JUMLAH + 1)
   ENDDO

   GOTO CMDLBL(BACA)

 SELESAI:
   SNDMSG MSG('Ditemukan ' *CAT %CHAR(&JUMLAH) *BCAT +
              'objek di atas 100MB') TOUSR(SYSENG)
ENDPGM
```

`CPF0864` adalah pesan akhir file. Menangkapnya dengan `MONMSG` adalah idiom standar CL untuk mengakhiri perulangan baca.

10A.7 Menjalankan SQL dari CL

```
/* Satu statement, tanpa hasil */
RUNSQL SQL('DELETE FROM APPWRK.STAGING WHERE TGL < CURRENT DATE - 7 DAYS') +
       COMMIT(*NONE)

/* Skrip berisi banyak statement dari file IFS */
RUNSQLSTM SRCSTMF('/scripts/pembersihan.sql') +
          COMMIT(*NONE) NAMING(*SQL) +
          ERRLVL(20)

/* Mengambil satu nilai ke variabel CL lewat outfile */
RUNSQL SQL('CREATE TABLE QTEMP/HASIL AS +
            (SELECT COUNT(*) AS JML FROM DTALIB.REKENING) +
            WITH DATA') COMMIT(*NONE)
```

`RUNSQL` tidak bisa mengembalikan hasil langsung ke variabel CL. Pola yang lazim: tulis hasilnya ke tabel di QTEMP, lalu baca dengan `DCLF` dan `RCVF`.

10A.8 Mengambil informasi sistem

```
PGM
   DCL VAR(&SISTEM)  TYPE(*CHAR) LEN(8)
   DCL VAR(&JOB)     TYPE(*CHAR) LEN(10)
   DCL VAR(&USER)    TYPE(*CHAR) LEN(10)
   DCL VAR(&NOMOR)   TYPE(*CHAR) LEN(6)
   DCL VAR(&TIPEJOB) TYPE(*CHAR) LEN(1)
   DCL VAR(&TANGGAL) TYPE(*CHAR) LEN(6)
   DCL VAR(&WAKTU)   TYPE(*CHAR) LEN(6)
   DCL VAR(&LIBL)    TYPE(*CHAR) LEN(2750)

   RTVNETA SYSNAME(&SISTEM)
   RTVJOBA JOB(&JOB) USER(&USER) NBR(&NOMOR) TYPE(&TIPEJOB)
   RTVSYSVAL SYSVAL(QDATE) RTNVAR(&TANGGAL)
   RTVSYSVAL SYSVAL(QTIME) RTNVAR(&WAKTU)

   /* Tipe job: I=interaktif, B=batch, A=autostart, M=subsystem monitor */
   IF COND(&TIPEJOB *EQ 'I') THEN(DO)
      SNDPGMMSG MSG('Program ini hanya boleh dijalankan sebagai batch') +
                MSGTYPE(*ESCAPE)
   ENDDO
ENDPGM
```

10A.9 Pola pengaman yang wajib ada di program operasional

Pola 1: Cegah eksekusi ganda

```
PGM
   DCL VAR(&LOCK) TYPE(*CHAR) LEN(10) VALUE('EODLOCK')

   /* Alokasikan data area sebagai kunci eksklusif */
   ALCOBJ OBJ((APPWRK/&LOCK *DTAARA *EXCL)) WAIT(0)
   MONMSG MSGID(CPF1002) EXEC(DO)
      SNDPGMMSG MSG('Proses sudah berjalan di job lain. Dibatalkan.') +
                MSGTYPE(*ESCAPE)
   ENDDO

   /* Kerja utama di sini */
   CALL PGM(APPPGM/PROSESEOD)

   DLCOBJ OBJ((APPWRK/&LOCK *DTAARA *EXCL))
ENDPGM
```

Pola 2: Verifikasi prasyarat sebelum bekerja

```
PGM
   DCL VAR(&STATUS) TYPE(*CHAR) LEN(1)

   /* Apakah checkpoint backup sudah selesai? */
   RTVDTAARA DTAARA(APPWRK/BCKSTATUS) RTNVAR(&STATUS)

   IF COND(&STATUS *NE 'S') THEN(DO)
      SNDPGMMSG MSG('Backup pre-EOD belum selesai. EOD tidak dijalankan.') +
                MSGTYPE(*ESCAPE)
   ENDDO

   /* Apakah masih ada sesi interaktif? */
   RUNSQL SQL('CREATE TABLE QTEMP/SESI AS +
               (SELECT COUNT(*) AS JML FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) +
                 WHERE JOB_TYPE = ''INT'') WITH DATA') COMMIT(*NONE)
ENDPGM
```

Pola 3: Timeout untuk menunggu

```
PGM
   DCL VAR(&COBA) TYPE(*DEC) LEN(3 0) VALUE(0)
   DCL VAR(&MAKS) TYPE(*DEC) LEN(3 0) VALUE(60)

 TUNGGU:
   CHGVAR VAR(&COBA) VALUE(&COBA + 1)

   CHKOBJ OBJ(APPWRK/SIAP) OBJTYPE(*DTAARA)
   MONMSG MSGID(CPF9801) EXEC(DO)
      IF COND(&COBA *GE &MAKS) THEN(DO)
         SNDPGMMSG MSG('Timeout menunggu sinyal siap setelah 60 menit') +
                   MSGTYPE(*ESCAPE)
      ENDDO
      DLYJOB DLY(60)
      GOTO CMDLBL(TUNGGU)
   ENDDO

   /* Sinyal ditemukan, lanjutkan */
ENDPGM
```

Perulangan tunggu tanpa timeout adalah salah satu penyebab paling umum EOD yang menggantung semalaman tanpa ada yang tahu.

10A.10 Data area dan data queue

Dua objek kecil ini adalah tulang punggung koordinasi antar job.

```
/* Data area: menyimpan satu nilai yang dibaca banyak job */
CRTDTAARA DTAARA(APPWRK/TGLBUKU) TYPE(*CHAR) LEN(10) +
          VALUE('2026-09-18') TEXT('Tanggal buku berjalan')

CHGDTAARA DTAARA(APPWRK/TGLBUKU) VALUE('2026-09-19')

RTVDTAARA DTAARA(APPWRK/TGLBUKU) RTNVAR(&TGL)

/* Data area khusus: LDA, lokal per job */
CHGDTAARA DTAARA(*LDA (1 10)) VALUE('PARAM1')
RTVDTAARA DTAARA(*LDA (1 10)) RTNVAR(&NILAI)
```

```
/* Data queue: antrean pesan antar job, cepat dan asinkron */
CRTDTAQ DTAQ(APPWRK/EODQ) MAXLEN(256) SEQ(*FIFO) +
        TEXT('Antrean sinyal EOD')

/* Mengirim, lewat API */
CALL PGM(QSNDDTAQ) PARM('EODQ' 'APPWRK' &PANJANG &DATA)

/* Menerima dengan timeout 300 detik */
CALL PGM(QRCVDTAQ) PARM('EODQ' 'APPWRK' &PANJANG &DATA &TUNGGU)
```

Data queue lebih baik daripada perulangan `DLYJOB` untuk menunggu sinyal: job penerima tidur tanpa memakan CPU, dan bangun seketika saat pesan datang.

10A.11 Kesalahan CL yang sering ditemukan dalam audit kode

Kesalahan	Akibat	Perbaikan
`MONMSG` global tanpa aksi	Semua error ditelan diam-diam	Tangani per perintah, teruskan escape
Tidak ada `MONMSG` sama sekali	Job berhenti tanpa pesan jelas	Tambahkan penanganan eksplisit
Perulangan tunggu tanpa timeout	Job menggantung selamanya	Batasi jumlah percobaan
Nama library tidak dikualifikasi	Salah lingkungan	Tulis `LIB/OBJ` eksplisit
`SNDMSG` alih-alih `SNDPGMMSG`	Pemanggil tidak tahu ada kegagalan	Pakai `*ESCAPE`
Password atau data sensitif di parameter	Terlihat di `WRKACTJOB`	Ambil dari objek terlindungi
Tidak melepas `ALCOBJ`	Objek terkunci permanen	Selalu pasangkan `DLCOBJ`
`DLYJOB` panjang di job interaktif	Sesi tampak menggantung	Jalankan sebagai batch

10A.12 Latihan

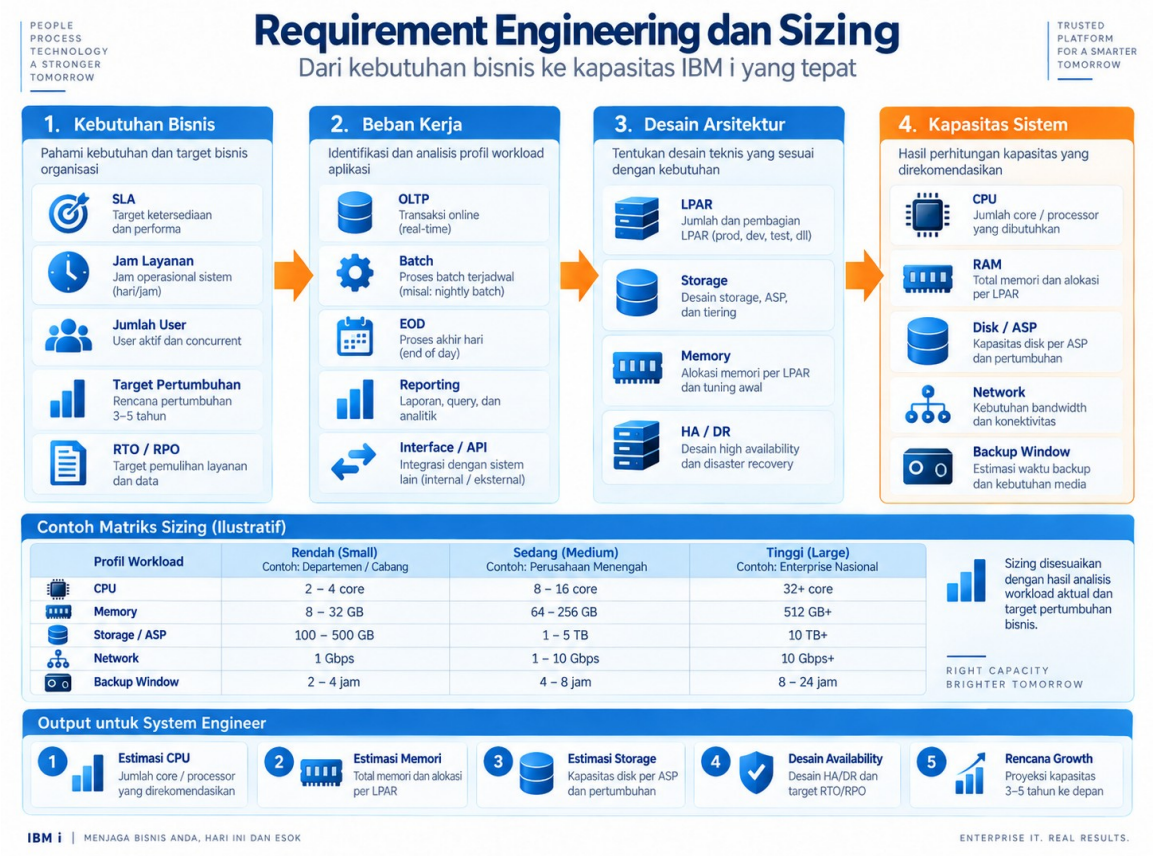
Tulis program CL yang:

1. Memeriksa apakah pemakaian ASP di atas 80%
2. Bila ya, mencari sepuluh library terbesar
3. Mengirim ringkasannya ke antrean pesan System Engineer
4. Mencatat hasilnya ke tabel riwayat
5. Mengembalikan escape message bila pemakaian di atas 90%

Kerangka jawabannya memakai `RTVSYSVAL`, `RUNSQL` dengan `QSYS2.SYSDISKSTAT`, outfile di QTEMP, `DCLF` dengan `RCVF`, dan `SNDPGMMSG`. Bandingkan hasil Anda dengan pola-pola di 10A.9.

Bagian III — Perencanaan dan Implementasi

Bab 11. Requirement Engineering dan Sizing



Gambar 16 — Dari kebutuhan bisnis ke kapasitas sistem

Sizing yang salah adalah kesalahan paling mahal dalam proyek IBM i, karena koreksinya menuntut pembelian perangkat keras baru.

11.1 Data yang wajib dikumpulkan sebelum sizing

Kategori	Data yang dibutuhkan	Sumber
Volume transaksi	Transaksi per hari, puncak per jam	Aplikasi eksisting, proyeksi bisnis
Pengguna	Jumlah sesi interaktif serentak	Log koneksi
Batch	Durasi EOD saat ini, jumlah job	Job accounting
Data	Ukuran basis data, laju pertumbuhan tahunan	`DSPLIB`, riwayat kapasitas
Ketersediaan	RTO, RPO, jam layanan	Kesepakatan dengan bisnis
Pertumbuhan	Proyeksi 3–5 tahun	Rencana bisnis

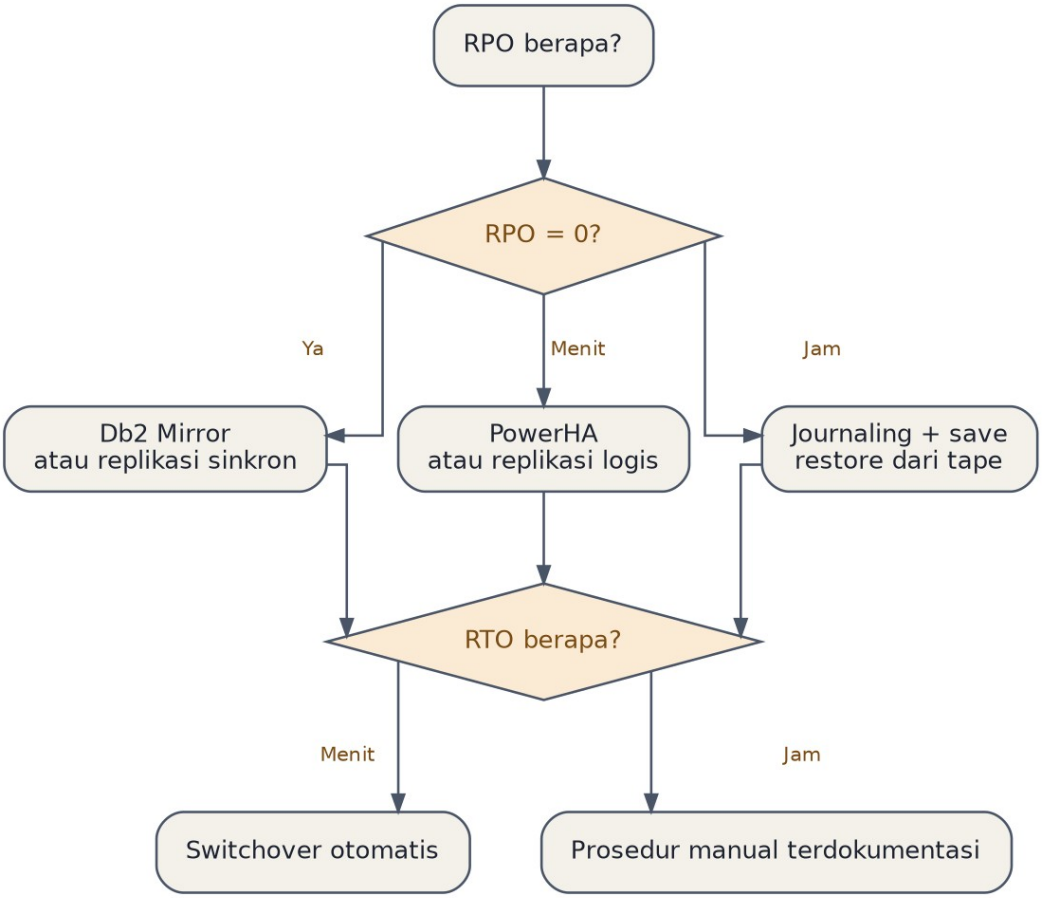
11.2 Satuan kapasitas

Satuan	Mengukur apa	Catatan
CPW	Commercial Processing Workload, beban transaksional	Metrik lama, masih dipakai untuk perbandingan relatif

Satuan	Mengukur apa	Catatan
rPerf	Kinerja relatif Power Systems	Untuk perbandingan lintas model
Processing units	Alokasi CPU logis ke partisi	1.0 = satu core penuh

Aturan praktis yang saya pakai: **sizing untuk beban puncak, bukan rata-rata, lalu tambahkan headroom 30%.** Di perbankan, puncak bukan jam sibuk siang hari, melainkan End of Month yang jatuh di akhir pekan.

11.3 Menerjemahkan RTO/RPO ke keputusan teknis

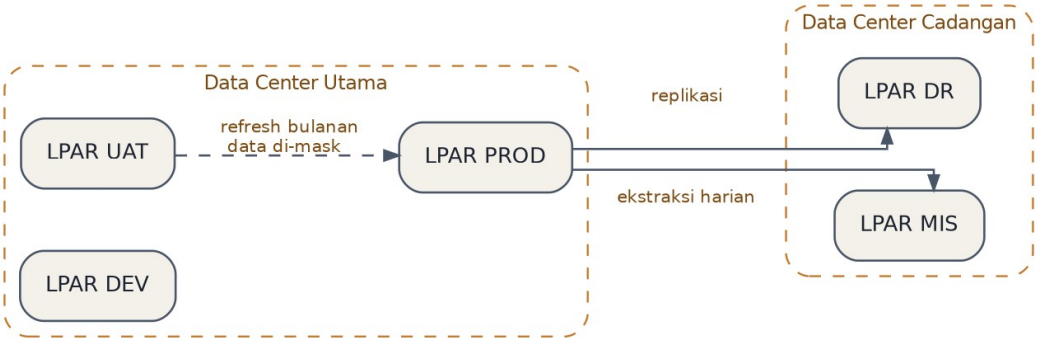


Gambar 17

Jangan menyebut angka RTO/RPO sebelum biaya masing-masing opsi dihitung. RPO nol berarti replikasi sinkron, dan replikasi sinkron berarti keterbatasan jarak fisik antar data center.

Bab 12. Desain Topologi

12.1 Pola lingkungan yang lazim di perbankan



Gambar 18

12.2 Prinsip pemisahan

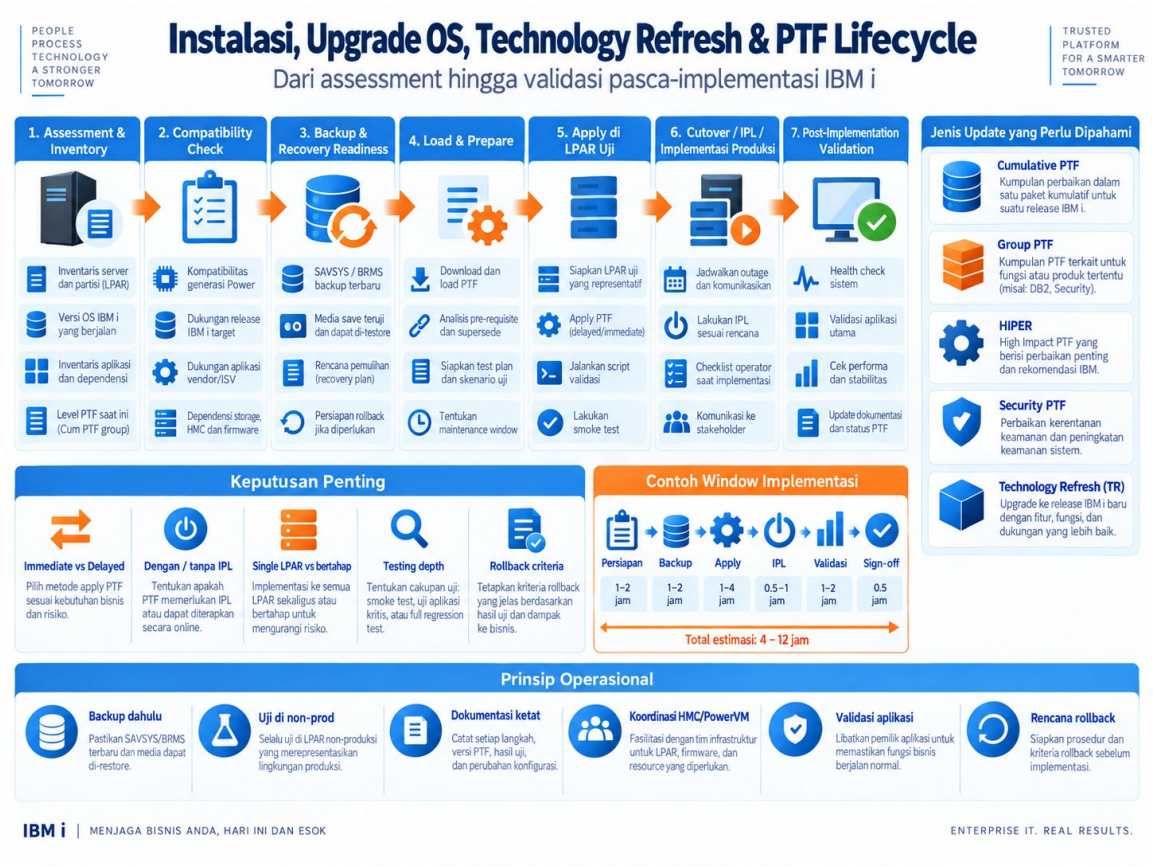
1. **Produksi tidak pernah berbagi partisi dengan non-produksi.** Bukan soal kinerja saja, tetapi isolasi kesalahan dan pemisahan wewenang.
2. **DR bukan tempat menumpang beban lain.** Partisi DR yang dipakai untuk pelaporan akan gagal saat benar-benar dibutuhkan.
3. **Data produksi yang masuk non-produksi wajib di-mask.** Ini kewajiban di bawah UU Pelindungan Data Pribadi, bukan preferensi.
4. **MIS terpisah dari produksi.** Kueri analitik ad hoc bisa mengunci tabel dan merusak SLA transaksional.

12.3 Rancangan ASP

ASP	Isi	Alasan
ASP 1 (system)	OS, library sistem	Wajib, tidak bisa dihindari
ASP 2	Journal receiver	Isolasi I/O tulis yang intensif
iASP	Data aplikasi	Prasyarat PowerHA switchover

Menempatkan journal receiver di ASP terpisah adalah salah satu perbaikan kinerja termurah yang bisa dilakukan pada sistem yang sudah berjalan.

Bab 13. Instalasi, Upgrade, dan Manajemen PTF



Gambar 19 — Siklus hidup instalasi, upgrade OS, TR, dan PTF

13.1 Jenis perbaikan

Jenis	Isi	Frekuensi penerapan
Individual PTF	Perbaikan satu masalah spesifik	Sesuai kebutuhan
Cumulative PTF Package	Kumpulan perbaikan OS	2–4 kali setahun
Group PTF	Kumpulan per area, misal Db2, HTTP, Java	Triwulanan
Technology Refresh	Fitur baru tanpa ganti rilis	1–2 kali setahun
Rilis baru	Versi OS baru, misal 7.5 ke 7.6	3–5 tahun

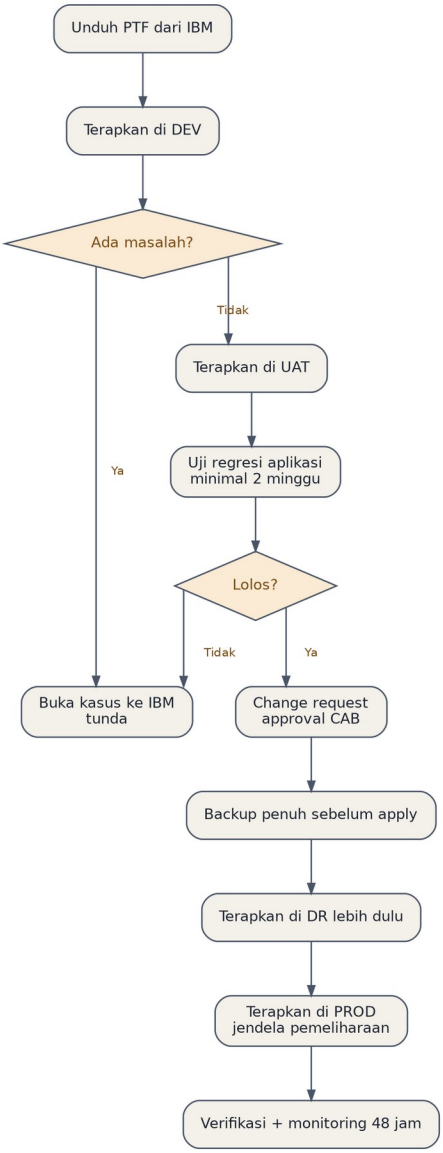
13.2 Memeriksa level saat ini

```
/* Level rilis OS */
DSPPTF LICPGM(5770SS1)

/* Daftar group PTF dan levelnya */
WRKPTFGRP

/* Cetak seluruh PTF terpasang untuk dokumentasi */
DSPPTF LICPGM(*ALL) OUTPUT(*PRINT)
```

Versi SQL yang lebih mudah dijadwalkan dan diarsipkan:

```
SELECT PTF_GROUP_NAME,
       PTF_GROUP_LEVEL,
       PTF_GROUP_TARGET_RELEASE,
       PTF_GROUP_STATUS
  FROM QSYS2.GROUP_PTF_INFO
 ORDER BY PTF_GROUP_NAME;

-- PTF yang butuh IPL untuk aktif
SELECT PTF_IDENTIFIER, PTF_PRODUCT_ID,
       PTF_LOADED_STATUS, PTF_ACTION_PENDING
  FROM QSYS2.PTF_INFO
 WHERE PTF_ACTION_PENDING = 'YES';
```

13.3 Alur penerapan PTF yang aman



Gambar 20

Menerapkan di DR lebih dulu memberi dua keuntungan: Anda menguji prosedur di lingkungan identik, dan jika terjadi masalah, produksi masih bersih untuk dijadikan tempat kembali.

13.4 Checklist upgrade rilis OS

- ☐ Verifikasi model server didukung rilis target (IBM i 7.6 menuntut Power10 ke atas)
- ☐ Periksa kompatibilitas seluruh Licensed Program Product
- ☐ Konfirmasi dukungan vendor aplikasi untuk rilis target, tertulis
- ☐ Periksa kompatibilitas perangkat lunak pihak ketiga: backup, monitoring, HA
- ☐ Jalankan analisis konversi objek program
- ☐ Pastikan kapasitas disk mencukupi, minimal 20% bebas
- ☐ Backup penuh sistem termasuk `SAVSYS`
- ☐ Verifikasi backup dapat dibaca, bukan hanya selesai tanpa error
- ☐ Siapkan rencana rollback tertulis dengan batas waktu keputusan
- ☐ Jadwalkan jendela dengan cadangan waktu dua kali lipat estimasi

Bab 14. Konfigurasi Awal

14.1 System values kritikal

System value	Nilai disarankan	Alasan
`QSECURITY`	40 atau 50	Di bawah 40 tidak dapat diterima untuk perbankan
`QMAXSIGN`	3	Batas percobaan login gagal
`QMAXSGNACN`	3	Nonaktifkan profil dan device
`QINACTITV`	15	Timeout sesi menganggur, menit
`QINACTMSGQ`	`*DSCJOB`	Putuskan sesi menganggur
`QPWDLVL`	3 atau 4	Password case-sensitive, panjang
`QPWDRQDDIF`	5	Tidak boleh mengulang 5 password terakhir
`QAUDCTL`	`*AUDLVL *OBJAUD`	Aktifkan audit
`QAUDLVL`	`*AUTFAIL *SECURITY *CREATE *DELETE *SAVRST`	Minimum untuk audit perbankan
`QALWOBJRST`	`*NONE`	Cegah restore objek dengan wewenang berbahaya
`QCRTAUT`	`*EXCLUDE`	Default wewenang objek baru
`QLMTSECOFR`	1	Batasi tempat login profil berwenang tinggi

Memeriksa dan mengubah:

```
/* Menampilkan */
DSPSYSVAL SYSVAL(QSECURITY)
WRKSYSVAL SYSVAL(*SEC)

/* Mengubah, butuh *SECADM dan sebagian butuh IPL */
CHGSYSVAL SYSVAL(QINACTITV) VALUE('15')
```

Semua system value bisa diambil sekaligus untuk baseline dokumentasi:

```
SELECT SYSTEM_VALUE_NAME, CURRENT_NUMERIC_VALUE,
       CURRENT_CHARACTER_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME IN (
     'QSECURITY','QMAXSIGN','QPWDLVL','QAUDCTL',
     'QAUDLVL','QALWOBJRST','QCRTAUT','QLMTSECOFR')
 ORDER BY SYSTEM_VALUE_NAME;
```

14.2 Konfigurasi TCP/IP dasar

```
/* Interface */
ADDTCPIFC INTNETADR('10.20.30.40') LIND(ETHLINE) +
          SUBNETMASK('255.255.255.0')

/* Rute default */
ADDTCPRTE RTEDEST(*DFTROUTE) SUBNETMASK(*NONE) +
          NEXTHOP('10.20.30.1')

/* Nama host */
CHGTCPDMN HOSTNAME(PRDIBMI) DMNNAME(bank.co.id) +
          INTNETADR('10.20.30.50')

/* Mulai server yang diperlukan saja */
STRTCPSVR SERVER(*TELNET)
STRTCPSVR SERVER(*HTTP) HTTPSVR(ADMIN)
```

Prinsip pengerasan: **matikan setiap server TCP yang tidak dipakai.** FTP, TFTP, dan Telnet tanpa SSL adalah temuan audit yang hampir pasti muncul.

```
-- Server TCP yang sedang aktif
SELECT SERVER_NAME, SERVER_STATUS, ACTIVE_JOBS
  FROM QSYS2.SERVER_SHARE_INFO;

-- Koneksi jaringan aktif dan asalnya
SELECT REMOTE_ADDRESS, LOCAL_PORT, PROTOCOL,
       CONNECTION_TYPE, JOB_NAME
  FROM QSYS2.NETSTAT_JOB_INFO
 ORDER BY REMOTE_ADDRESS;
```

14.3 Access Client Solutions

ACS menggantikan iSeries Access lama. Yang perlu dikuasai System Engineer:

Fitur ACS	Kegunaan harian
5250 Emulator	Sesi layar hijau
Run SQL Scripts	Menjalankan IBM i Services, skrip administrasi
Navigator for i	Antarmuka web untuk work management dan kinerja
Performance Data Investigator	Analisis grafis Collection Services
Integrated File System	Menjelajah IFS
Schema	Administrasi objek Db2

Bab 15. Manajemen Konfigurasi dan Baseline

Baseline adalah rekaman kondisi sistem yang disepakati sebagai acuan. Tanpa baseline, Anda tidak bisa membuktikan apa yang berubah saat terjadi insiden.

15.1 Isi minimum dokumen baseline

1. Daftar seluruh system value dan nilainya
2. Konfigurasi subsystem, job queue, class, memory pool
3. Daftar library aplikasi dan pemiliknya
4. Konfigurasi jaringan dan server TCP yang aktif
5. Daftar user profile dengan special authority
6. Konfigurasi journaling: file apa dijurnal ke journal mana
7. Jadwal backup dan isi setiap save
8. Level OS, PTF group, dan Licensed Program Product
9. Konfigurasi HA/DR dan parameter replikasi

15.2 Skrip pengambilan baseline otomatis

```
PGM
   DCL VAR(&TGL) TYPE(*CHAR) LEN(8)
   RTVSYSVAL SYSVAL(QDATE) RTNVAR(&TGL)

   /* System values */
   WRKSYSVAL SYSVAL(*ALL) OUTPUT(*PRINT)

   /* Konfigurasi subsystem */
   DSPSBSD SBSD(QGPL/EODSBS) OUTPUT(*PRINT)

   /* User profile dengan wewenang khusus */
   DSPUSRPRF USRPRF(*ALL) TYPE(*BASIC) OUTPUT(*OUTFILE) +
             OUTFILE(BASELINE/USRPRF)

   /* Objek aplikasi */
   DSPOBJD OBJ(APPLIB/*ALL) OBJTYPE(*ALL) DETAIL(*FULL) +
           OUTPUT(*OUTFILE) OUTFILE(BASELINE/OBJLIST)

   /* Wewenang objek */
   DSPOBJAUT OBJ(APPLIB/*ALL) OBJTYPE(*ALL) +
             OUTPUT(*OUTFILE) OUTFILE(BASELINE/OBJAUT)

   SNDMSG MSG('Baseline selesai ' *CAT &TGL) TOUSR(SYSENG)
ENDPGM
```

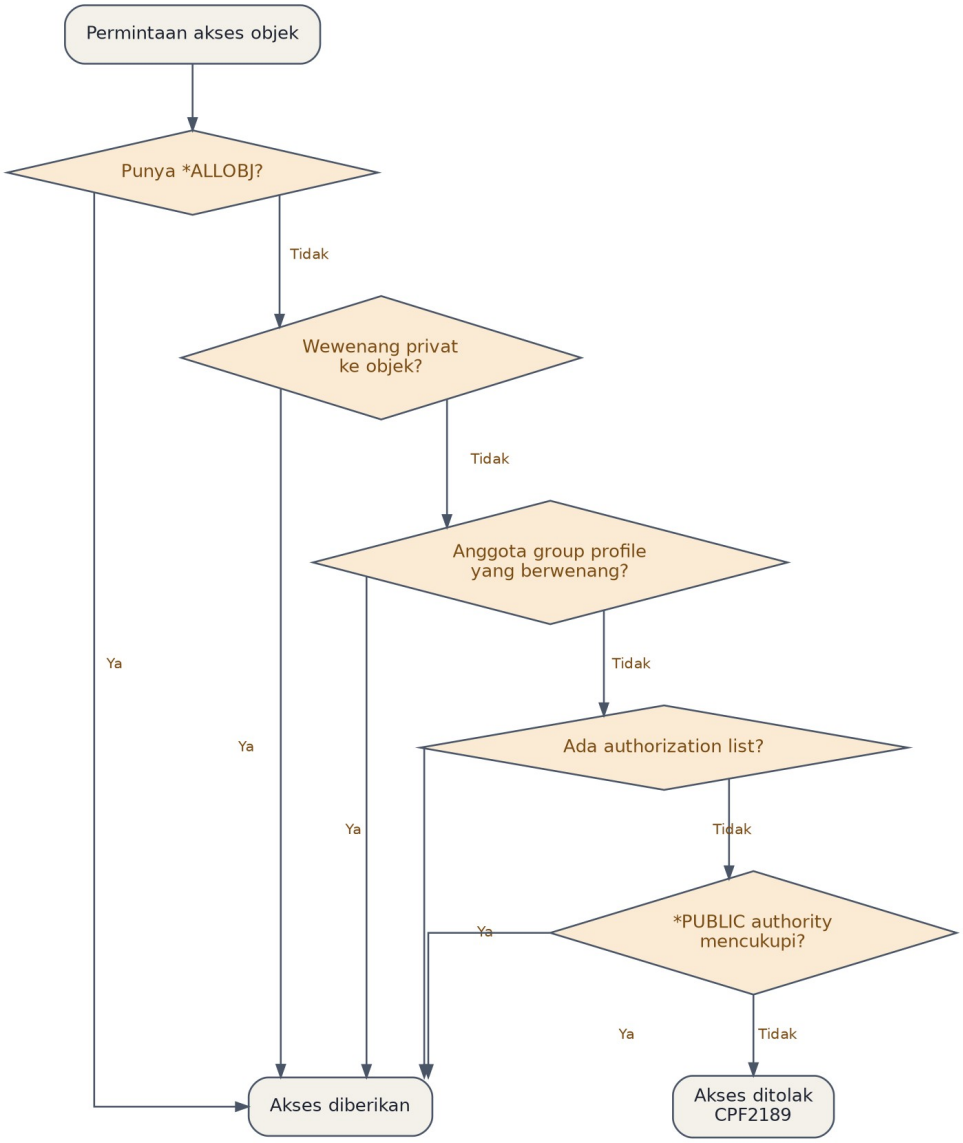
Jadwalkan bulanan, simpan keluarannya, dan bandingkan antar periode. Perbedaan yang tidak bisa dijelaskan oleh change request yang disetujui adalah temuan yang harus ditindaklanjuti.

⚙️ Bagian IV — Operasi Harian

Bab 16. Administrasi Pengguna dan Wewenang

Gambar 21 — Administrasi pengguna dan wewenang

16.1 Model wewenang IBM i



Gambar 22

Urutan pemeriksaan ini penting dipahami, karena menjelaskan mengapa mencabut wewenang privat tidak selalu memblokir akses — `*PUBLIC` masih bisa memberikannya.

16.2 Special authority dan risikonya

Special authority	Kemampuan	Risiko
`*ALLOBJ`	Akses seluruh objek	Tertinggi, lewati semua kontrol
`*SECADM`	Kelola user profile	Bisa membuat profil berwenang
`*SAVSYS`	Save/restore tanpa wewenang objek	Bisa menyalin seluruh data
`*JOBCTL`	Kendalikan job, ubah prioritas	Bisa hentikan subsystem
`*SPLCTL`	Akses seluruh spooled file	Bisa baca laporan rahasia
`*SERVICE`	Alat servis sistem	Bisa ubah memori langsung
`*AUDIT`	Ubah konfigurasi audit	Bisa matikan jejak audit
`*IOSYSCFG`	Konfigurasi jaringan	Bisa buka jalur keluar data

Audit rutin yang wajib dijalankan minimal bulanan:

```
SELECT AUTHORIZATION_NAME,
       STATUS,
       SPECIAL_AUTHORITIES,
       USER_CLASS_NAME,
       PREVIOUS_SIGNON,
       DAYS(CURRENT DATE) - DAYS(DATE(PREVIOUS_SIGNON)) AS hari_tidak_dipakai
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES <> ''
   AND USER_CLASS_NAME <> '*USER'
 ORDER BY hari_tidak_dipakai DESC;
```

Profil dengan `*ALLOBJ` yang tidak dipakai lebih dari 90 hari harus dinonaktifkan. Di lingkungan perbankan, jumlah profil `*ALLOBJ` idealnya di bawah lima, seluruhnya berupa akun terkendali dengan password vaulting.

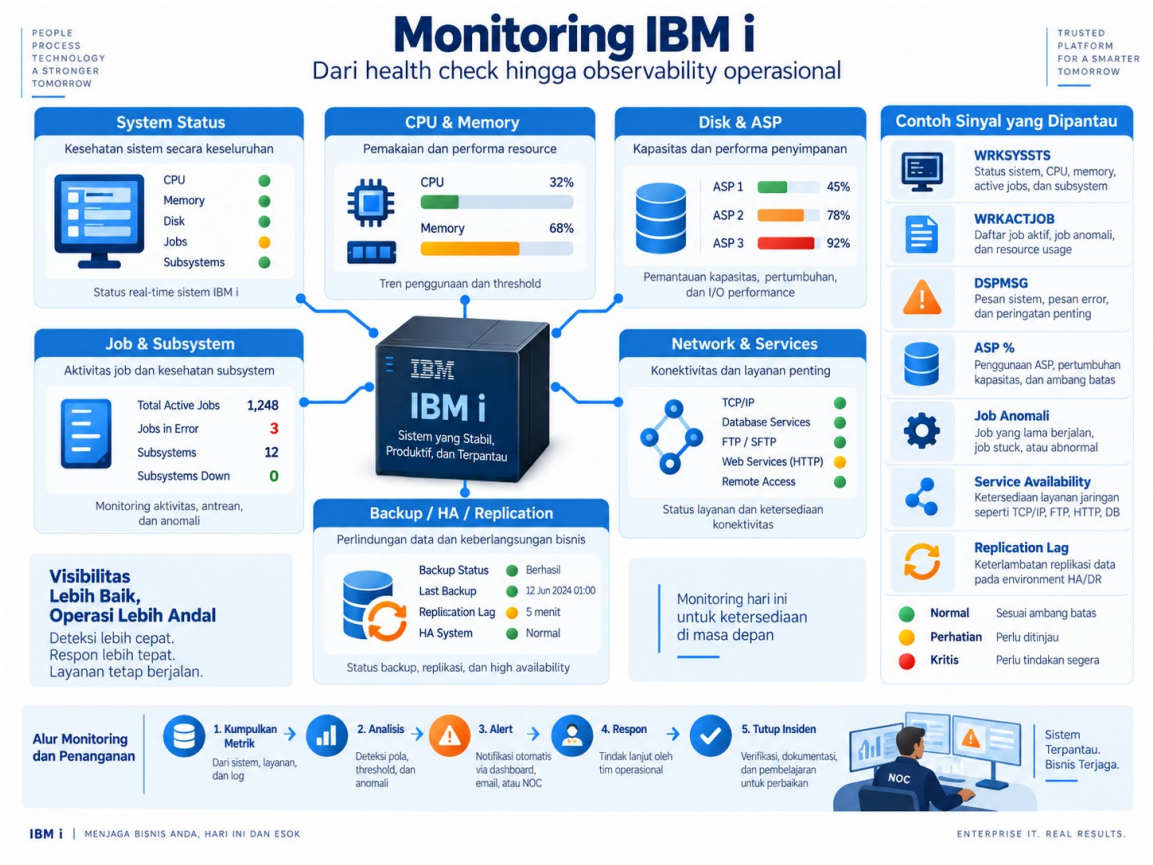
16.3 Membuat profil dengan pola yang benar

```
/* Group profile untuk peran, bukan untuk orang */
CRTUSRPRF USRPRF(GRPTELLER) PASSWORD(*NONE) +
          STATUS(*DISABLED) TEXT('Group untuk teller')

/* Pengguna individual, mewarisi dari group */
CRTUSRPRF USRPRF(BUDI001) PASSWORD(*NONE) +
          USRCLS(*USER) +
          GRPPRF(GRPTELLER) OWNER(*GRPPRF) +
          INLMNU(*SIGNOFF) INLPGM(APPLIB/MENUTELLER) +
          LMTCPB(*YES) +
          SPCAUT(*NONE) +
          TEXT('Budi Santoso - Teller Cabang Sudirman')
```

Empat parameter yang sering dilewatkan dan berakibat temuan audit:

Parameter	Nilai aman	Akibat jika salah
`LMTCPB`	`*YES`	Pengguna bisa mengetik perintah CL sembarang
`INLMNU`	`*SIGNOFF`	Pengguna keluar ke menu sistem
`SPCAUT`	`*NONE`	Wewenang berlebih
`OWNER`	`*GRPPRF`	Objek dimiliki perorangan, sulit dialihkan

📊 Bab 17. Monitoring



Gambar 23 — Pemantauan IBM i: dari health check ke observability

17.1 Yang harus dipantau dan ambangnya

Metrik	Ambang perhatian	Ambang kritis	Cara ambil
CPU utilization	70%	85%	`WRKSYSSTS`
Disk utilization ASP	75%	85%	`WRKDSKSTS`
Disk busy	30%	40%	`WRKDSKSTS`
Faulting rate pool	Bergantung pool	Naik dua kali baseline	`WRKSYSSTS`
Job dalam status MSGW	1	3	`WRKACTJOB`
Job queue tertahan	1	1	`WRKJOBQ`
Temporary storage	60%	80%	`WRKSYSSTS`
Journal receiver	60% ASP	75% ASP	`WRKJRNA`

17.2 Memantau pesan yang butuh jawaban

Job yang berstatus MSGW menunggu jawaban manusia. Di lingkungan batch semalam, satu job MSGW yang tidak terdeteksi bisa menahan seluruh rantai EOD sampai pagi.

```
SELECT JOB_NAME,
       JOB_STATUS,
       SUBSYSTEM,
       JOB_ENTERED_SYSTEM_TIME,
       FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         JOB_STATUS_FILTER => '*MSGW'))
 ORDER BY JOB_ENTERED_SYSTEM_TIME;
```

Membaca pesan yang menunggu:

```
SELECT MESSAGE_ID,
       MESSAGE_TEXT,
       MESSAGE_SECOND_LEVEL_TEXT,
       FROM_JOB,
       MESSAGE_TIMESTAMP
  FROM QSYS2.MESSAGE_QUEUE_INFO
 WHERE MESSAGE_QUEUE_NAME = 'QSYSOPR'
   AND MESSAGE_TYPE = 'INQUIRY'
 ORDER BY MESSAGE_TIMESTAMP DESC;
```

17.3 Watch — deteksi otomatis tanpa perangkat tambahan

Mekanisme STRWCH memantau munculnya pesan tertentu dan memicu program Anda sendiri.

```
/* Pantau pesan disk penuh dan job abnormal */
STRWCH SSNID(WATCHOPS) +
       WCHPGM(APPLIB/WCHHANDLER) +
       WCHMSG((CPF0907) (CPI0948) (CPF1240) (CPA5305)) +
       WCHMSGQ((QSYS/QSYSOPR *ALL))

/* Melihat watch yang aktif */
WRKWCH WCH(*MSGW)

/* Menghentikan */
ENDWCH SSNID(WATCHOPS)
```

Program penangannya:

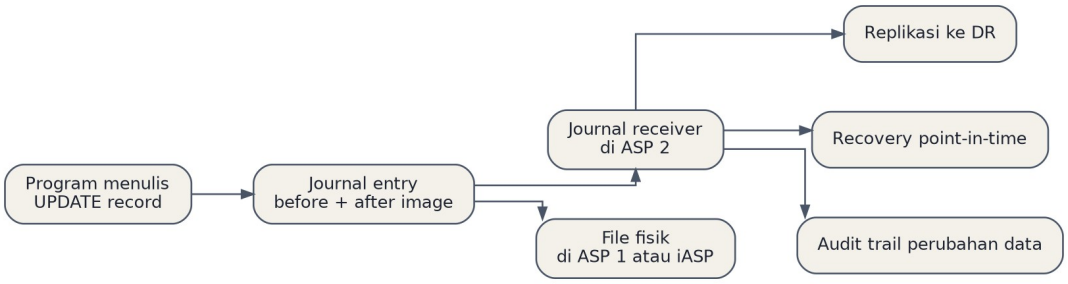
```
PGM PARM(&WCHOPT &SSNID &EVT &WCHMSG)
   DCL VAR(&WCHOPT) TYPE(*CHAR) LEN(10)
   DCL VAR(&SSNID)  TYPE(*CHAR) LEN(10)
   DCL VAR(&EVT)    TYPE(*CHAR) LEN(10)
   DCL VAR(&WCHMSG) TYPE(*CHAR) LEN(2000)
   DCL VAR(&MSGID)  TYPE(*CHAR) LEN(7)

   CHGVAR VAR(&MSGID) VALUE(%SST(&WCHMSG 1 7))

   /* Kirim ke sistem monitoring lewat pesan dan email */
   SNDMSG MSG('ALERT IBM i: ' *CAT &MSGID) TOUSR(SYSENG)

   /* Panggil program integrasi ke sistem tiket */
   CALL PGM(APPLIB/SNDALERT) PARM(&MSGID)
ENDPGM
```

Bab 18. Manajemen Penyimpanan dan Journaling

18.1 Mengapa journaling wajib

Journaling mencatat setiap perubahan record sebelum dan sesudah. Tanpa journaling, tiga hal mustahil

dilakukan: commitment control, replikasi, dan pemulihan ke titik waktu tertentu.



Gambar 24

18.2 Menyiapkan journaling

```
/* 1. Buat receiver */
CRTJRNRCV JRNRCV(JRNLIB/RCVDTA0001) +
          ASP(2) THRESHOLD(500000) +
          TEXT('Receiver data aplikasi')

/* 2. Buat journal */
CRTJRN JRN(JRNLIB/JRNDTA) +
       JRNRCV(JRNLIB/RCVDTA0001) +
       MNGRCV(*SYSTEM) DLTRCV(*NO) +
       RCVSIZOPT(*MAXOPT3) +
       TEXT('Journal data aplikasi')

/* 3. Mulai journaling seluruh file di library */
STRJRNPF FILE(DTALIB/*ALL) JRN(JRNLIB/JRNDTA) +
         IMAGES(*BOTH) OMTJRNE(*OPNCLO)

/* 4. Verifikasi */
WRKJRNA JRN(JRNLIB/JRNDTA)
```

Catatan penting: `DLTRCV(*NO)` berarti receiver lama tidak dihapus otomatis. Ini disengaja — receiver adalah bahan pemulihan, dan penghapusan harus terjadi hanya setelah receiver tersimpan di backup. Konsekuensinya Anda wajib punya proses pembersihan terjadwal, kalau tidak ASP akan penuh.

18.3 Membaca journal untuk investigasi

Ini salah satu kemampuan paling berharga saat menyelidiki "siapa yang mengubah saldo rekening ini".

```
DSPJRN JRN(JRNLIB/JRNDTA) +
       FILE((DTALIB/REKENING)) +
       RCVRNG(*CURCHAIN) +
       FROMTIME('18/09/2026' '00:00:00') +
       TOTIME('18/09/2026' '23:59:59') +
       ENTTYP(*RCD) +
       OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4) +
       OUTFILE(QTEMP/JRNOUT)
```

Lalu analisis:

```
SELECT JOENTT  AS tipe_entri,
       JOJOB   AS nama_job,
       JOUSER  AS pengguna,
       JOPGM   AS program,
       JOTSTP  AS waktu,
       JOESD   AS isi_record
  FROM QTEMP.JRNOUT
 WHERE JOENTT IN ('UP','DL','PT')
 ORDER BY JOTSTP;
```

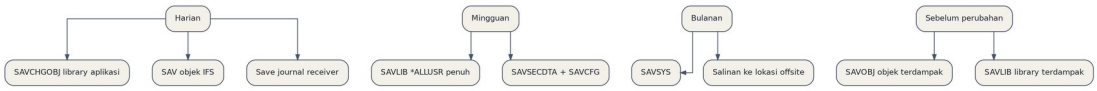
Kode entri yang paling sering dicari: `PT` penulisan record baru, `UP` pembaruan, `DL` penghapusan, `UB` before-image pembaruan.

💾 Bab 19. Backup dan Recovery

19.1 Tingkatan penyelamatan

Perintah	Menyimpan apa	Kapan dipakai
`SAVSYS`	Sistem operasi, system value, user profile	Sebelum upgrade, bulanan
`SAVLIB *NONSYS`	Seluruh library termasuk QSYS2	Mingguan, butuh sistem restricted
`SAVLIB *ALLUSR`	Seluruh library pengguna	Mingguan
`SAVOBJ`	Objek tertentu	Sebelum perubahan
`SAVCHGOBJ`	Objek yang berubah saja	Harian
`SAV`	Objek IFS	Harian
`SAVSECDTA`	Data keamanan	Setiap perubahan wewenang
`SAVCFG`	Konfigurasi perangkat	Setelah perubahan konfigurasi

19.2 Strategi backup berlapis



Gambar 25

19.3 Save While Active

Ini kunci agar backup tidak memakan jendela batch. Sistem mengambil titik konsisten (checkpoint), lalu menyalin dari titik itu sementara aplikasi tetap berjalan.

```
SAVLIB LIB(DTALIB) DEV(TAPMLB01) +
       SAVACT(*SYNCLIB) +
       SAVACTMSGQ(QGPL/SAVMSGQ) +
       SAVACTWAIT(120) +
       ACCPTH(*YES)
```

Yang perlu dipahami:

1. `SAVACT(*SYNCLIB)` memberi titik konsistensi tunggal untuk seluruh library — yang Anda butuhkan agar data relasional tetap koheren.
2. Pesan `CPI3712` dikirim ke `SAVACTMSGQ` saat checkpoint tercapai. Setelah pesan itu, aplikasi bisa dilanjutkan.

3. `SAVACTWAIT(120)` adalah batas tunggu untuk mendapatkan lock. Jika ada job yang menahan record terlalu lama, save gagal.
4. Save While Active **tidak menggantikan** kebutuhan restore testing.

19.4 Uji pemulihan

Backup yang tidak pernah diuji restore bukan backup, melainkan asumsi. Jadwal minimum yang saya sarankan:

Uji	Frekuensi	Bukti yang disimpan
Restore satu objek acak	Bulanan	Log restore, verifikasi isi
Restore satu library penuh ke UAT	Triwulanan	Berita acara, durasi aktual
Restore sistem penuh ke DR	Tahunan	Berita acara, RTO aktual vs target
Baca ulang media offsite	Semesteran	Laporan verifikasi media

Bab 20. Penjadwalan dan Output Management

20.1 Job scheduler bawaan

```
/* Jadwalkan pekerjaan harian */
ADDJOBSCDE JOB(BACKUPHRN) +
           CMD(CALL PGM(APPLIB/BCKDAILY)) +
           FRQ(*WEEKLY) SCDDAY(*ALL) +
           SCDTIME('220000') +
           JOBQ(QGPL/EODJOBQ) +
           TEXT('Backup harian objek berubah')

/* Melihat jadwal */
WRKJOBSCDE

/* Riwayat eksekusi */
DSPLOG PERIOD((*AVAIL *CURRENT)) MSGID(CPC1221)
```

Untuk lingkungan core banking dengan rantai ketergantungan kompleks, penjadwal bawaan sering tidak memadai. Ini dibahas di Bab 31.

20.2 Spooled file dan retensi

```
-- Spooled file terbesar yang menumpuk
SELECT USER_NAME, OUTPUT_QUEUE_NAME,
       SPOOLED_FILE_NAME, TOTAL_PAGES,
       SIZE / 1024 / 1024 AS ukuran_mb,
       CREATE_TIMESTAMP
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
 WHERE CREATE_TIMESTAMP < CURRENT DATE - 90 DAYS
 ORDER BY SIZE DESC
 FETCH FIRST 50 ROWS ONLY;
```

Spooled file lama adalah penyebab diam-diam pembengkakan penyimpanan. Tetapkan kebijakan retensi tertulis, dan ingat bahwa laporan perbankan tertentu punya kewajiban retensi regulatoris yang lebih panjang dari kebijakan TI Anda.

📕 Bab 21. Runbook dan Integrasi ITIL

Gambar 26 — Runbook dan integrasinya dengan praktik ITIL

21.1 Anatomi runbook yang berguna

Runbook yang baik bisa dijalankan pukul tiga pagi oleh orang yang belum pernah menangani masalah tersebut.

Bagian	Isi
Judul dan pemicu	Gejala persis, message ID, tampilan layar
Dampak	Layanan apa yang terganggu, siapa terdampak
Prasyarat	Wewenang yang dibutuhkan, alat yang harus terbuka
Langkah diagnosis	Perintah persis, keluaran yang diharapkan
Langkah perbaikan	Perintah persis, dengan peringatan risiko
Verifikasi	Cara memastikan masalah benar-benar selesai
Eskalasi	Nama, nomor, batas waktu sebelum eskalasi
Riwayat revisi	Siapa, kapan, apa yang berubah

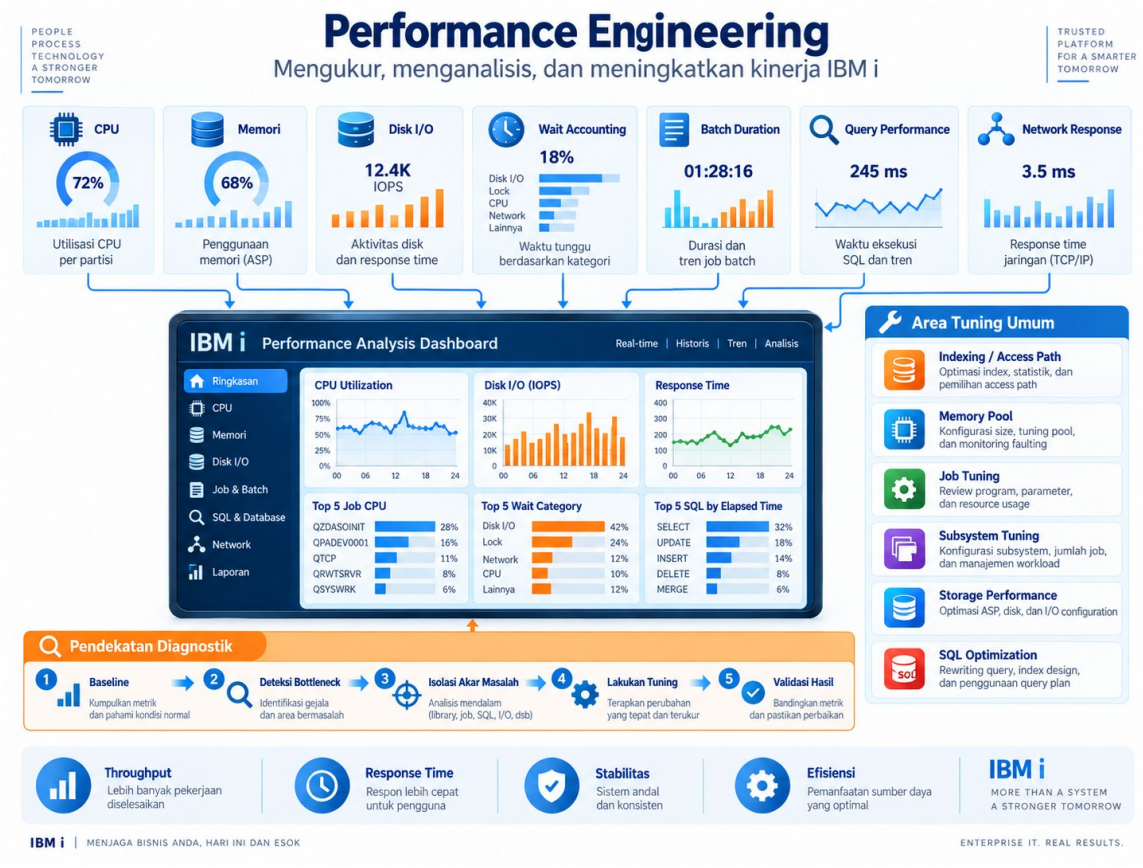
21.2 Pemetaan ke praktik ITIL 4

Praktik ITIL	Implementasi di IBM i
Incident management	Watch otomatis, runbook, eskalasi
Problem management	Analisis job log dan journal, RCA tertulis
Change management	Object promotion terkontrol, CAB

Praktik ITIL	Implementasi di IBM i
Service configuration	Baseline bulanan, CMDB
Monitoring and event	IBM i Services terjadwal, integrasi ke SIEM
Service continuity	Uji DR tahunan, PowerHA
Capacity and performance	Collection Services, tren bulanan

🔒 Bagian V — Kinerja, Keandalan, dan Keamanan

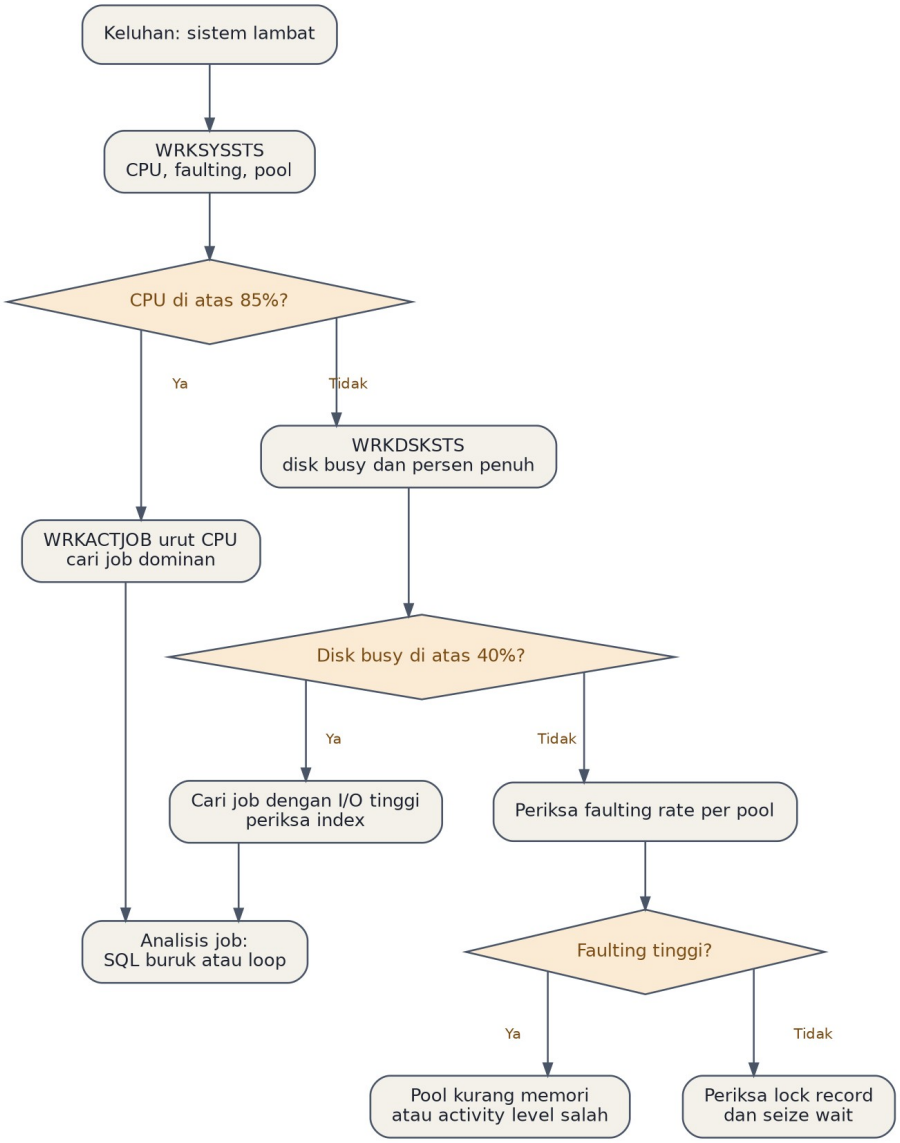
⚡ Bab 22. Performance Engineering



Gambar 27 — Kerangka performance engineering

22.1 Urutan diagnosis yang benar

Kesalahan paling umum adalah langsung menyalahkan CPU. Ikuti urutan ini:



Gambar 28

22.2 Membaca WRKSYSSTS

Kolom	Arti	Yang dicari
% CPU used	Pemakaian prosesor	Konsisten di atas 85% = masalah
Elapsed time	Periode pengukuran	Reset dengan F10 sebelum mengukur
Pool size	Memori pool, KB	Bandingkan dengan beban
Max active	Activity level	Terlalu rendah = job antre di pool
DB faults/pages	Page fault database	Naik tajam = pool kurang memori
Non-DB faults/pages	Page fault non-database	Idem

Aturan praktis untuk pool interaktif: **faulting total di bawah 10 per detik per prosesor**. Untuk pool batch toleransinya lebih tinggi, karena batch memang menyapu data dalam jumlah besar.

22.3 Collection Services

```
/* Memastikan koleksi berjalan */
CHKCOLSRV

/* Memulai dengan interval 5 menit */
STRPFRCOL COLSRVPRF(*CUSTOM) INTERVAL(5) +
          LIB(QMPGDATA) DFTCOLIVL(300) +
          CYCTIME(000000) CYCITV(24) +
          RETPERIOD(30 *DAYS)

/* Membuat file database dari koleksi untuk dianalisis */
CRTPFRDTA FROMMGTCOL(QMPGDATA/Q182000001) +
          TOLIB(PFRDATA)
```

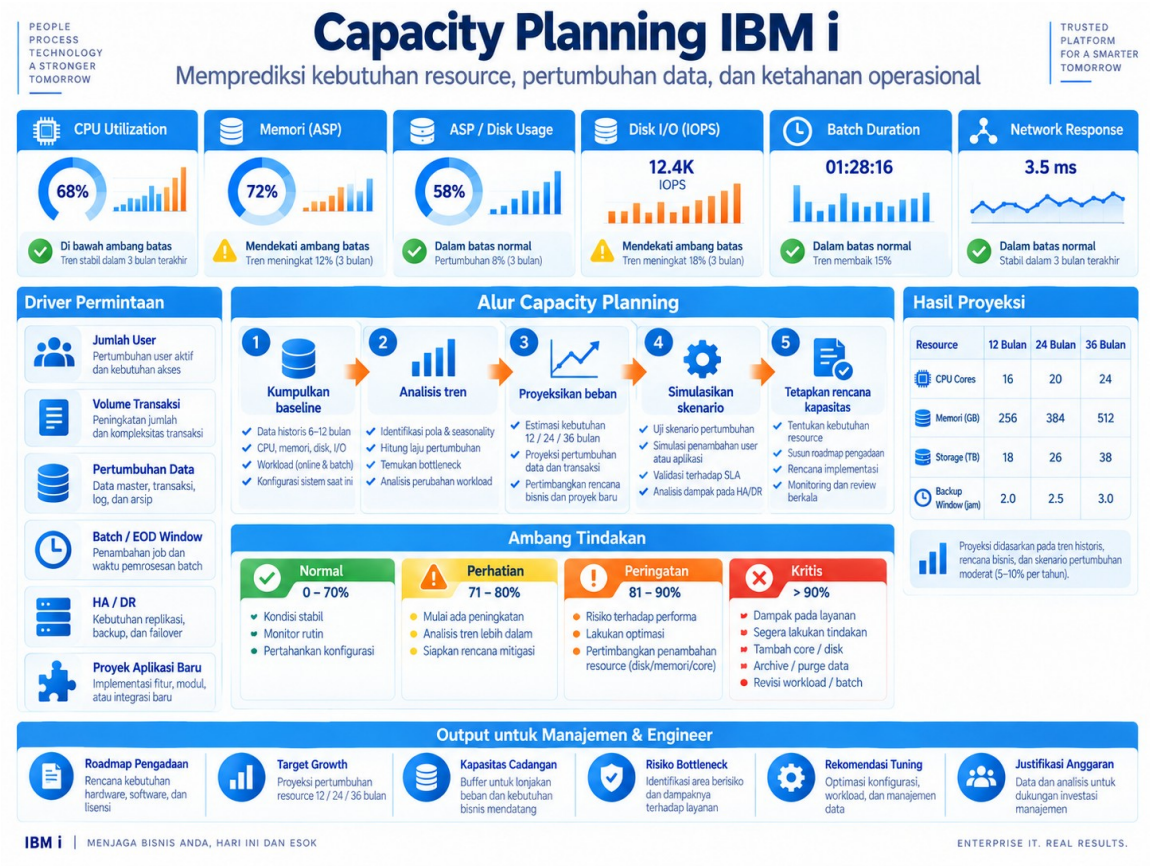
Analisis dilakukan lewat Performance Data Investigator di Navigator for i, atau langsung dengan SQL ke tabel `QAPMxxxx`.

22.4 Menemukan SQL yang mahal

```
-- Index yang disarankan sistem
SELECT TABLE_SCHEMA, TABLE_NAME,
       KEY_COLUMNS_ADVISED,
       TIMES_ADVISED,
       ESTIMATED_CREATION_TIME,
       LAST_ADVISED
  FROM QSYS2.SYSIXADV
 WHERE TIMES_ADVISED > 100
 ORDER BY TIMES_ADVISED DESC;

-- Tabel terbesar yang mungkin butuh reorganisasi
SELECT TABLE_SCHEMA, TABLE_NAME,
       NUMBER_ROWS,
       NUMBER_DELETED_ROWS,
       DATA_SIZE / 1024 / 1024 AS ukuran_mb
  FROM QSYS2.SYSTABLESTAT
 WHERE NUMBER_DELETED_ROWS > 100000
 ORDER BY NUMBER_DELETED_ROWS DESC;
```

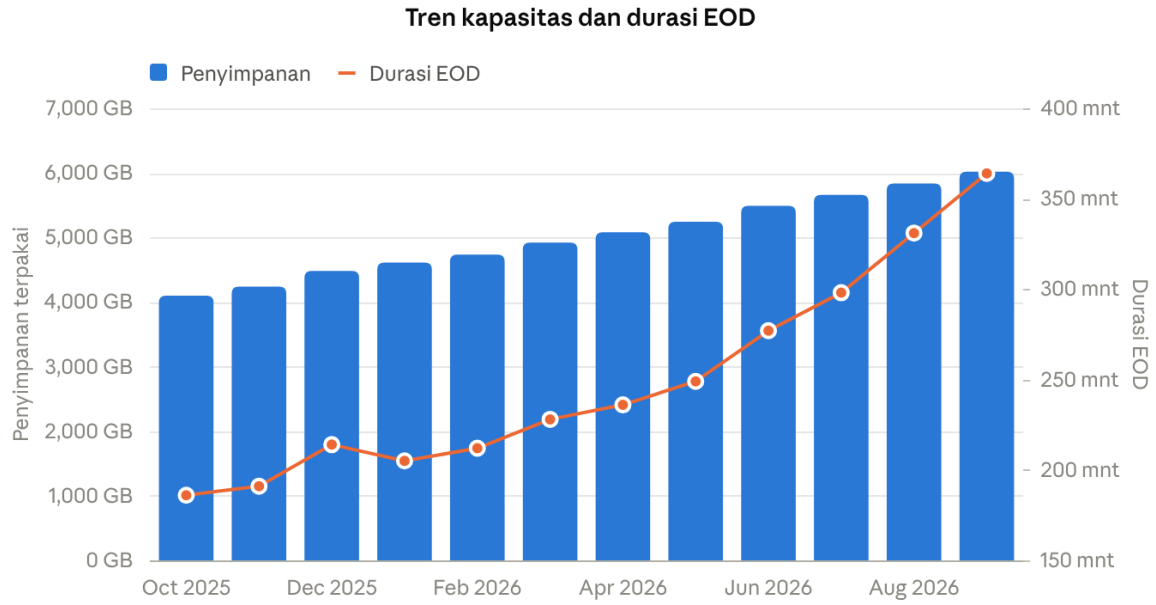
Jangan membuat setiap index yang disarankan. Setiap index menambah beban tulis, dan di lingkungan batch yang menulis jutaan record, index berlebih justru memperlambat EOD.

Bab 23. Capacity Planning



Gambar 29 — Alur perencanaan kapasitas dan ambang tindakan

Perencanaan kapasitas berbasis tren, bukan snapshot. Pola pertumbuhan yang khas pada sistem core banking terlihat seperti ini:



Angka ilustratif untuk latihan perencanaan, bukan data produksi

Gambar 30

Angka ilustratif · 12 bulan

Dua hal terbaca dari pola semacam ini. Pertumbuhan penyimpanan cenderung linier mengikuti volume transaksi. Sebaliknya durasi EOD melompat tidak linier begitu ambang tertentu terlampaui — biasanya saat pool memori mulai kekurangan dan faulting naik tajam. Titik belok itulah yang harus dideteksi jauh sebelum batch window terlampaui.

23.1 Data yang perlu direkam bulanan

Metrik	Sumber	Kegunaan
CPU rata-rata dan puncak	Collection Services	Proyeksi kebutuhan core
Persentase ASP terpakai	`SYSDISKSTAT`	Proyeksi kebutuhan disk
Jumlah transaksi harian	Tabel aplikasi	Korelasi beban
Durasi EOD	Log job scheduler	Deteksi degradasi dini
Jumlah sesi puncak	`ACTIVE_JOB_INFO`	Sizing interaktif
Ukuran journal receiver harian	`WRKJRNA`	Proyeksi ASP journal

23.2 Kapan mulai proses pengadaan

Mulai proses pengadaan ketika proyeksi menunjukkan ambang kritis akan tercapai dalam **dua kali lead time pengadaan**. Jika pengadaan perangkat keras di institusi Anda memakan enam bulan, mulailah saat proyeksi menunjukkan dua belas bulan tersisa.

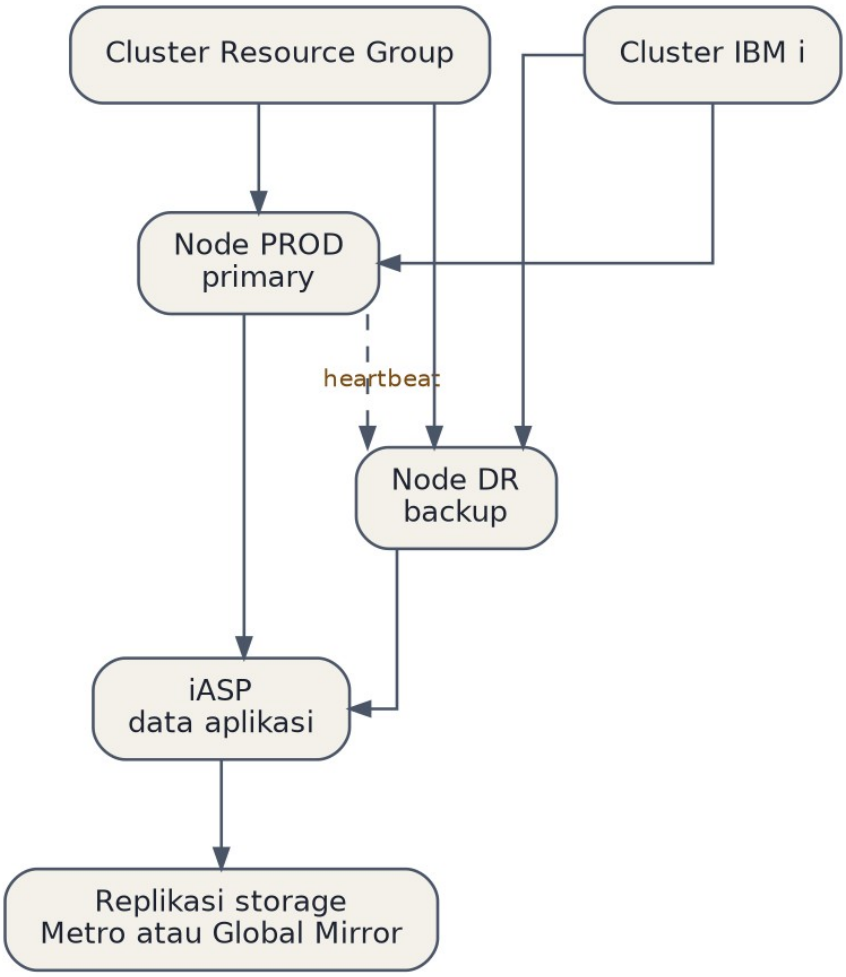
🛟 Bab 24. High Availability dan Disaster Recovery

Gambar 31 — High availability dan disaster recovery

24.1 Perbandingan pendekatan

Pendekatan	RPO	RTO	Biaya	Keterbatasan
Tape restore	Jam sampai hari	Jam sampai hari	Rendah	Tidak layak untuk core banking
Replikasi logis	Detik sampai menit	Menit	Menengah	Perlu penanganan objek non-jurnal
PowerHA dengan iASP	Mendekati nol	Menit	Tinggi	Butuh storage bersama atau replikasi storage
Db2 Mirror	Nol	Mendekati nol	Sangat tinggi	Batasan jarak, kompleksitas tinggi

24.2 Arsitektur PowerHA



Gambar 32

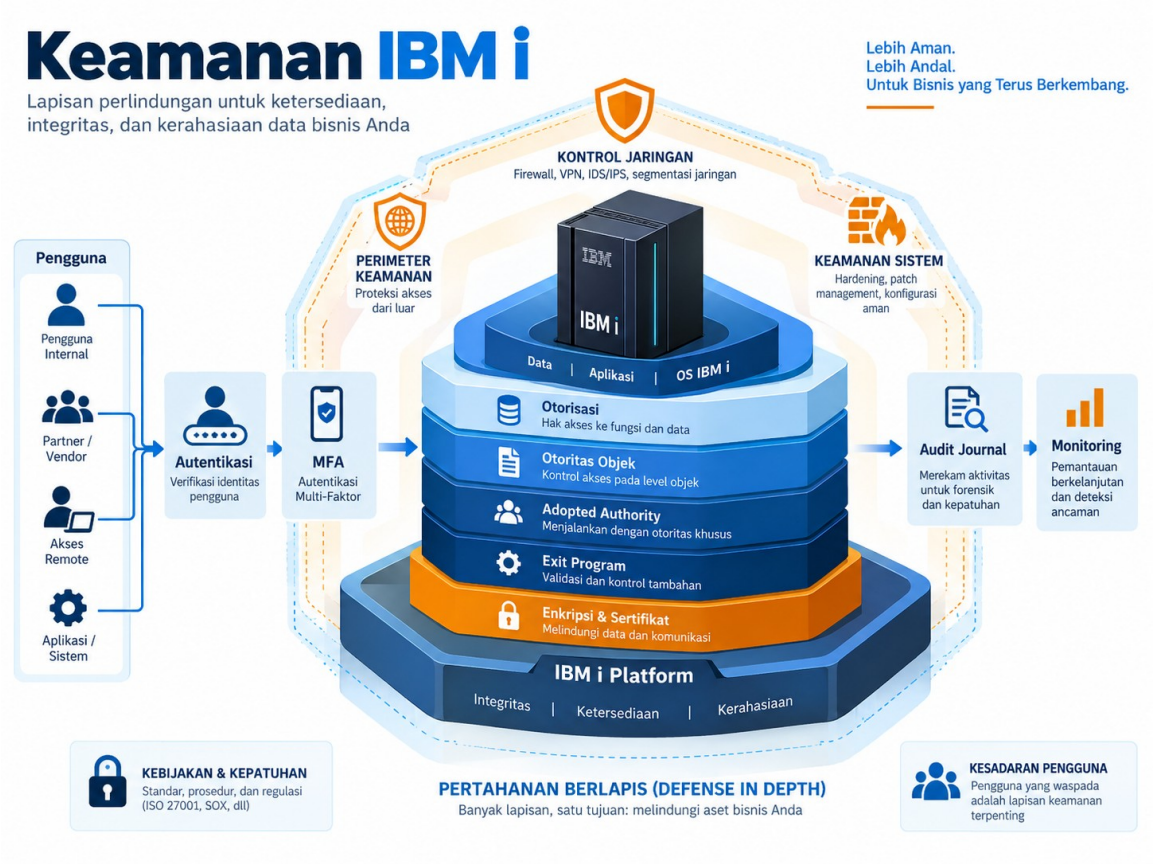
Yang sering luput: **objek di luar iASP tidak ikut berpindah.** User profile, system value, konfigurasi perangkat, job scheduler entry, dan sebagian objek di QSYS harus disinkronkan lewat mekanisme terpisah, biasanya administrative domain.

24.3 Checklist uji DR tahunan

- ☐ Tetapkan skenario: kehilangan total data center utama
- ☐ Umumkan jendela dan bekukan perubahan sebelum uji
- ☐ Catat waktu deklarasi bencana
- ☐ Lakukan switchover atau failover sesuai prosedur tertulis
- ☐ Verifikasi integritas data: rekonsiliasi saldo, hitung record
- ☐ Uji seluruh kanal: ATM switch, mobile, internet banking
- ☐ Jalankan satu siklus EOD lengkap di lingkungan DR
- ☐ Catat RTO aktual dan bandingkan dengan target
- ☐ Lakukan switchback terkendali

- ☐ Tulis berita acara dengan temuan dan rencana perbaikan

🔒 Bab 25. Keamanan IBM i



Gambar 33 — Pertahanan berlapis pada keamanan IBM i

25.1 Tingkat keamanan

Level	Deskripsi	Layak produksi?
20	Password saja, semua pengguna setara `*ALLOBJ`	Tidak
30	Password dan wewenang objek	Minimum absolut, tidak memadai untuk bank
40	Ditambah proteksi integritas sistem	Standar minimum perbankan
50	Ditambah proteksi tingkat lanjut	Disarankan untuk data sangat sensitif

Perpindahan dari 30 ke 40 bisa memutus program lawas yang mengakses antarmuka sistem tidak didukung. Prosedurnya: aktifkan audit `*PGMFAIL`, jalankan di level 30 selama minimal satu siklus bulanan penuh, analisis pelanggaran yang tercatat, perbaiki, baru naikkan level.

```
/* Persiapan naik ke level 40 */
CHGSYSVAL SYSVAL(QAUDLVL) VALUE('*AUTFAIL *PGMFAIL *SECURITY')

/* Setelah satu bulan, ekstraksi pelanggaran */
DSPJRN JRN(QSYS/QAUDJRN) ENTTYP(AF) +
       RCVRNG(*CURCHAIN) +
       OUTPUT(*OUTFILE) OUTFILFMT(*TYPE5) +
       OUTFILE(QTEMP/AUDAF)
```

```
SELECT AFVIOL AS jenis_pelanggaran,
       AFUSER AS pengguna,
       AFPGM  AS program,
       AFOBJ  AS objek,
       COUNT(*) AS jumlah
  FROM QTEMP.AUDAF
 GROUP BY AFVIOL, AFUSER, AFPGM, AFOBJ
 ORDER BY jumlah DESC;
```

25.2 Exit point

Wewenang objek tidak memadai untuk mengendalikan akses lewat ODBC, FTP, atau layanan jaringan lain. Exit program menyaring di titik masuk.

Exit point	Mengendalikan
QIBM_QZDA_INIT	Koneksi ODBC/JDBC
QIBM_QZDA_SQL1	Statement SQL lewat ODBC
QIBM_QTMF_SERVER_REQ	Perintah FTP
QIBM_QPWFS_FILE_SERV	Akses IFS lewat file share
QIBM_QTG_DEVINIT	Sesi Telnet
QIBM_QSY_VLD_PASSWRD	Validasi password kustom

```
/* Melihat exit program terpasang */
WRKREGINF EXITPNT(QIBM_QZDA_INIT)

/* Mendaftarkan exit program */
ADDEXITPGM EXITPNT(QIBM_QZDA_INIT) FORMAT(ZDAI0100) +
           PGMNBR(1) PGM(SECLIB/ODBCEXIT) +
           TEXT('Kendali akses ODBC')
```

25.3 Multi-Factor Authentication di IBM i 7.6

Rilis 7.6 membawa MFA yang berjalan native di dalam sistem operasi, tidak lagi menuntut produk pihak ketiga. Tiga implikasi desain:

1. **Akun layanan tidak bisa memakai MFA interaktif.** Pisahkan dengan tegas akun manusia dan akun aplikasi; yang kedua diamankan dengan pembatasan alamat IP dan exit program.
2. **Prosedur pemulihan wajib disiapkan lebih dulu.** Apa yang terjadi bila perangkat token seorang administrator hilang pukul dua pagi saat EOD gagal?
3. **Uji dampak ke otomasi.** Skrip yang login lewat SSH atau ODBC akan putus bila MFA diterapkan tanpa pengecualian yang direncanakan.

25.4 Row and Column Access Control

RCAC menyembunyikan data di level basis data, sehingga berlaku untuk semua jalur akses termasuk kueri ad hoc.

```
-- Masking nomor rekening kecuali untuk peran tertentu
CREATE MASK DTALIB.MASK_NOREK ON DTALIB.REKENING
  FOR COLUMN NO_REK RETURN
    CASE
      WHEN VERIFY_GROUP_FOR_USER(SESSION_USER,'GRPOPS') = 1
        THEN NO_REK
      ELSE 'XXXXXXXXXXXX' || SUBSTR(NO_REK,13,4)
    END
  ENABLE;

-- Pembatasan baris berdasarkan cabang pengguna
CREATE PERMISSION DTALIB.PERM_CABANG ON DTALIB.REKENING
  FOR ROWS WHERE
    VERIFY_GROUP_FOR_USER(SESSION_USER,'GRPPUSAT') = 1
    OR KD_CABANG = (SELECT KD_CABANG FROM DTALIB.PEGAWAI
                    WHERE USERID = SESSION_USER)
  ENFORCED FOR ALL ACCESS
  ENABLE;

ALTER TABLE DTALIB.REKENING ACTIVATE ROW ACCESS CONTROL;
ALTER TABLE DTALIB.REKENING ACTIVATE COLUMN ACCESS CONTROL;
```

RCAC berlaku juga untuk profil dengan `*ALLOBJ`. Itu menjadikannya kontrol yang jauh lebih kuat daripada wewenang objek biasa, dan jawaban praktis untuk temuan audit klasik "administrator bisa melihat seluruh data nasabah".

🔒 Bab 26. Audit Journal

Gambar 34 — Audit journal: sumber peristiwa sampai tindak lanjut

Audit journal `QAUDJRN` adalah sumber kebenaran untuk pertanyaan audit.

Kode	Peristiwa	Pertanyaan audit yang dijawab
`AF`	Kegagalan wewenang	Siapa mencoba akses yang tidak berhak
`CO`	Pembuatan objek	Objek apa yang dibuat di produksi
`DO`	Penghapusan objek	Siapa menghapus apa
`CP`	Perubahan user profile	Perubahan wewenang pengguna
`PW`	Password salah	Percobaan login paksa
`SV`	Perubahan system value	Perubahan konfigurasi keamanan
`ZC` / `ZR`	Perubahan atau pembacaan objek yang diaudit	Akses ke data sensitif
`RO` / `RU`	Restore objek atau wewenang	Objek yang masuk lewat restore
`ST`	Penggunaan service tool	Akses alat sistem tingkat rendah

```
-- Perubahan system value 30 hari terakhir
SELECT ENTRY_TIMESTAMP,
       USER_NAME,
       SYSTEM_VALUE_NAME,
       OLD_VALUE,
       NEW_VALUE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL(
         'QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'SV',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS));
```

Bab 27. Kepatuhan

27.1 Pemetaan kontrol ke mekanisme IBM i

Kontrol yang diminta	Mekanisme IBM i	Bukti untuk auditor
Kendali akses berbasis peran	Group profile, authorization list	`DSPUSRPRF`, matriks wewenang
Pemisahan tugas	Profil terpisah, `LMTCPB(*YES)`	Matriks RACI, daftar profil
Jejak audit tak terubah	`QAUDJRN`	Ekstraksi journal, bukti receiver tersimpan
Enkripsi saat transit	TLS pada semua server TCP	Konfigurasi DCM, hasil pemindaian
Enkripsi saat diam	Enkripsi ASP atau kolom	Konfigurasi, bukti kunci terkelola
Manajemen password	System value `QPWD*`	Cetakan system value
Pemulihan bencana	PowerHA, uji tahunan	Berita acara uji DR
Retensi log	Kebijakan receiver dan spooled file	Kebijakan tertulis, bukti arsip
Manajemen perubahan	Object promotion, CAB	Tiket change, log promotion

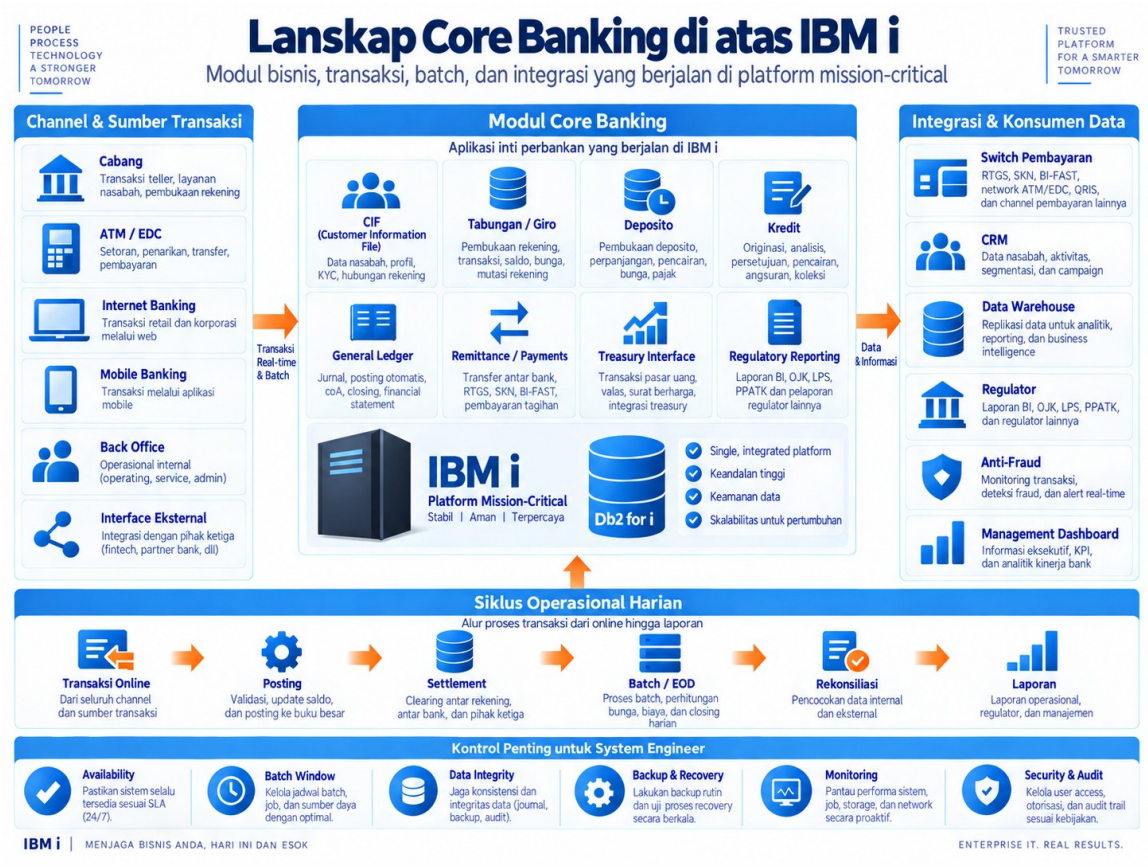
27.2 Kerangka acuan

Untuk institusi keuangan di Indonesia, acuan yang perlu dipetakan bersamaan: ketentuan OJK tentang penyelenggaraan teknologi informasi dan manajemen risiko TI bagi bank umum, UU No. 27 Tahun 2022 tentang Pelindungan Data Pribadi, PCI DSS bila memproses data kartu, dan ISO 27001 sebagai kerangka payung. Verifikasi nomor dan status terkini setiap peraturan sebelum dijadikan rujukan formal — ketentuan OJK cukup sering diperbarui.

🏦 Bagian VI — Core Banking di atas IBM i

Seluruh contoh di bagian ini generik dan teranonimkan. Nama library, job, dan file adalah rekaan untuk keperluan ilustrasi.

Bab 28. Lanskap Core Banking di IBM i



Gambar 35 — Lanskap core banking di atas IBM i

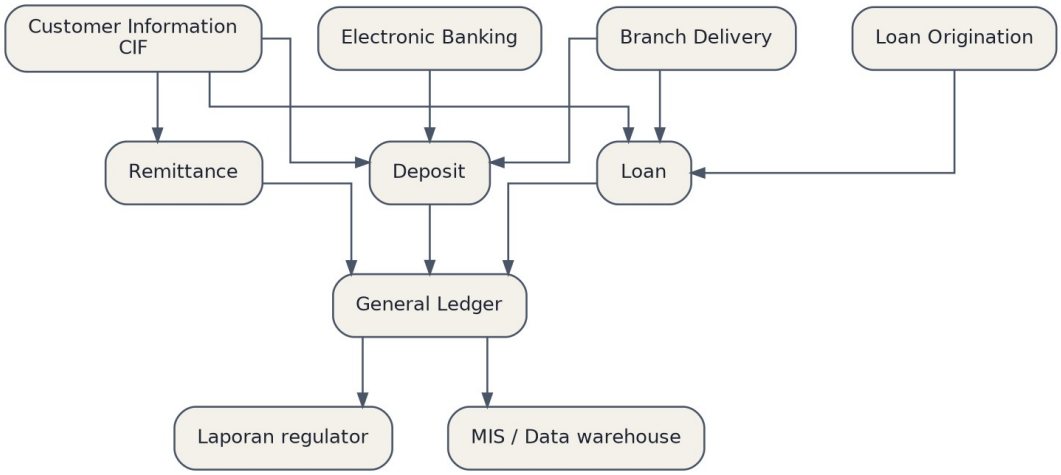
28.1 Dua produk bernama Silverlake

Ini sumber kebingungan yang sering muncul dalam dokumen teknis, dan perlu diluruskan sekali di awal.

	Silverlake Axis SIBS	Jack Henry SilverLake System
Vendor	Silverlake Axis, Malaysia/Singapura	Jack Henry & Associates, Amerika Serikat
Berdiri	1989, pendiri Dr. Goh Peng Ooi	Bagian dari Jack Henry Banking
Pasar	ASEAN, termasuk Indonesia	Community bank di AS
Platform	IBM i on Power	IBM Power i
Relevansi untuk Indonesia	Ini yang dipakai	Tidak relevan

Untuk konteks perbankan Indonesia, yang dimaksud selalu **Silverlake Axis SIBS**. Hampir separuh dari dua puluh bank terbesar di Asia Tenggara berjalan di atas solusi ini, dan versi 12 telah digelar di Indonesia, antara lain di Bank Mandiri Taspen dengan cakupan lebih dari 390 cabang.

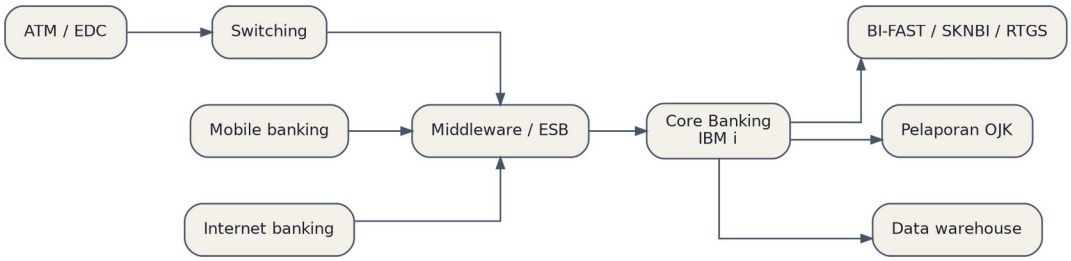
28.2 Modul inti dan aliran data



Gambar 36

CIF adalah pusat gravitasi. Setiap perubahan struktur data nasabah berdampak ke seluruh modul hilir, yang menjelaskan mengapa perubahan di area ini selalu berisiko tinggi.

28.3 Ekosistem di sekitar core



Gambar 37

Implikasi terpenting dari gambar ini bagi System Engineer: **core banking hampir tidak pernah menjadi titik kegagalan yang terlihat langsung oleh nasabah.** Yang terlihat adalah kanal. Tetapi ketika core melambat, seluruh kanal ikut melambat, dan keluhan masuk dari segala arah sekaligus.

🏗️ Bab 29. Arsitektur Lingkungan dan Partisi

29.1 Pemisahan lingkungan

Lingkungan	Data	Siapa punya akses	Refresh
PROD	Nyata	Operasi, terbatas	—
DR	Replika PROD	Sama dengan PROD	Kontinu
UAT	Salinan PROD, di-mask	Bisnis, QA, vendor	Bulanan
SIT	Sintetis atau mask	Pengembang, vendor	Sesuai proyek
DEV	Sintetis	Pengembang	Bebas
MIS	Ekstraksi PROD	Analis, pelaporan	Harian setelah EOD

29.2 Masking data saat refresh ke non-produksi

Ini kewajiban di bawah UU Pelindungan Data Pribadi, bukan praktik opsional.

```
-- Dijalankan setelah restore ke UAT, sebelum akses dibuka
UPDATE DTALIB.NASABAH
   SET NAMA      = 'NASABAH ' || DIGITS(ID_NASABAH),
       NIK       = '32' || SUBSTR(DIGITS(ID_NASABAH),1,14),
       ALAMAT    = 'ALAMAT UJI COBA',
       NO_HP     = '0800' || SUBSTR(DIGITS(ID_NASABAH),1,8),
       EMAIL     = 'uji' || STRIP(CHAR(ID_NASABAH)) || '@contoh.test',
       TGL_LAHIR = DATE('1980-01-01') + (MOD(ID_NASABAH, 7000)) DAYS;

-- Verifikasi tidak ada data asli tersisa
SELECT COUNT(*) AS sisa_data_asli
  FROM DTALIB.NASABAH
 WHERE NAMA NOT LIKE 'NASABAH %';
```

Jalankan masking sebagai bagian dari prosedur restore yang sama, bukan langkah terpisah yang bisa terlupa. Akses ke UAT dibuka hanya setelah kueri verifikasi mengembalikan nol.

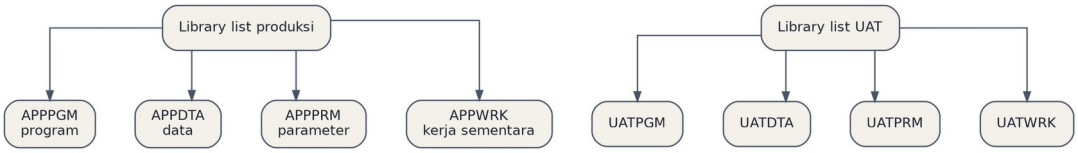
🏦 Bab 30. Administrasi Core Banking bagi System Engineer

30.1 Batas wewenang antar peran

Objek	System Engineer	App Support	Vendor
Library sistem QSYS	Penuh	Baca	Tidak ada
Library program aplikasi	Baca, restore	Baca	Baca dengan pendampingan
Library data aplikasi	Save/restore	Baca terbatas	Tidak ada di PROD
Subsystem dan job queue	Penuh	Kendali job sendiri	Tidak ada
User profile	Buat dan ubah	Tidak ada	Tidak ada
Parameter aplikasi	Tidak ada	Ubah dengan approval	Saran saja

Prinsip pegangan: **System Engineer menguasai wadahnya, Application Support menguasai isinya.** System Engineer yang mengubah parameter bunga adalah pelanggaran pemisahan tugas, sekalipun secara teknis ia mampu.

30.2 Environment separation

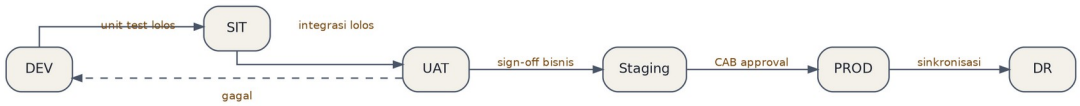


Gambar 38

Bahaya terbesar: job produksi yang tanpa sengaja menunjuk library UAT, atau sebaliknya. Pencegahannya dipasang di awal setiap job kritikal.

```
PGM
   DCL VAR(&SYSNAME) TYPE(*CHAR) LEN(8)
   DCL VAR(&PESAN)   TYPE(*CHAR) LEN(132)

   RTVNETA SYSNAME(&SYSNAME)

   /* Di mesin produksi, library UAT tidak boleh ada di libl */
   IF COND(&SYSNAME *EQ 'PRDIBMI') THEN(DO)
      CHKOBJ OBJ(UATDTA) OBJTYPE(*LIB)
      MONMSG MSGID(CPF9801) EXEC(GOTO CMDLBL(AMAN))

      CHGVAR VAR(&PESAN) VALUE('FATAL: library UAT terdeteksi +
         di lingkungan produksi. Job dihentikan.')
      SNDPGMMSG MSG(&PESAN) TOPGMQ(*EXT) MSGTYPE(*ESCAPE)
   ENDDO

 AMAN:
   RETURN
ENDPGM
```

30.3 Object promotion



Gambar 39

Tiga aturan yang tidak boleh dilanggar:

1. **Kompilasi hanya sekali, di lingkungan paling awal.** Objek yang sama dipindahkan naik, tidak dikompilasi ulang di tiap tingkat. Kompilasi ulang berarti yang diuji bukan yang dipasang.
2. **Promosi memakai akun layanan, bukan akun perorangan.** Jejaknya harus menunjuk tiket, bukan orang yang kebetulan berjaga.
3. **Setiap promosi menyimpan salinan objek lama.** Rollback harus bisa dieksekusi dalam hitungan menit.

```
PGM PARM(&LIB &OBJ &TIKET)
   DCL VAR(&LIB)   TYPE(*CHAR) LEN(10)
   DCL VAR(&OBJ)   TYPE(*CHAR) LEN(10)
   DCL VAR(&TIKET) TYPE(*CHAR) LEN(15)
   DCL VAR(&SAVF)  TYPE(*CHAR) LEN(10)

   CHGVAR VAR(&SAVF) VALUE('B' *CAT %SST(&TIKET 1 9))

   CRTSAVF FILE(ROLLBACK/&SAVF) +
           TEXT('Backup sebelum promosi ' *CAT &TIKET)
   MONMSG MSGID(CPF5813)

   SAVOBJ OBJ(&OBJ) LIB(&LIB) DEV(*SAVF) +
          SAVF(ROLLBACK/&SAVF) OBJTYPE(*ALL)
   MONMSG MSGID(CPF3770) EXEC(DO)
      SNDMSG MSG('Objek belum ada, promosi pertama') TOUSR(SYSENG)
   ENDDO
ENDPGM
```

30.4 Parameter aplikasi yang wajib dipantau

Walaupun mengubahnya bukan wewenang Anda, tiga parameter ini berdampak langsung ke operasi sistem.

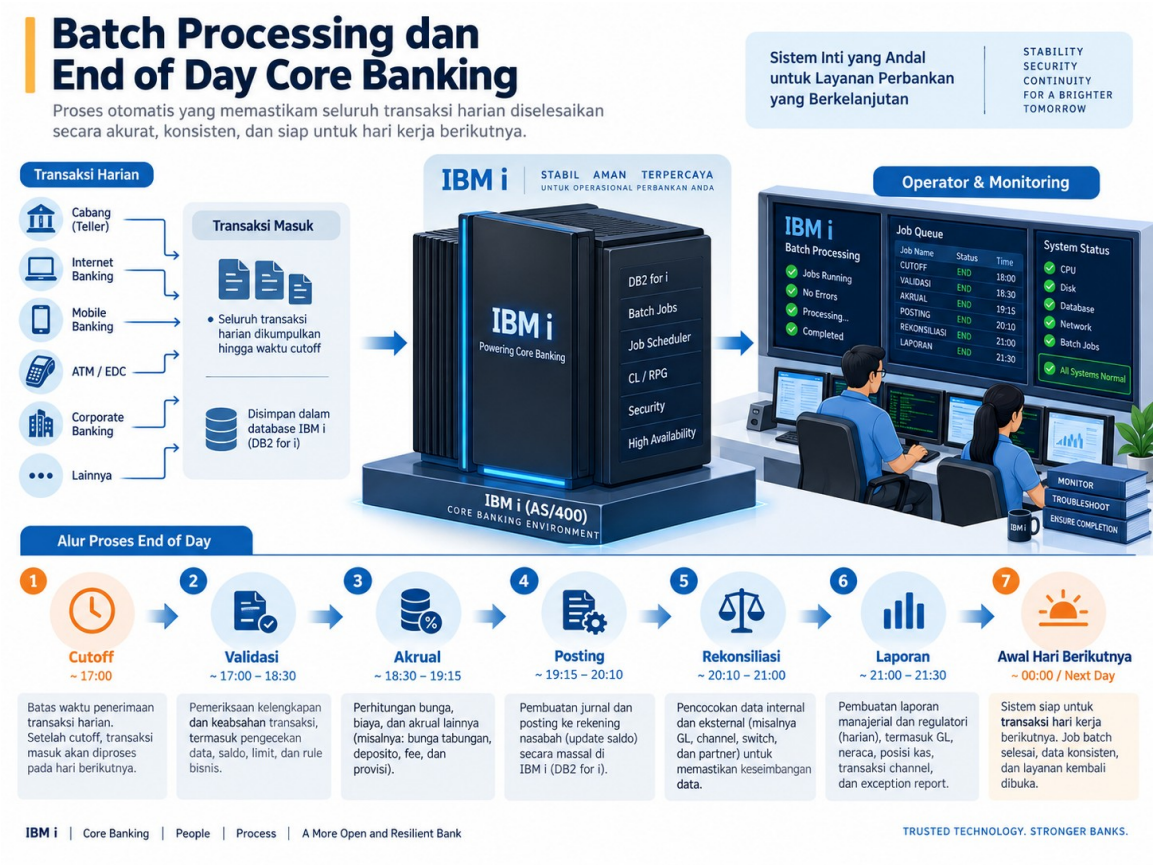
Parameter	Dampak teknis	Gejala bila salah
Kalender hari kerja	Menentukan apakah EOD berjalan	EOD tidak jalan di hari libur, atau sebaliknya
Cut-off time	Titik pemisah tanggal buku	Transaksi masuk ke tanggal buku yang salah
Tanggal buku aplikasi	Dasar seluruh perhitungan akrual	Selisih dengan `QDATE` mengacaukan semuanya

Pemeriksaan pagi yang murah dan sangat berguna:

```
SELECT (SELECT TGL_BUKU FROM APPPRM.PARAM_SISTEM
         WHERE KD_PARAM = 'TGLBUKU') AS tgl_buku_aplikasi,
       CURRENT DATE AS tgl_sistem,
       DAYS(CURRENT DATE) -
       DAYS((SELECT TGL_BUKU FROM APPPRM.PARAM_SISTEM
              WHERE KD_PARAM = 'TGLBUKU')) AS selisih_hari
  FROM SYSIBM.SYSDUMMY1;
```

Selisih lebih dari satu hari pada pagi hari kerja berarti EOD semalam tidak selesai. Jadikan ini pemeriksaan pertama setiap pagi, sebelum membuka email.

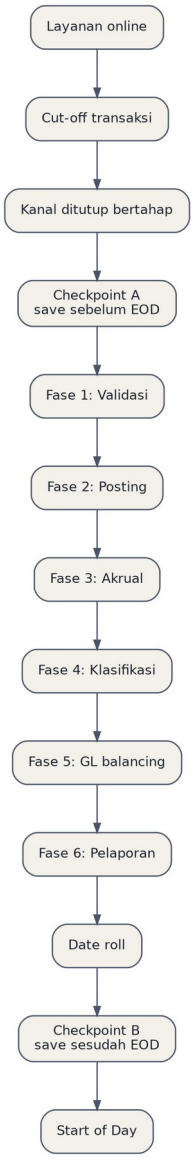
🌙 Bab 31. Batch Processing dan End of Day



Gambar 40 — Batch processing dan End of Day core banking

Ini bab terpenting di buku ini. End of Day adalah titik di mana kegagalan teknis langsung menjadi kegagalan bisnis yang terlihat regulator.

31.1 Anatomi siklus harian



Gambar 41

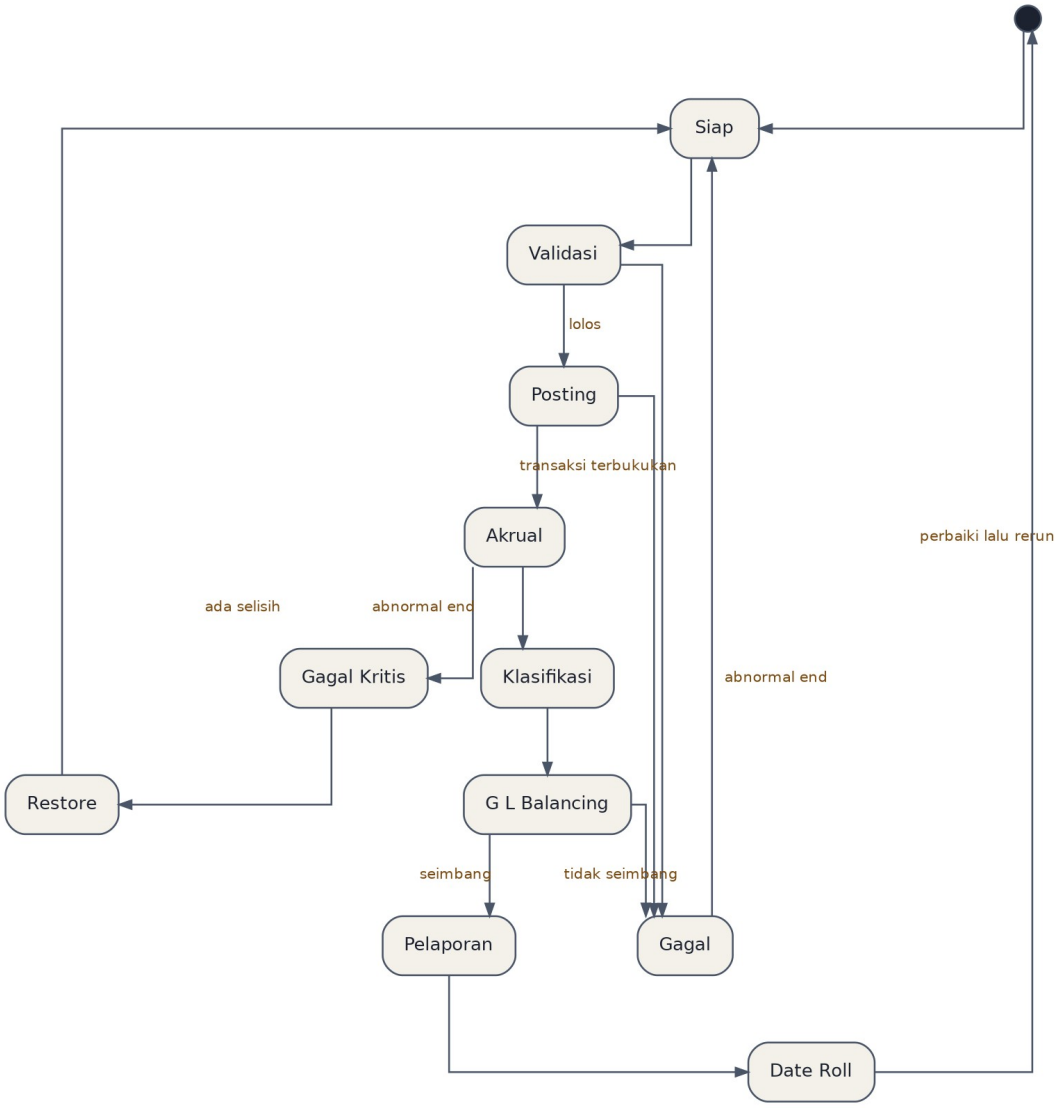
31.2 Karakteristik tiap fase

Fase	Sifat	Boleh diulang?	Jika gagal
Validasi	Baca saja	Ya, bebas	Perbaiki data, ulang
Posting	Tulis, idempoten bila dirancang benar	Ya, dari restart point	Rerun dari checkpoint
Akrual	Tulis, **tidak idempoten**	Tidak tanpa reversal	Wajib restore, bukan rerun
Klasifikasi	Tulis, idempoten	Ya	Rerun
GL balancing	Baca dan tulis kontrol	Ya	Selidiki selisih dulu
Pelaporan	Baca, hasilkan file	Ya, bebas	Rerun

Fase akrual adalah yang paling berbahaya. Menjalankan ulang perhitungan bunga tanpa membalik hasil sebelumnya akan membukukan bunga dua kali. Ini kesalahan yang berujung pada koreksi manual ribuan

rekening dan laporan insiden ke regulator.

31.3 Rantai ketergantungan dan restart point



Gambar 42

Perhatikan pembedaan `Gagal` dan `GagalKritis`. Kegagalan di fase akrual tidak boleh ditangani dengan rerun biasa.

31.4 Tabel kontrol EOD

Setiap job EOD harus mencatat statusnya. Tanpa ini, tidak ada cara menjawab pertanyaan "sudah sampai mana" pada pukul tiga pagi.

```
CREATE TABLE APPWRK.EOD_KONTROL (
    TGL_BUKU     DATE          NOT NULL,
    URUTAN       SMALLINT      NOT NULL,
    KD_FASE      CHAR(10)      NOT NULL,
    NM_FASE      VARCHAR(60)   NOT NULL,
    STATUS       CHAR(1)       NOT NULL DEFAULT 'P',
    JAM_MULAI    TIMESTAMP,
    JAM_SELESAI  TIMESTAMP,
    JML_RECORD   BIGINT        DEFAULT 0,
    JOB_NAME     VARCHAR(50),
    PESAN_ERROR  VARCHAR(500),
    PRIMARY KEY (TGL_BUKU, URUTAN)
);

-- Status: P=pending, R=running, S=selesai, E=error, K=dilewati
```

Program CL pembungkus yang menjalankan satu fase dan mencatat hasilnya:

```
PGM PARM(&URUTAN &KDFASE &PGMFASE)
   DCL VAR(&URUTAN)  TYPE(*CHAR) LEN(2)
   DCL VAR(&KDFASE)  TYPE(*CHAR) LEN(10)
   DCL VAR(&PGMFASE) TYPE(*CHAR) LEN(10)
   DCL VAR(&JOBNAME) TYPE(*CHAR) LEN(10)
   DCL VAR(&MSGID)   TYPE(*CHAR) LEN(7)
   DCL VAR(&MSGDTA)  TYPE(*CHAR) LEN(500)

   RTVJOBA JOB(&JOBNAME)

   /* Tandai fase dimulai */
   RUNSQL SQL('UPDATE APPWRK.EOD_KONTROL +
               SET STATUS = ''R'', JAM_MULAI = CURRENT TIMESTAMP, +
                   JOB_NAME = ''' *CAT &JOBNAME *TCAT ''' +
               WHERE URUTAN = ' *CAT &URUTAN *TCAT '') +
          COMMIT(*NONE)

   /* Jalankan program fase, tangkap semua escape message */
   CALL PGM(APPPGM/&PGMFASE)
   MONMSG MSGID(CPF0000 MCH0000 RNX0000 RNQ0000) EXEC(DO)
      RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)

      RUNSQL SQL('UPDATE APPWRK.EOD_KONTROL +
                  SET STATUS = ''E'', JAM_SELESAI = CURRENT TIMESTAMP, +
                      PESAN_ERROR = ''' *CAT &MSGID *TCAT ' ' +
                                    *CAT %SST(&MSGDTA 1 200) *TCAT ''' +
                  WHERE URUTAN = ' *CAT &URUTAN *TCAT '') +
             COMMIT(*NONE)

      /* Bangunkan manusia */
      SNDMSG MSG('EOD GAGAL fase ' *CAT &KDFASE *BCAT &MSGID) +
             TOUSR(EODOPS) MSGTYPE(*INQ)
      SNDPGMMSG MSG('EOD dihentikan pada fase ' *CAT &KDFASE) +
                MSGTYPE(*ESCAPE)
   ENDDO

   /* Tandai fase selesai */
   RUNSQL SQL('UPDATE APPWRK.EOD_KONTROL +
               SET STATUS = ''S'', JAM_SELESAI = CURRENT TIMESTAMP +
               WHERE URUTAN = ' *CAT &URUTAN *TCAT '') +
          COMMIT(*NONE)
ENDPGM
```

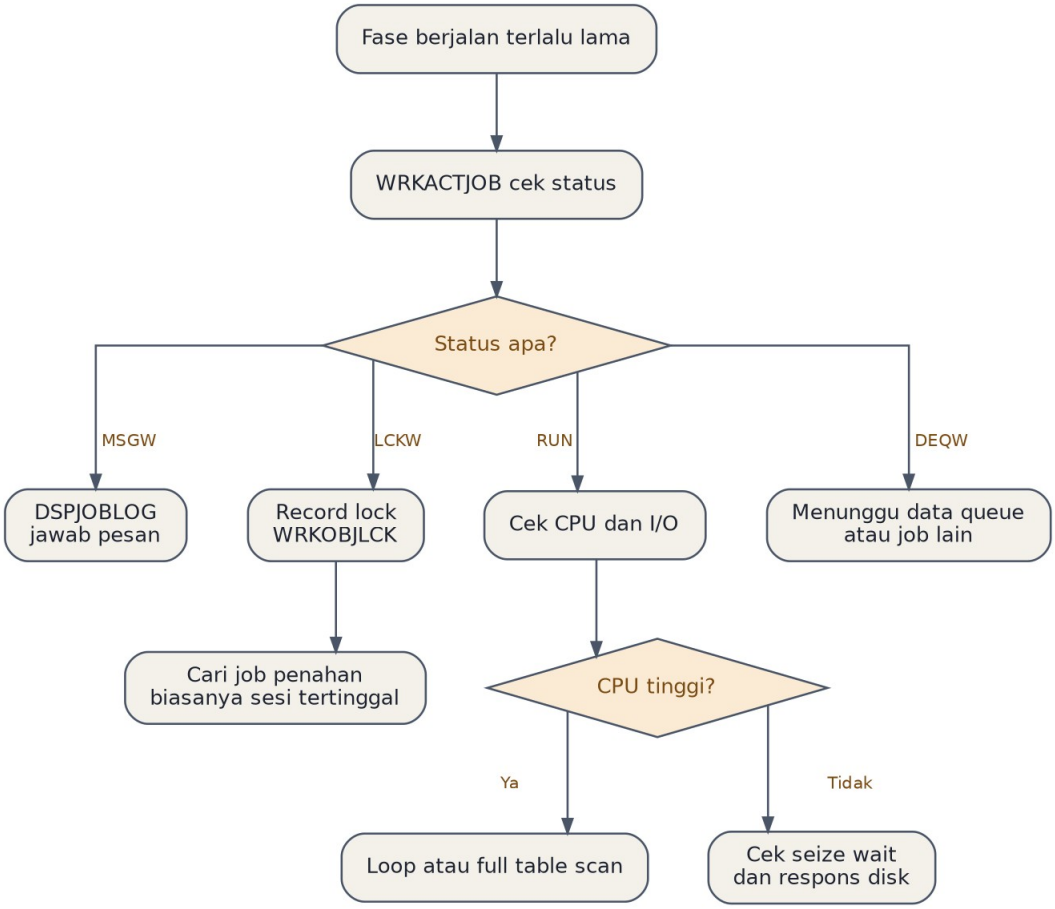
31.5 Memantau EOD yang sedang berjalan

```
-- Dashboard progres EOD malam ini
SELECT URUTAN,
       NM_FASE,
       CASE STATUS
         WHEN 'P' THEN 'Menunggu'
         WHEN 'R' THEN 'Berjalan'
         WHEN 'S' THEN 'Selesai'
         WHEN 'E' THEN 'GAGAL'
         WHEN 'K' THEN 'Dilewati'
       END AS status,
       JAM_MULAI,
       JAM_SELESAI,
       JML_RECORD,
       PESAN_ERROR
  FROM APPWRK.EOD_KONTROL
 WHERE TGL_BUKU = (SELECT TGL_BUKU FROM APPPRM.PARAM_SISTEM
                    WHERE KD_PARAM = 'TGLBUKU')
 ORDER BY URUTAN;
```

Bandingkan durasi tiap fase dengan rata-rata tujuh hari terakhir. Fase yang tiba-tiba dua kali lebih lama adalah peringatan dini, bukan anomali yang boleh diabaikan.

```
-- Deteksi degradasi dibanding baseline seminggu
WITH baseline AS (
  SELECT URUTAN, NM_FASE,
         AVG(TIMESTAMPDIFF(4, CHAR(JAM_SELESAI - JAM_MULAI))) AS rata_menit
    FROM APPWRK.EOD_KONTROL
   WHERE STATUS = 'S'
     AND TGL_BUKU BETWEEN CURRENT DATE - 8 DAYS AND CURRENT DATE - 1 DAY
   GROUP BY URUTAN, NM_FASE
),
malam_ini AS (
  SELECT URUTAN,
         TIMESTAMPDIFF(4, CHAR(JAM_SELESAI - JAM_MULAI)) AS menit
    FROM APPWRK.EOD_KONTROL
   WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'S'
)
SELECT b.NM_FASE,
       DECIMAL(b.rata_menit,7,1) AS baseline_menit,
       m.menit AS malam_ini,
       DECIMAL((m.menit - b.rata_menit) / b.rata_menit * 100,7,1) AS persen_naik
  FROM baseline b JOIN malam_ini m ON b.URUTAN = m.URUTAN
 WHERE m.menit > b.rata_menit * 1.3
 ORDER BY persen_naik DESC;
```

31.6 Diagnosis EOD yang macet



Gambar 43

```
/* Job yang menahan objek */
WRKOBJLCK OBJ(DTALIB/REKENING) OBJTYPE(*FILE)

/* Melihat apa yang dikerjakan job tanpa mengganggunya */
WRKJOB JOB(123456/EODUSER/EODPOST) OPTION(*PGMSTK)

/* Job log job yang masih berjalan */
DSPJOBLOG JOB(123456/EODUSER/EODPOST)
```

Versi SQL, jauh lebih cepat dibaca saat panik:

```
SELECT JOB_NAME, LOCK_STATE, LOCK_SCOPE,
       TABLE_NAME, RRN AS nomor_record
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'DTALIB'
 ORDER BY TABLE_NAME;

-- Job EOD yang menunggu paling lama
SELECT JOB_NAME, JOB_STATUS, FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         SUBSYSTEM_LIST_FILTER => 'EODSBS'))
 WHERE JOB_STATUS IN ('LCKW','MSGW','DEQW')
 ORDER BY ELAPSED_TIME DESC;
```

31.7 Matriks keputusan saat EOD gagal

Kondisi	Tindakan	Siapa memutuskan
Fase baca saja gagal	Rerun langsung	Operator EOD
Posting gagal sebelum commit	Rerun dari restart point	App Support
Posting gagal setelah commit sebagian	Rerun bila idempoten, jika tidak restore	App Support + System Engineer
Fase akrual gagal di tengah	**Restore, jangan rerun**	Manajer TI + App Support
GL tidak seimbang	Hentikan, selidiki, jangan lanjutkan	Manajer TI + Akuntansi
Batch window hampir habis	Eskalasi, siapkan opsi buka layanan terlambat	Manajemen bisnis

Aturan yang harus tertulis dan disepakati sebelum insiden terjadi: **tidak ada seorang pun boleh memutuskan rerun fase akrual sendirian pada pukul tiga pagi.** Keputusan itu menuntut dua orang dan satu catatan tertulis.

31.8 Batas waktu keputusan

Waktu	Keputusan yang harus sudah diambil
02:00	Bila belum melewati posting, eskalasi ke manajer TI
03:30	Bila belum melewati akrual, siapkan rencana buka layanan terlambat
05:00	Keputusan final: lanjutkan, atau buka dengan tanggal buku kemarin
06:00	Informasikan ke bisnis, siapkan komunikasi ke nasabah

Angka-angka ini disesuaikan dengan jam layanan institusi Anda, tetapi kerangkanya harus ada. Keputusan tanpa batas waktu yang disepakati selalu diambil terlambat.

💾 Bab 32. Backup dan Recovery Lingkungan Core Banking

32.1 Posisi backup dalam siklus harian

Checkpoint A adalah yang paling kritikal — inilah titik yang dipakai ketika fase akrual gagal dan rerun tidak dimungkinkan.

```
PGM
   DCL VAR(&TGL)  TYPE(*CHAR) LEN(6)
   DCL VAR(&SAVF) TYPE(*CHAR) LEN(10)
```

```
   RTVSYSVAL SYSVAL(QDATE) RTNVAR(&TGL)
   CHGVAR VAR(&SAVF) VALUE('PRE' *CAT &TGL)

   /* Pastikan tidak ada sesi aplikasi tersisa */
   ENDSBS SBS(QINTER) OPTION(*IMMED)
   DLYJOB DLY(60)

   /* Save while active agar jendela pendek */
   SAVLIB LIB(DTALIB APPPRM) DEV(TAPMLB01) +
          SAVACT(*SYNCLIB) +
          SAVACTMSGQ(QGPL/SAVMSGQ) +
          SAVACTWAIT(300) +
          ACCPTH(*YES) CLEAR(*ALL)
   MONMSG MSGID(CPF0000) EXEC(DO)
      SNDMSG MSG('FATAL: save sebelum EOD gagal. +
                  EOD TIDAK BOLEH dimulai.') +
             TOUSR(EODOPS) MSGTYPE(*INQ)
      SNDPGMMSG MSG('Save pre-EOD gagal') MSGTYPE(*ESCAPE)
   ENDDO

   /* Simpan juga journal receiver */
   SAVOBJ OBJ(*ALL) LIB(JRNLIB) OBJTYPE(*JRNRCV) +
          DEV(TAPMLB01) SEQNBR(*END)

   SNDMSG MSG('Save pre-EOD selesai, EOD boleh dimulai') TOUSR(EODOPS)
ENDPGM
```

Pola penting di sini: **bila save gagal, EOD tidak boleh dimulai.** Menjalankan EOD tanpa titik pulih berarti mempertaruhkan seluruh data hari itu.

32.2 Skenario pemulihan

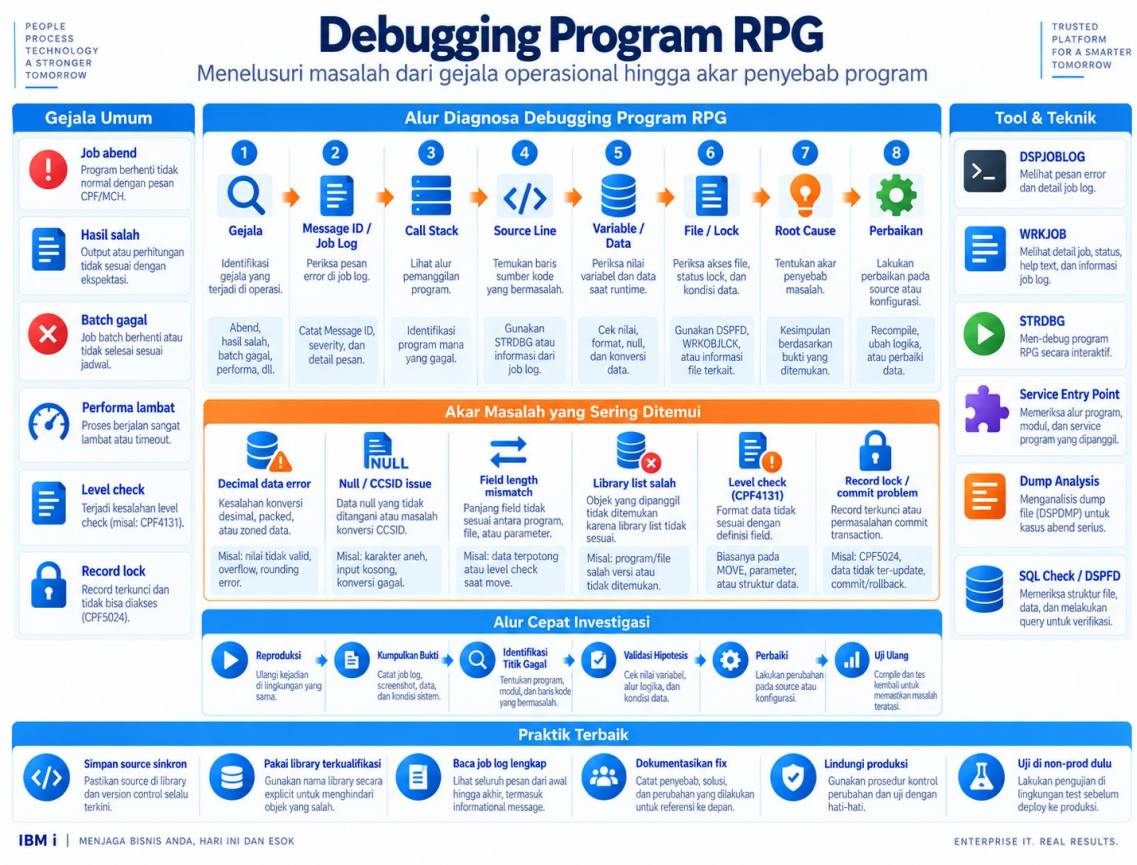
Skenario	Sumber pemulihan	Perkiraan durasi	Risiko
Satu program rusak	Salinan rollback	Menit	Rendah
Satu tabel korup	Restore tabel + apply journal	1–3 jam	Sedang
Fase akrual salah	Restore Checkpoint A + rerun EOD	3–6 jam	Tinggi
Library data hilang	Restore penuh + apply journal	4–8 jam	Tinggi
Partisi hilang	Failover ke DR	Sesuai RTO	Sangat tinggi

32.3 Apply journal setelah restore

```
/* 1. Restore dari save */
RSTLIB SAVLIB(DTALIB) DEV(TAPMLB01) RSTLIB(DTALIB)

/* 2. Terapkan perubahan dari journal sampai titik waktu tertentu */
APYJRNCHG JRN(JRNLIB/JRNDTA) +
          FILE((DTALIB/*ALL)) +
          RCVRNG(*LASTSAVE) +
          TOTIME('18/09/2026' '16:00:00') +
          OPTION(*IGNINQMSG)

/* 3. Verifikasi kuantitatif */
RUNSQL SQL('INSERT INTO APPWRK.VERIFIKASI +
            SELECT CURRENT TIMESTAMP, ''REKENING'', +
                   COUNT(*), SUM(SALDO) FROM DTALIB.REKENING') +
       COMMIT(*NONE)
```

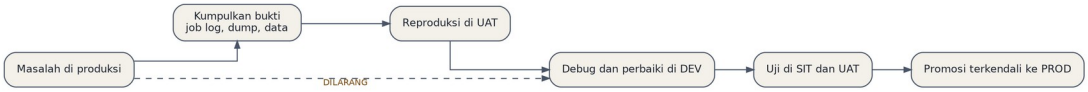
Selalu tutup pemulihan dengan verifikasi kuantitatif: jumlah record dan total saldo kontrol. Restore yang "selesai tanpa error" tetapi menghasilkan total saldo berbeda adalah bencana yang lebih buruk daripada restore yang gagal terang-terangan.

🔍 Bab 33. Debugging Program RPG



Gambar 44 — Alur diagnosis debugging program RPG

33.1 Prinsip yang tidak bisa ditawar



Gambar 45

Debug di produksi diperbolehkan hanya untuk **mengamati**, tidak untuk mengubah. Sesi debug produksi harus tercatat dalam tiket, berbatas waktu, dilakukan dengan profil khusus yang diaudit, dan tidak pernah memakai kemampuan mengubah isi variabel.

33.2 Langkah pertama selalu job log

```
/* Job log lengkap dengan pesan tingkat kedua */
DSPJOBLOG JOB(123456/EODUSER/EODPOST) OUTPUT(*PRINT)

/* Pastikan job berjalan dengan logging memadai */
CHGJOB JOB(123456/EODUSER/EODPOST) LOG(4 00 *SECLVL) +
       LOGCLPGM(*YES)
```

Membaca job log lewat SQL, jauh lebih cepat untuk mencari pola:

```
SELECT ORDINAL_POSITION,
       MESSAGE_ID,
       MESSAGE_TYPE,
       SEVERITY,
       MESSAGE_TEXT,
       MESSAGE_SECOND_LEVEL_TEXT,
       FROM_PROGRAM,
       FROM_PROCEDURE,
       FROM_STATEMENT
  FROM TABLE(QSYS2.JOBLOG_INFO('123456/EODUSER/EODPOST'))
 WHERE SEVERITY >= 30
 ORDER BY ORDINAL_POSITION;
```

`FROM_PROCEDURE` dan `FROM_STATEMENT` adalah dua kolom paling berharga: keduanya menunjuk baris persis di dalam program ILE tempat masalah terjadi.

33.3 Program contoh untuk latihan debug

Program berikut menghitung bunga harian, dan mengandung dua kesalahan yang khas ditemui di lingkungan nyata.

```
**free
ctl-opt dftactgrp(*no) actgrp('EODGRP') option(*srcstmt: *nodebugio);

dcl-f REKENING usage(*update) keyed;
dcl-f BUNGALOG usage(*output);

dcl-s wSaldo       packed(17:2);
dcl-s wRate        packed(7:6);
dcl-s wBunga       packed(17:2);
dcl-s wHariSetahun int(5) inz(365);
dcl-s wJmlProses   int(10) inz(0);

dcl-pr GetRate packed(7:6);
  pKdProduk char(4) const;
end-pr;

read REKENING;
dow not %eof(REKENING);

  if STATUS = 'A';
     wSaldo = SALDO;
     wRate  = GetRate(KD_PRODUK);

     // KESALAHAN 1: pembagian sebelum perkalian
     // presisi hilang untuk rate kecil
     wBunga = wSaldo * (wRate / wHariSetahun);

     // KESALAHAN 2: rate tidak ditemukan mengembalikan 0
     // bunga jadi 0 tanpa peringatan apa pun

     SALDO = SALDO + wBunga;
     update REKENINGR;

     LOG_NOREK = NO_REK;
     LOG_BUNGA = wBunga;
     LOG_TGL   = %date();
     write BUNGALOGR;

     wJmlProses += 1;
```

```
  endif;

  read REKENING;
enddo;

*inlr = *on;
```

Perbaikan untuk keduanya:

```
// Perbaikan 1: kalikan dulu, bagi terakhir, presisi antara lebih besar
dcl-s wHitung packed(30:10);

wHitung = wSaldo * wRate;
wBunga  = %dech(wHitung / wHariSetahun: 17: 2);

// Perbaikan 2: validasi rate secara eksplisit
wRate = GetRate(KD_PRODUK);
if wRate = 0;
   // jangan diam-diam membukukan nol
   sndEscape('Rate tidak ditemukan untuk produk ' + KD_PRODUK);
   return;
endif;
```

Pelajaran yang berlaku jauh di luar contoh ini: **nilai nol yang berasal dari data tidak ditemukan harus dibedakan dari nol hasil perhitungan.** Kesalahan semacam ini tidak memunculkan error, tidak tercatat di job log, dan baru ketahuan berminggu-minggu kemudian saat nasabah mengeluh bunganya tidak masuk.

33.4 Menyiapkan program agar bisa di-debug

```
/* Kompilasi dengan debug view lengkap */
CRTBNDRPG PGM(APPPGM/HITBUNGA) +
          SRCSTMF('/src/hitbunga.rpgle') +
          DBGVIEW(*SOURCE) OPTION(*EVENTF) +
          ACTGRP('EODGRP')

/* Untuk modul ILE */
CRTRPGMOD MODULE(APPPGM/HITBUNGA) +
          SRCSTMF('/src/hitbunga.rpgle') +
          DBGVIEW(*ALL)

CRTPGM PGM(APPPGM/EODBUNGA) +
       MODULE(APPPGM/HITBUNGA APPPGM/UTILITY) +
       BNDSRVPGM(APPPGM/SVCRATE) ACTGRP('EODGRP')
```

Nilai DBGVIEW	Apa yang terlihat saat debug
`*NONE`	Tidak bisa di-debug
`*STMT`	Hanya nomor statement, tanpa kode sumber
`*SOURCE`	Kode sumber asli
`*LIST`	Listing kompilasi, termasuk isi copybook
`*ALL`	Semua view di atas

Untuk program dengan banyak `/copy`, pakai `*LIST`. Ini satu-satunya view yang menampilkan isi copybook yang sudah diekspansi, dan sebagian besar bug sulit justru bersembunyi di sana.

33.5 Debug job batch yang sedang berjalan

Inilah teknik yang paling sering dibutuhkan saat EOD bermasalah.

```
/* 1. Dari sesi Anda, layani job target */
STRSRVJOB JOB(123456/EODUSER/EODPOST)

/* 2. Mulai debug pada program yang dijalankan job tersebut */
STRDBG PGM(APPPGM/HITBUNGA) UPDPROD(*NO)

/* 3. Pasang breakpoint lewat layar debug: F6 pada baris terpilih */

/* 4. Pasang watch: berhenti saat variabel berubah */
WATCH VAR(wBunga)

/* 5. Setelah selesai, WAJIB dibersihkan */
ENDDBG
ENDSRVJOB
```

Parameter `UPDPROD(*NO)` adalah pengaman terpenting: ia mencegah pengubahan objek di library produksi selama sesi debug. Jangan pernah menyetelnya ke `*YES` di produksi.

Perintah dalam sesi debug	Fungsi
`EVAL nama_variabel`	Menampilkan isi variabel
`EVAL struktur.subfield`	Isi subfield data structure
`EVAL array(1)`	Elemen array tertentu
`EVAL %SUBSTR(teks 1 20)`	Sebagian isi variabel teks
`BREAK baris`	Pasang breakpoint
`BREAK baris WHEN kondisi`	Breakpoint bersyarat
`WATCH variabel`	Berhenti saat nilai berubah
`STEP 1`	Jalankan satu langkah
`DISPLAY MODULE`	Pindah antar modul ILE
`ATTR variabel`	Tipe dan panjang variabel

Breakpoint bersyarat adalah yang paling berguna di lingkungan batch. Berhenti pada setiap dari dua juta record tidak ada gunanya; berhenti hanya pada record bermasalah baru berguna:

```
BREAK 42 WHEN NO_REK = '1234567890123456'
BREAK 58 WHEN wBunga < 0
```

33.6 Debug service program

Kekeliruan umum: memanggil `STRDBG` pada program utama saja, lalu bingung mengapa breakpoint di service program tidak pernah kena.

```
STRDBG PGM(APPPGM/EODBUNGA) +
       SRVPGM(APPPGM/SVCRATE APPPGM/SVCTGL) +
       UPDPROD(*NO) OPMSRC(*NO)

/* Setelah masuk sesi debug: DISPLAY MODULE SVCRATE */
```

33.7 Membaca dump program yang berakhir abnormal

Di dalam dump, bagian yang dicari berurutan:

1. **Message ID dan teks** di bagian atas
2. **Statement number** tempat kegagalan terjadi
3. **Call stack**, rantai pemanggilan sampai titik gagal

4. **Isi variabel**, khususnya field kunci dan indikator
5. **Isi record buffer** file terbuka, untuk tahu record mana yang sedang diproses

Message ID RPG yang paling sering muncul saat EOD:

Message ID	Arti	Penyebab lazim
`RNQ0100`	Pembagian dengan nol	Divisor dari tabel parameter kosong
`RNQ0103`	Hasil terlalu besar untuk field	Akumulasi melebihi definisi field
`RNQ0112`	Tanggal tidak valid	Data tanggal kosong atau salah format
`RNQ0202`	Pemanggilan gagal	Program atau service program tidak ditemukan
`RNQ0211`	Program tidak ditemukan	Library list salah
`RNQ0231`	Operasi pada file tidak terbuka	Urutan open/close keliru
`CPF4131`	Level check	Format record berubah tanpa kompilasi ulang
`CPF5029`	Kesalahan I/O data	Data korup atau tipe tidak cocok

`CPF4131` layak diperhatikan khusus. Ia muncul ketika struktur tabel diubah tetapi program pemakainya tidak dikompilasi ulang. Di lingkungan core banking dengan ratusan program, satu perubahan kolom bisa memicu puluhan `CPF4131` di tengah EOD.

33.8 Menelusuri masalah SQL di dalam program

Pesan optimizer yang paling penting dikenali di job log:

Pesan	Arti
`CPI4321`	Index dibangun sementara — tanda index permanen dibutuhkan
`CPI4322`	Table scan dilakukan — tidak ada index yang cocok
`CPI4327`	Index dipakai — kondisi yang diinginkan
`CPI432F`	Index disarankan oleh optimizer

Satu `CPI4322` pada tabel berisi puluhan juta record di tengah EOD sudah cukup untuk menjelaskan kenapa fase yang biasanya 20 menit tiba-tiba menjadi tiga jam.

🌙 Bab 34. Kinerja Batch dan EOD

34.1 Mengapa pool terpisah

Selama EOD, job batch menyapu data dalam jumlah besar. Bila berjalan di pool yang sama dengan sisa sesi interaktif, dua hal buruk terjadi sekaligus: batch mengusir halaman memori milik sesi interaktif, dan sesi interaktif yang tertinggal menahan record yang dibutuhkan batch.

```
/* Profil memori malam, dijalankan otomatis pukul 17:00 */
PGM
   CHGSHRPOOL POOL(*INTERACT) SIZE(2000000) ACTLVL(20)
   CHGSHRPOOL POOL(*SHRPOOL2) SIZE(24000000) ACTLVL(16)
   SNDMSG MSG('Profil memori malam diaktifkan') TOUSR(SYSENG)
ENDPGM

/* Profil memori siang, dikembalikan pukul 06:00 */
PGM
   CHGSHRPOOL POOL(*INTERACT) SIZE(16000000) ACTLVL(120)
   CHGSHRPOOL POOL(*SHRPOOL2) SIZE(8000000) ACTLVL(8)
   SNDMSG MSG('Profil memori siang diaktifkan') TOUSR(SYSENG)
ENDPGM
```

34.2 Paralelisasi job EOD

Tidak semua fase bisa diparalelkan.

Bisa diparalelkan	Tidak bisa
Perhitungan per rekening yang independen	Posting ke GL yang berbagi record kontrol
Pembuatan laporan yang berbeda	Perhitungan yang bergantung urutan
Klasifikasi per portofolio terpisah	Penomoran berurutan
Ekstraksi data ke MIS	Date roll

Pola pemecahan berdasarkan modulo nomor rekening:

```
PGM
   DCL VAR(&I) TYPE(*DEC) LEN(2 0) VALUE(0)

   /* Kirim 4 job paralel, masing-masing seperempat data */
   LOOP:
   CHGVAR VAR(&I) VALUE(&I + 1)

   SBMJOB CMD(CALL PGM(APPPGM/HITBUNGA) PARM(&I '4')) +
          JOB(BUNGA&I) JOBQ(QGPL/EODJOBQ) +
          JOBD(APPPGM/EODJOBD)

   IF COND(&I *LT 4) THEN(GOTO CMDLBL(LOOP))

   /* Tunggu keempatnya selesai sebelum fase berikutnya */
   CALL PGM(APPPGM/TUNGGUJOB) PARM('BUNGA' '4')
ENDPGM
```

Di dalam program, pembagian dilakukan lewat SQL:

```
SELECT NO_REK, SALDO, KD_PRODUK
  FROM DTALIB.REKENING
 WHERE STATUS = 'A'
   AND MOD(INT(SUBSTR(NO_REK, 13, 4)), :pTotalJob) = :pNomorJob - 1
  FOR UPDATE OF SALDO;
```

Peringatan: menambah jumlah job paralel tidak selalu mempercepat. Setelah titik tertentu, job saling berebut memori dan I/O, dan total durasi justru naik. Ukur, jangan menebak — mulai dari dua job, naikkan bertahap, catat durasi total tiap konfigurasi.

34.3 Reorganisasi tabel

Tabel dengan banyak record terhapus memperlambat pembacaan berurutan, dan pembacaan berurutan

justru pola akses dominan dalam batch.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
       NUMBER_ROWS,
       NUMBER_DELETED_ROWS,
       DECIMAL(NUMBER_DELETED_ROWS * 100.0 /
               NULLIF(NUMBER_ROWS + NUMBER_DELETED_ROWS, 0), 5, 1)
         AS persen_terhapus
  FROM QSYS2.SYSTABLESTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND NUMBER_DELETED_ROWS > 50000
 ORDER BY persen_terhapus DESC;
```

```
RGZPFM FILE(DTALIB/TRXHIST) +
       ALWCANCEL(*YES) +
       LOCK(*SHRUPD) +
       RBDACCPTH(*YES)
```

`ALWCANCEL(*YES)` memungkinkan reorganisasi dihentikan dan dilanjutkan dari titik terakhir. Ini penting: reorganisasi tabel besar bisa memakan berjam-jam, dan Anda perlu bisa menghentikannya saat batch window menipis.

🔒 Bab 35. Tata Kelola dan Audit Core Banking

35.1 Pemisahan tugas yang wajib ada

Gambar 46

35.2 Dual control untuk tindakan berisiko tinggi

Tindakan berikut menuntut dua orang, dengan bukti tertulis:

- Restore data produksi
- Rerun fase akrual
- Perubahan parameter tanggal buku
- Pemberian wewenang `*ALLOBJ`
- Perubahan konfigurasi audit
- Penghapusan journal receiver yang belum tersimpan
- Perubahan data langsung lewat SQL di produksi

35.3 Pertanyaan audit yang harus bisa Anda jawab

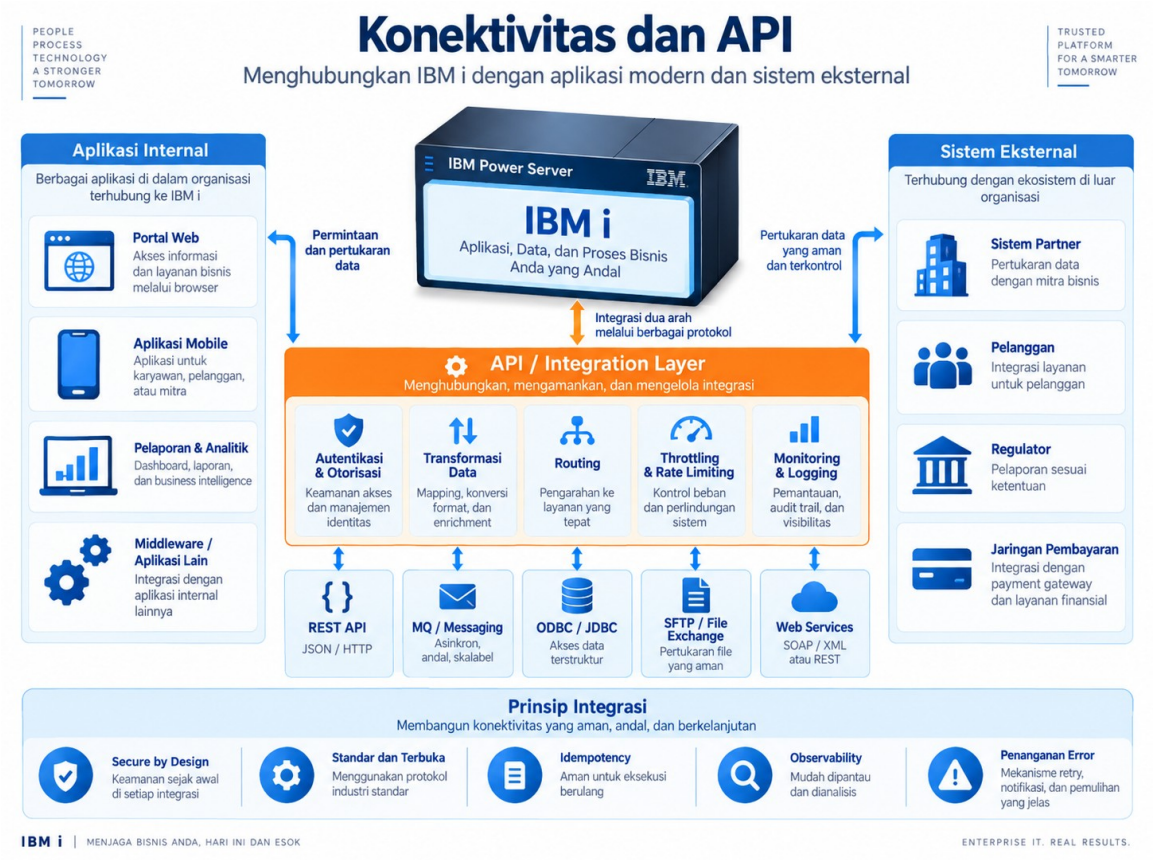
Pertanyaan auditor	Sumber jawaban
Siapa mengubah saldo rekening ini dan kapan	`DSPJRN` pada journal data
Siapa saja yang punya wewenang `*ALLOBJ`	`QSYS2.USER_INFO`
Apakah EOD selalu selesai tepat waktu 12 bulan terakhir	Tabel `EOD_KONTROL`
Bukti uji restore terakhir	Berita acara uji pemulihan

Pertanyaan auditor	Sumber jawaban
Perubahan apa saja ke produksi bulan lalu	Log object promotion + tiket change
Apakah ada akses SQL langsung ke tabel produksi	Exit point ODBC + audit journal
Bagaimana data nasabah dilindungi di UAT	Prosedur masking + bukti verifikasi

Siapkan kueri untuk setiap baris tabel ini sebagai skrip siap pakai. Audit yang berjalan lancar adalah audit yang jawabannya sudah disiapkan sebelum pertanyaannya diajukan.

🚀 Bagian VII — Integrasi dan Modernisasi

🚀 Bab 36. Konektivitas dan API



Gambar 47 — Konektivitas dan lapisan API

36.1 Pilihan jalur integrasi

Jalur	Cocok untuk	Catatan keamanan
ODBC/JDBC	Pelaporan, ETL ke data warehouse	Wajib exit program, batasi IP
REST lewat IWS	Kanal digital, integrasi modern	Wajib TLS dan autentikasi
MQ	Pertukaran andal antar sistem	Jaminan pengiriman
Data queue	Antar job di dalam IBM i	Cepat, tidak ada jaminan persistensi
Transfer file	Batch antar sistem	Enkripsi saat transit dan diam

36.2 Mengekspos program RPG sebagai REST API

Integrated Web Services memungkinkan program RPG yang sudah ada dipanggil lewat HTTP tanpa menulis ulang logikanya.

```
**free
ctl-opt nomain;

dcl-pr cekSaldo extpgm('CEKSALDO');
  pNoRek  char(16) const;
  pSaldo  packed(17:2);
  pStatus char(1);
end-pr;

dcl-proc cekSaldo export;
  dcl-pi *n;
    pNoRek  char(16) const;
    pSaldo  packed(17:2);
    pStatus char(1);
  end-pi;

  exec sql
    SELECT SALDO, STATUS
      INTO :pSaldo, :pStatus
      FROM DTALIB.REKENING
     WHERE NO_REK = :pNoRek;

  if sqlcode <> 0;
     pSaldo  = 0;
     pStatus = 'X';   // tidak ditemukan
  endif;
end-proc;
```

Program ini kemudian didaftarkan lewat konsol IWS, dan parameter dipetakan ke JSON secara otomatis. Yang perlu diperhatikan System Engineer:

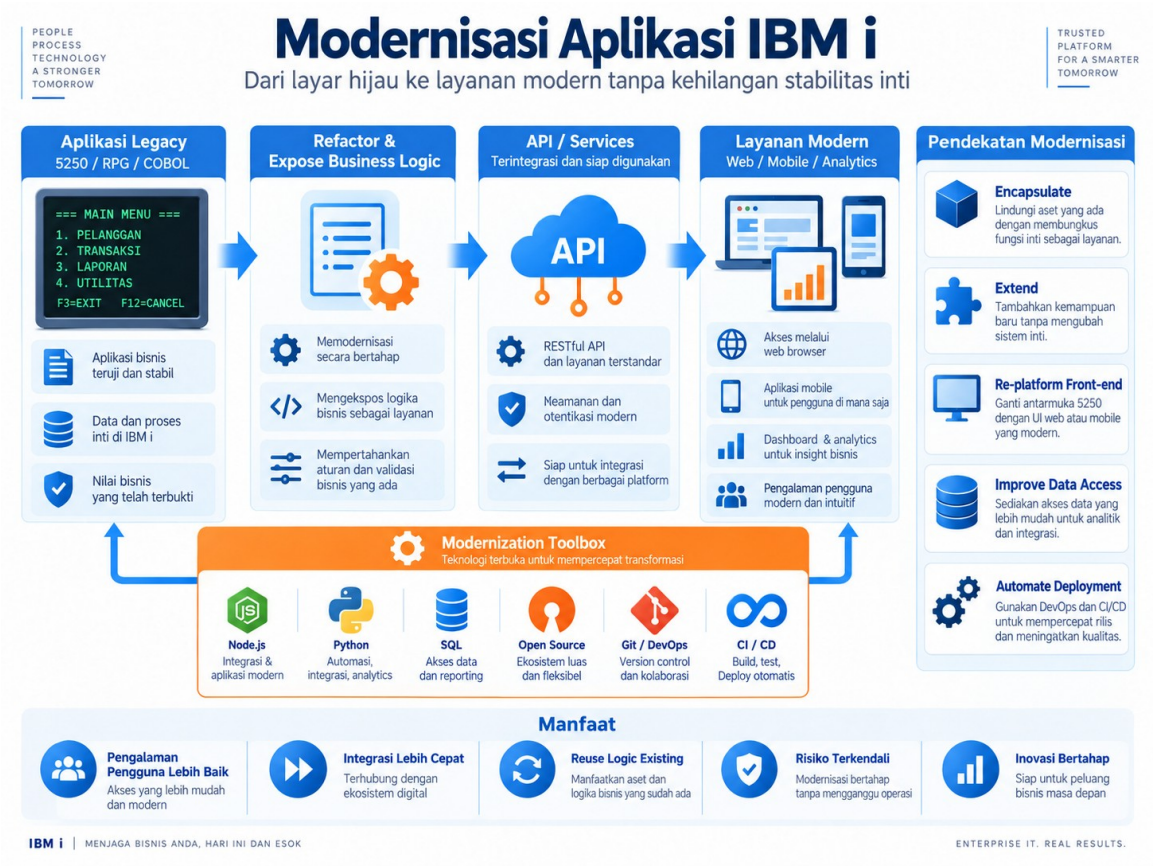
1. **Jalankan IWS di partisi terpisah bila memungkinkan**, atau minimal di subsystem terpisah agar lonjakan trafik API tidak menggerus pool transaksional.
2. **Batasi jumlah instans server.** Setiap instans memakan memori dan koneksi database.
3. **Pantau dengan cara yang sama seperti job lain** — server IWS tetap job biasa yang bisa MSGW dan menahan lock.

36.3 Memanggil layanan luar dari IBM i

```
-- HTTP GET langsung dari SQL
SELECT SYSTOOLS.HTTPGETCLOB(
         'https://api.contoh.test/kurs/hari-ini',
         '<httpHeader><header name="Accept" value="application/json"/></httpHeader>'
       ) AS respons
  FROM SYSIBM.SYSDUMMY1;
```

Praktis untuk integrasi sederhana. Untuk beban produksi yang tinggi, gunakan middleware, bukan pemanggilan langsung dari SQL.

🚀 Bab 37. Modernisasi Aplikasi



Gambar 48 — Jalur modernisasi aplikasi IBM i

37.1 Tangga modernisasi RPG

Gambar 49

Setiap anak tangga bisa dilakukan bertahap, per program, tanpa proyek besar sekali jalan. Inilah keunggulan modernisasi di IBM i dibanding penggantian platform total.

37.2 Perbandingan gaya kode

Fixed format lama:

```
     C     KEY            CHAIN     REKENING                  50
     C                    IF        *IN50 = *OFF
     C                    EVAL      WSALDO = SALDO
     C                    ENDIF
```

Free-format modern:

```
**free
chain KEY REKENING;
if %found(REKENING);
   wSaldo = SALDO;
endif;
```

Dengan SQL, yang lebih mudah dipelihara dan dioptimasi:

```
**free
exec sql
  SELECT SALDO INTO :wSaldo
    FROM DTALIB.REKENING
   WHERE NO_REK = :wKey;

if sqlcode = 0;
   // ditemukan
endif;
```

37.3 Open source di IBM i

```
# Dari PASE shell (QP2TERM atau SSH)
yum install nodejs20
yum install python3
yum install git

# Verifikasi
node --version
python3 --version
git --version
```

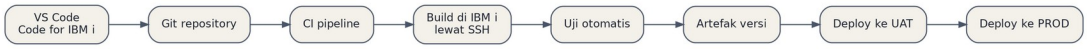
Contoh Node.js yang membaca Db2 for i:

```
const { odbc } = require('odbc');

async function cekSaldo(noRek) {
  const conn = await odbc.connect('DSN=*LOCAL');
  const hasil = await conn.query(
    'SELECT SALDO, STATUS FROM DTALIB.REKENING WHERE NO_REK = ?',
    [noRek]
  );
  await conn.close();
  return hasil[0] ?? null;
}
```

Catatan penting bagi System Engineer: **job PASE tidak terlihat sama dengan job IBM i biasa di `WRKACTJOB`.** Mereka berjalan di bawah `QP0ZSPWP`, dan pemantauannya butuh pendekatan berbeda. Tetapkan subsystem dan pool terpisah untuk beban open source.

🚀 Bab 38. DevOps di IBM i

38.1 Alur kerja modern



Gambar 50

38.2 Kode sumber di Git, bukan di source physical file

Perpindahan dari `QRPGLESRC` ke IFS dan Git adalah perubahan paling berdampak untuk tata kelola perubahan.

```
# Struktur repositori yang lazim
/src
  /rpgle
    hitbunga.rpgle
    ceksaldo.rpgle
  /sqlrpgle
    laporan.sqlrpgle
  /cl
    eodwrapper.clle
  /sql
    ddl_rekening.sql
/build
  build.sh
  compile.json
```

Skrip build sederhana:

```
#!/bin/bash
# Dijalankan lewat SSH dari CI runner
set -e

LIB=${1:-DEVPGM}
SRCDIR=/home/builder/src

for f in $SRCDIR/rpgle/*.rpgle; do
  nama=$(basename "$f" .rpgle | tr '[:lower:]' '[:upper:]')
  echo "Kompilasi $nama ke $LIB"
  system "CRTBNDRPG PGM($LIB/$nama) SRCSTMF('$f') \
          DBGVIEW(*SOURCE) OPTION(*EVENTF) REPLACE(*YES)"
done

echo "Build selesai"
```

38.3 Batasan yang perlu diterima

Harapan	Kenyataan di IBM i
Rollback instan seperti container	Perlu salinan objek lama, bukan image
Lingkungan sekali pakai	Partisi mahal, lingkungan bersifat tetap
Uji otomatis penuh	Banyak kode lawas tanpa uji, butuh investasi besar
Deploy berkali-kali sehari	Tidak realistis untuk core banking; mingguan sudah agresif

Jangan memaksakan praktik DevOps yang lahir dari dunia web ke sistem yang memindahkan uang. Ambil yang relevan: kendali versi, pipeline build, uji otomatis. Tinggalkan yang tidak: deploy kontinu ke produksi perbankan bukan kemajuan, melainkan risiko.

Bab 39. IBM i dan Hybrid Cloud

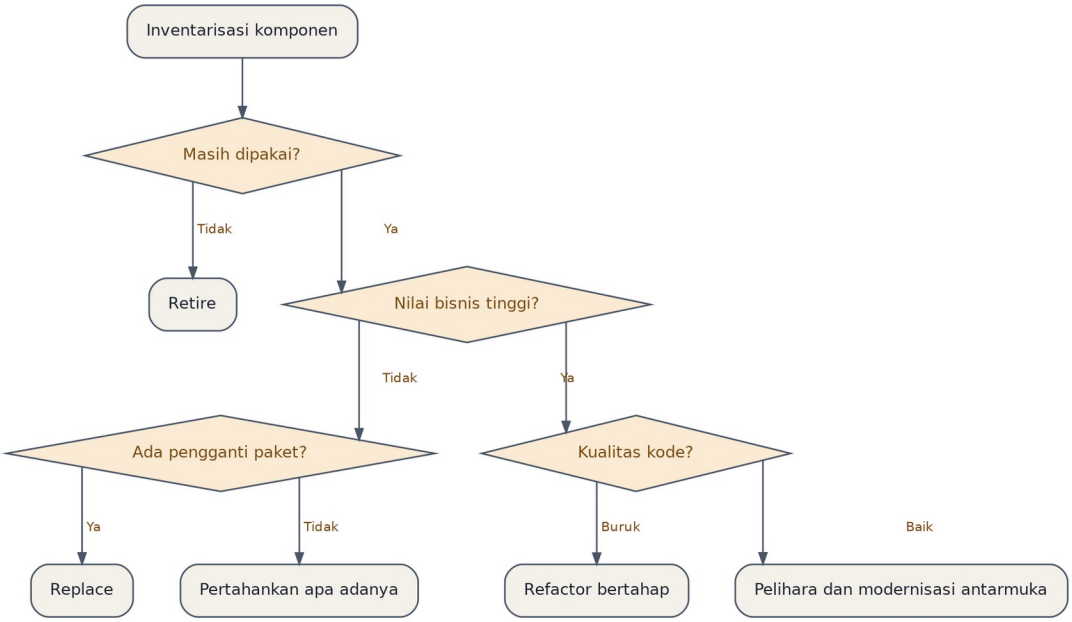
Gambar 51 — Arsitektur hybrid cloud IBM i

Opsi	Kendali	Kesesuaian untuk core banking
On-premise	Penuh	Paling lazim, sesuai ketentuan lokalisasi data
Power Virtual Server	Menengah	Cocok untuk DR dan lingkungan uji
Hosting Power pihak ketiga	Menengah	Bergantung kepatuhan penyedia
Migrasi ke platform lain	—	Proyek multi-tahun berisiko sangat tinggi

Pertimbangan regulatoris di Indonesia soal penempatan data dan pusat data harus diselesaikan sebelum diskusi teknis dimulai, bukan sesudahnya.

Bab 40. Manajemen Siklus Hidup Legacy

40.1 Empat pilihan untuk setiap komponen



Gambar 52

40.2 Menemukan komponen yang sudah mati

```
-- Program yang tidak pernah dipanggil dua tahun terakhir
SELECT OBJNAME, OBJTYPE, OBJOWNER,
       OBJCREATED, LAST_USED_TIMESTAMP,
       DAYS_USED_COUNT,
       OBJSIZE / 1024 / 1024 AS ukuran_mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('APPPGM','*PGM'))
 WHERE LAST_USED_TIMESTAMP < CURRENT DATE - 2 YEARS
    OR LAST_USED_TIMESTAMP IS NULL
 ORDER BY OBJSIZE DESC;
```

Jangan langsung menghapus. Prosedur yang aman: pindahkan ke library karantina selama enam bulan, pantau apakah ada yang mencarinya, baru arsipkan dan hapus.

```
/* Karantina, bukan hapus */
MOVOBJ OBJ(APPPGM/PGMLAMA) OBJTYPE(*PGM) TOLIB(KARANTINA)
```

40.3 Risiko yang sering diremehkan

1. **Dokumentasi tidak ada.** Satu-satunya spesifikasi adalah kodenya sendiri.
2. **Orang yang menulisnya sudah pensiun.** Tidak ada yang bisa menjelaskan alasan di balik keputusan desain.
3. **Perilaku tak terduga menjadi fitur.** Proses hilir diam-diam bergantung pada bug yang tidak pernah diperbaiki.
4. **Uji regresi tidak ada.** Tidak ada cara membuktikan perubahan tidak merusak apa pun.

Untuk sistem core banking, pendekatan yang paling masuk akal hampir selalu **modernisasi bertahap dari**

pinggiran ke tengah: mulai dari antarmuka dan pelaporan, lalu integrasi, dan biarkan mesin transaksi inti menjadi yang terakhir disentuh.

Bagian VIII — Tata Kelola dan Penutup

Bab 41. Manajemen Risiko dan Knowledge Transfer

41.1 Risiko yang khas platform ini

Risiko	Kemungkinan	Dampak	Mitigasi
Ketergantungan pada satu orang	Tinggi	Sangat tinggi	Dokumentasi, pasangan senior-junior
Perangkat keras melewati batas dukungan	Sedang	Tinggi	Roadmap refresh 5 tahun
Vendor aplikasi berhenti mendukung	Rendah	Sangat tinggi	Klausul escrow kode sumber
Kode lawas tanpa dokumentasi	Tinggi	Sedang	Inventarisasi bertahap
Kehilangan kemampuan restore	Rendah	Katastrofik	Uji restore terjadwal
Regenerasi SDM gagal	Tinggi	Tinggi	Program magang internal

41.2 Program transfer pengetahuan yang berhasil

1. **Dokumentasikan saat menyelesaikan masalah, bukan nanti.** Runbook ditulis pada hari insiden terjadi, ketika detailnya masih segar.
2. **Rekam sesi troubleshooting nyata.** Lima belas menit rekaman layar saat seseorang menelusuri EOD gagal lebih berharga daripada lima puluh halaman manual.
3. **Rotasi tugas terjadwal.** Setiap orang harus pernah menjadi penanggung jawab utama untuk setiap area kritikal, minimal sekali setahun.
4. **Uji dengan skenario, bukan kuis.** "EOD macet di fase akrual pukul 02:30, apa yang Anda lakukan?" mengungkap lebih banyak daripada pertanyaan definisi.
5. **Rekrut dari luar platform.** Pengalaman Linux, Oracle, atau Java bukan penghalang; konsep operasi bisa dipindahkan, dan sintaks bisa dipelajari.

🔍 Bab 42. Studi Kasus

42.1 EOD yang melambat perlahan

Gejala. Durasi EOD naik dari 3 jam menjadi 6 jam dalam enam bulan, tanpa penambahan volume transaksi yang berarti.

Diagnosis. Analisis tabel `EOD_KONTROL` menunjukkan kenaikan hanya terjadi di fase klasifikasi. Job log fase tersebut memunculkan `CPI4322` berulang pada satu tabel riwayat. Pemeriksaan `SYSTABLESTAT` menemukan tabel tersebut berisi 38% record terhapus.

Akar masalah. Proses pembersihan bulanan menghapus record lama tetapi tidak pernah diikuti reorganisasi. Ruang terhapus membuat pembacaan berurutan menyapu jauh lebih banyak halaman daripada yang diperlukan.

Perbaikan. Reorganisasi dengan `RGZPFM ALWCANCEL(*YES)` dipecah ke beberapa akhir pekan, ditambah index baru sesuai saran optimizer. Durasi kembali ke 3 jam 10 menit.

Pelajaran. Degradasi bertahap tidak memicu alarm apa pun. Hanya perbandingan tren yang bisa menangkapnya, dan itu menuntut pencatatan durasi tiap fase sejak awal.

42.2 Bunga yang tidak terbukukan

Gejala. Nasabah mengeluh bunga deposito tidak masuk. Jumlah yang terdampak sekitar 400 rekening, seluruhnya pada satu kode produk baru.

Diagnosis. EOD berjalan normal, tidak ada error, tidak ada entri di job log. Penelusuran journal menunjukkan field bunga memang ditulis dengan nilai nol.

Akar masalah. Produk baru dibuat di tabel produk, tetapi rate-nya belum didaftarkan di tabel parameter. Fungsi pengambil rate mengembalikan nol untuk kode yang tidak ditemukan, dan program membukukan nol itu tanpa protes.

Perbaikan. Program diubah agar membedakan rate nol yang sah dari rate tidak ditemukan, dan menghentikan fase dengan escape message bila menemui yang kedua. Ditambahkan pemeriksaan pra-EOD yang memvalidasi setiap kode produk aktif punya rate terdaftar.

Pelajaran. Kegagalan diam adalah yang paling mahal. Sistem yang berteriak saat ada yang salah jauh lebih baik daripada sistem yang terus berjalan dengan data yang salah.

42.3 Restore yang "berhasil" tetapi salah

Gejala. Setelah restore akibat korupsi tabel, EOD berjalan lancar. Keesokan harinya GL tidak seimbang.

Akar masalah. Restore mengembalikan tabel ke kondisi save, tetapi `APYJRNCHG` dijalankan sampai waktu yang keliru — satu jam sebelum cut-off, bukan pada cut-off. Transaksi satu jam terakhir hilang.

Perbaikan. Prosedur restore diubah agar selalu diakhiri verifikasi kuantitatif: jumlah record dan total saldo kontrol dibandingkan dengan laporan hari itu, sebelum sistem dinyatakan pulih.

Pelajaran. "Restore selesai tanpa error" bukan bukti pemulihan berhasil. Yang menjadi bukti adalah angka yang cocok.

🏗️ Bab 43. Tren dan Arah Platform

Tiga arah yang perlu diikuti System Engineer IBM i:

1. **Keamanan menjadi tema utama rilis.** IBM i 7.6 berfokus pada keamanan, dengan MFA native sebagai penanda paling jelas. Ekspektasi regulator akan terus naik searah.
2. **Batas perangkat keras makin ketat.** IBM i 7.6 menuntut Power10. Siklus refresh perangkat keras dan

siklus upgrade OS kini praktis menyatu.

3. **Integrasi menggantikan penggantian.** Tren yang berlaku bukan meninggalkan IBM i, melainkan membungkusnya dengan API dan menempatkannya sebagai mesin transaksi di belakang kanal modern.

Bagi karier seorang System Engineer, implikasinya jelas: nilai Anda tidak terletak pada hafalan perintah CL, melainkan pada kemampuan menjembatani sistem yang memindahkan uang dengan dunia yang menuntut API, audit, dan ketersediaan tanpa jeda.

📕 Bagian IX — Katalog Runbook

Bagian ini berisi runbook siap pakai untuk insiden yang paling sering terjadi. Setiap runbook ditulis agar bisa dijalankan pukul tiga pagi oleh orang yang belum pernah menangani masalah itu sebelumnya.

RB-001: Disk ASP melewati ambang kritis

Pemicu. Pesan `CPF0907` atau `CPI0948` di QSYSOPR, atau pemantauan melaporkan ASP di atas 85%.

Dampak. Job gagal menulis, EOD berhenti, dalam kasus ekstrem sistem berhenti melayani.

Diagnosis

```
WRKSYSSTS
WRKDSKSTS
```

```
-- Pemakaian per unit
SELECT UNIT_NUMBER, PERCENT_USED,
       UNIT_STORAGE_CAPACITY / 1073741824 AS kapasitas_gb
  FROM QSYS2.SYSDISKSTAT ORDER BY PERCENT_USED DESC;

-- Library terbesar
SELECT OBJLONGSCHEMA, SUM(OBJSIZE)/1048576 AS mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'))
 GROUP BY OBJLONGSCHEMA ORDER BY mb DESC FETCH FIRST 15 ROWS ONLY;

-- Journal receiver yang menumpuk
SELECT JOURNAL_LIBRARY, JOURNAL_NAME, RECEIVER_NAME,
       RECEIVER_SIZE/1048576 AS mb, ATTACHED_TIMESTAMP
  FROM QSYS2.JOURNAL_RECEIVER_INFO
 ORDER BY RECEIVER_SIZE DESC FETCH FIRST 20 ROWS ONLY;
```

Perbaikan, berurutan dari yang paling aman

1. Hapus spooled file yang melewati masa retensi
2. Bersihkan QTEMP dari job yang sudah mati dengan `RCLSTG` terjadwal
3. Simpan lalu hapus journal receiver lama yang sudah ter-backup
4. Hapus save file di library kerja yang sudah dipindah ke tape
5. Pindahkan library arsip ke ASP lain atau media eksternal

```
/* Hapus receiver yang sudah tersimpan, satu per satu, dengan verifikasi */
DSPJRNRCVA JRNRCV(JRNLIB/RCVDTA0087)
DLTJRNRCV JRNRCV(JRNLIB/RCVDTA0087) DLTOPT(*IGNINQMSG)
```

PERINGATAN. Jangan pernah menghapus journal receiver yang belum tersimpan di backup. Itu menghapus kemampuan pemulihan ke titik waktu.

Verifikasi. `WRKSYSSTS` menunjukkan pemakaian kembali di bawah 80%, dan tidak ada job baru yang gagal.

Eskalasi. Bila di atas 90% dan pembersihan tidak cukup: manajer TI segera, siapkan penambahan disk darurat.

RB-002: Job berstatus MSGW di tengah EOD

Pemicu. Pemantauan mendeteksi job EOD berstatus MSGW, atau progres EOD berhenti maju.

Dampak. Seluruh rantai EOD tertahan.

Diagnosis

```
SELECT JOB_NAME, SUBSYSTEM, FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_STATUS_FILTER => '*MSGW'));
```

```
DSPJOBLOG JOB(nomor/user/nama)
```

Cari pesan bertipe INQUIRY di bagian bawah job log. Baca **teks tingkat kedua** sebelum menjawab apa pun.

Perbaikan

Pesan	Jawaban yang benar	Alasan
`CPA5305` media penuh	`C` lalu ganti media	Jangan `I`, data hilang
`CPF4131` level check	`C` batalkan	Butuh kompilasi ulang, bukan diabaikan
`RNQ0100` bagi nol	`C` batalkan	Data parameter harus diperbaiki dulu
`CPA32B2` file tidak ditemukan	`C` batalkan	Periksa library list

PERINGATAN. Jangan menjawab `I` (ignore) pada pesan apa pun di fase akrual atau posting. Mengabaikan berarti melanjutkan dengan data yang salah.

Verifikasi. Job berubah status ke RUN, dan tabel `EOD_KONTROL` menunjukkan fase kembali maju.

Eskalasi. Bila tidak yakin jawaban yang benar: App Support, jangan menebak.

RB-003: Record lock menahan job EOD

Pemicu. Job EOD berstatus LCKW lebih dari lima menit.

Diagnosis

```
SELECT JOB_NAME, LOCK_STATE, TABLE_SCHEMA, TABLE_NAME, RRN
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'DTALIB';
```

```
WRKOBJLCK OBJ(DTALIB/REKENING) OBJTYPE(*FILE)
```

Identifikasi job penahan. Penyebab paling umum: sesi interaktif yang ditinggalkan pengguna dalam keadaan

mengedit record.

Perbaikan

1. Identifikasi pemilik job penahan
2. Bila sesi interaktif yang ditinggalkan, hubungi pemiliknya bila jam kerja
3. Bila di luar jam kerja atau tidak ada respons dalam 10 menit, akhiri job tersebut

```
/* Akhiri dengan kesempatan menutup rapi */
ENDJOB JOB(nomor/user/nama) OPTION(*CNTRLD) DELAY(30)

/* Bila tidak berhenti, paksa */
ENDJOB JOB(nomor/user/nama) OPTION(*IMMED)
```

PERINGATAN. `OPTION(*IMMED)` pada job yang sedang menulis dapat meninggalkan data setengah jadi. Coba `*CNTRLD` lebih dulu, selalu.

Pencegahan. Setel `QINACTITV` ke 15 menit dengan `QINACTMSGQ(*DSCJOB)`, dan akhiri subsystem QINTER sebelum EOD dimulai.

RB-004: EOD gagal di fase akrual

Pemicu. Tabel `EOD_KONTROL` menunjukkan status `E` pada fase akrual.

Dampak. Kritis. Data bunga berpotensi sebagian terbukukan.

JANGAN LAKUKAN. Jangan menjalankan ulang fase akrual. Jangan menjawab pesan apa pun dengan ignore. Jangan melanjutkan ke fase berikutnya.

Diagnosis

```
-- Sejauh mana akrual sempat berjalan
SELECT COUNT(*) AS sudah_terbukukan, SUM(LOG_BUNGA) AS total
  FROM DTALIB.BUNGALOG
 WHERE LOG_TGL = CURRENT DATE;

-- Bandingkan dengan jumlah rekening yang seharusnya
SELECT COUNT(*) AS seharusnya
  FROM DTALIB.REKENING WHERE STATUS = 'A';
```

```
SELECT MESSAGE_ID, MESSAGE_TEXT, MESSAGE_SECOND_LEVEL_TEXT,
       FROM_PROGRAM, FROM_PROCEDURE, FROM_STATEMENT
  FROM TABLE(QSYS2.JOBLOG_INFO('nomor/user/nama'))
 WHERE SEVERITY >= 30 ORDER BY ORDINAL_POSITION;
```

Prosedur

1. Hentikan seluruh job EOD yang masih berjalan
2. Catat jumlah record yang sudah terproses
3. Eskalasi ke manajer TI dan App Support — ini keputusan dua orang
4. Keputusan: restore dari Checkpoint A, lalu jalankan ulang EOD dari awal
5. Dokumentasikan seluruh langkah dengan stempel waktu

```
/* Restore, hanya setelah persetujuan dua orang tercatat */
RSTLIB SAVLIB(DTALIB) DEV(TAPMLB01) RSTLIB(DTALIB)
APYJRNCHG JRN(JRNLIB/JRNDTA) FILE((DTALIB/*ALL)) +
          RCVRNG(*LASTSAVE) TOTIME(tanggal 'jam-cutoff')
```

Verifikasi wajib sebelum melanjutkan

- Jumlah record tabel utama cocok dengan laporan sebelum EOD
- Total saldo kontrol cocok
- Tabel log bunga kosong untuk tanggal buku berjalan

Eskalasi. Manajer TI segera. Bila melewati pukul 05:00, manajemen bisnis untuk keputusan jam buka layanan.

RB-005: GL tidak seimbang setelah EOD

Pemicu. Fase GL balancing melaporkan selisih.

JANGAN LAKUKAN. Jangan menjalankan date roll. Jangan membuka layanan. Jangan mencoba memperbaiki dengan penyesuaian manual sebelum penyebabnya diketahui.

Diagnosis

```
-- Cari sisi mana yang timpang
SELECT KD_GL, SUM(DEBET) AS total_debet,
       SUM(KREDIT) AS total_kredit,
       SUM(DEBET) - SUM(KREDIT) AS selisih
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU = CURRENT DATE
 GROUP BY KD_GL
HAVING SUM(DEBET) <> SUM(KREDIT)
 ORDER BY ABS(SUM(DEBET) - SUM(KREDIT)) DESC;

-- Transaksi yang gagal berpasangan
SELECT NO_REF, COUNT(*) AS jml_leg, SUM(DEBET - KREDIT) AS net
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU = CURRENT DATE
 GROUP BY NO_REF
HAVING SUM(DEBET - KREDIT) <> 0;
```

Prosedur. Ini bukan masalah teknis semata. Libatkan Akuntansi sejak awal. System Engineer menyediakan data, Akuntansi memutuskan perlakuan.

Eskalasi. Manajer TI dan Kepala Akuntansi bersamaan, tanpa menunggu.

RB-006: Replikasi ke DR tertinggal

Pemicu. Pemantauan melaporkan lag replikasi di atas ambang.

Diagnosis

```
-- Selisih antara receiver terpasang dan yang sudah tereplikasi
SELECT JOURNAL_NAME, RECEIVER_NAME, ATTACHED_TIMESTAMP,
       RECEIVER_SIZE/1048576 AS mb
  FROM QSYS2.JOURNAL_RECEIVER_INFO
 WHERE STATUS = 'ATTACHED';
```

```
/* Untuk PowerHA */
WRKCLU
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA)
```

Penyebab yang paling sering

Gejala	Penyebab	Perbaikan
Lag naik hanya saat EOD	Bandwidth tidak memadai untuk lonjakan	Jadwalkan ulang atau tambah kapasitas
Lag naik terus, tidak turun	Proses apply di DR terhenti	Periksa job apply di sisi DR
Lag melonjak tiba-tiba	Gangguan jaringan	Periksa jalur komunikasi
Lag setelah reorganisasi	Volume perubahan besar	Normal, tunggu pemulihan

PERINGATAN. Jangan melakukan switchover saat lag tinggi. Anda akan kehilangan data sebesar lag tersebut.

⚡ RB-007: Kinerja menurun mendadak di jam layanan

Pemicu. Keluhan dari cabang atau kanal digital, respons lambat.

Diagnosis berurutan, jangan melompat

```
WRKSYSSTS       /* F10 reset, tunggu 2 menit, F5 refresh */
WRKACTJOB       /* F16 urut berdasarkan CPU */
WRKDSKSTS
```

```
-- Job dengan CPU tertinggi
SELECT JOB_NAME, AUTHORIZATION_NAME, SUBSYSTEM,
       ELAPSED_CPU_PERCENTAGE, ELAPSED_TOTAL_DISK_IO_COUNT, FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC FETCH FIRST 15 ROWS ONLY;
```

Pola penyebab yang khas

Temuan	Kemungkinan penyebab
Satu job menyedot CPU	Kueri ad hoc tanpa index, atau program looping
Faulting tinggi di pool interaktif	Batch bocor ke pool interaktif
Disk busy tinggi, CPU rendah	Table scan pada tabel besar
Banyak job LCKW	Kontensi penguncian
CPU tinggi merata	Beban memang melewati kapasitas

Perbaikan cepat untuk job liar

```
/* Turunkan prioritas dulu, jangan langsung diakhiri */
CHGJOB JOB(nomor/user/nama) RUNPTY(50)

/* Bila jelas kueri ad hoc dari pelaporan, baru akhiri */
ENDJOB JOB(nomor/user/nama) OPTION(*CNTRLD) DELAY(60)
```

Pencegahan. Batasi akses ODBC ke jam tertentu lewat exit program, dan arahkan pelaporan ke partisi MIS.

💾 RB-008: Backup semalam gagal

Pemicu. Pemeriksaan pagi menemukan backup tidak selesai.

Dampak. Tidak ada titik pulih untuk hari itu. Ini kondisi berisiko tinggi meskipun tidak terlihat mengganggu layanan.

Diagnosis

```
DSPLOG PERIOD((*AVAIL *CURRENT)) MSGID(CPF3700 CPF3800)
WRKMEDIBRM
DSPJOBLOG JOB(nomor/user/BACKUP)
```

Penyebab yang paling sering

Pesan	Penyebab	Perbaikan
`CPF3793`	Media tidak cukup	Sediakan media, jalankan ulang
`CPF3812`	Objek terkunci	Akhiri sesi penahan, ulangi
`CPA5305`	Media penuh di tengah jalan	Ganti media, ulangi dari awal
`CPF4257`	Perangkat tape tidak siap	Periksa perangkat, `VRYCFG`

Prosedur. Jalankan ulang backup segera, meskipun di jam layanan, memakai `SAVACT(*SYNCLIB)` agar tidak mengganggu operasi. Jangan menunda ke malam berikutnya — itu berarti dua hari tanpa titik pulih.

Eskalasi. Bila backup gagal dua malam berturut-turut, ini insiden tingkat tinggi. Laporkan ke manajer TI dan catat sebagai risiko terbuka.

RB-009: Level check CPF4131 massal

Pemicu. Banyak program gagal dengan `CPF4131` setelah perubahan struktur tabel.

Diagnosis

```
-- Program yang memakai tabel yang berubah
SELECT PROGRAM_NAME, PROGRAM_LIBRARY, OBJECT_NAME, OBJECT_TYPE
  FROM QSYS2.PROGRAM_INFO P,
  LATERAL (SELECT * FROM TABLE(QSYS2.BOUND_MODULE_INFO(
            P.PROGRAM_LIBRARY, P.PROGRAM_NAME)))
 WHERE P.PROGRAM_LIBRARY = 'APPPGM';
```

```
DSPPGMREF PGM(APPPGM/*ALL) OUTPUT(*OUTFILE) OUTFILE(QTEMP/PGMREF)
```

```
SELECT WHPNAM AS program, WHFNAM AS file_dipakai
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'REKENING';
```

Perbaikan. Kompilasi ulang seluruh program yang terdaftar. Bila perubahan struktur belum disetujui untuk produksi, kembalikan struktur tabel ke versi sebelumnya lewat prosedur rollback.

Pencegahan. Setiap perubahan DDL wajib disertai daftar program terdampak, dihasilkan dari `DSPPGMREF` di lingkungan UAT sebelum promosi.

RB-010: Profil pengguna terkunci massal

Pemicu. Banyak laporan pengguna tidak bisa login, `CPF1394`.

Diagnosis

```
SELECT AUTHORIZATION_NAME, STATUS, SIGN_ON_ATTEMPTS_NOT_VALID,
       PREVIOUS_SIGNON
  FROM QSYS2.USER_INFO
 WHERE STATUS = '*DISABLED'
 ORDER BY PREVIOUS_SIGNON DESC;
```

```
-- Percobaan login gagal dari audit journal
SELECT ENTRY_TIMESTAMP, USER_NAME, REMOTE_ADDRESS
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
        JOURNAL_ENTRY_TYPES => 'PW',
        STARTING_TIMESTAMP => CURRENT TIMESTAMP - 4 HOURS));
```

Bedakan dua situasi

Pola	Kemungkinan	Tindakan
Terkunci bertahap, tersebar	Kebijakan password baru, pengguna lupa	Reset terkendali
Terkunci serentak, satu sumber IP	Serangan brute force	Blokir IP, lapor Security
Terkunci setelah perubahan aplikasi	Akun layanan memakai password lama	Perbaiki konfigurasi

PERINGATAN. Jangan mengaktifkan kembali profil secara massal tanpa memeriksa pola. Bila ini serangan, Anda baru saja membuka kembali pintunya.

⚙️ RB-011: IPL tidak selesai atau sistem tidak naik

Pemicu. Setelah restart terjadwal atau kegagalan daya, sistem tidak mencapai kondisi operasional.

Diagnosis. Baca System Reference Code di panel HMC atau konsol.

SRC	Arti umum
`A600 5008`	IPL sedang berjalan, tunggu
`A900 2000`	Konsol tidak tersedia
`B900 3121`	Job QSTRUP gagal
`C600 xxxx`	Progres IPL normal
`A6xx 0277`	Menunggu klaim ruang penyimpanan

Prosedur

1. Catat SRC lengkap sebelum melakukan apa pun
2. Bila SRC menunjukkan progres (`C6xx`), tunggu — IPL setelah kegagalan tidak normal bisa memakan berjam-jam
3. Bila QSTRUP gagal, masuk lewat DST dan periksa program startup
4. Jangan melakukan IPL paksa berulang; itu memperpanjang proses klaim penyimpanan

Eskalasi. Bila SRC tidak dikenali atau tidak ada progres selama satu jam: buka kasus IBM dengan tingkat keparahan 1, sertakan SRC lengkap.

RB-012: Kehabisan job number atau job tabel penuh

Pemicu. `CPF1352` atau job baru gagal dimulai.

Diagnosis

```
SELECT COUNT(*) AS total_job FROM TABLE(QSYS2.ACTIVE_JOB_INFO());

SELECT SYSTEM_VALUE_NAME, CURRENT_NUMERIC_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME IN ('QMAXJOB','QACTJOB','QTOTJOB','QADLACTJ');
```

Perbaikan

```
/* Naikkan batas, berlaku tanpa IPL untuk sebagian nilai */
CHGSYSVAL SYSVAL(QMAXJOB) VALUE('200000')

/* Bersihkan job yang sudah selesai tapi masih menahan entri */
RCLSPLSTG DAYS(5)
```

Penyebab tersembunyi yang sering terlewat. Job yang berakhir tetapi spooled file-nya masih ada tetap menahan entri di job tabel. Kebijakan retensi spooled file yang longgar adalah penyebab paling umum.

RB-013: Akses ODBC mencurigakan terdeteksi

Pemicu. Exit program melaporkan pola akses tidak biasa, atau audit menemukan kueri ke tabel sensitif.

Diagnosis

```
SELECT REMOTE_ADDRESS, AUTHORIZATION_NAME, JOB_NAME,
       CLIENT_APPLICATION_NAME, CLIENT_PROGRAM_ID
  FROM QSYS2.ACTIVE_JOB_INFO(SUBSYSTEM_LIST_FILTER => 'QUSRWRK')
 WHERE JOB_NAME LIKE '%QZDASOINIT%';
```

Prosedur

1. Jangan memutus koneksi sebelum bukti terkumpul, kecuali ada eksfiltrasi aktif
2. Rekam alamat IP, profil, nama aplikasi klien, dan kueri yang dijalankan
3. Libatkan Security Officer sebelum mengambil tindakan
4. Bila ada indikasi pengambilan data massal, putuskan segera dan eskalasi sebagai insiden keamanan

Catatan. Tindakan Anda di sini menjadi bukti. Dokumentasikan setiap langkah dengan stempel waktu, dan jangan mengubah apa pun yang bisa dianggap merusak jejak.

RB-014: Tabel riwayat tumbuh tak terkendali

Pemicu. Pertumbuhan ASP melebihi proyeksi, atau kinerja batch menurun.

Diagnosis

```
SELECT TABLE_NAME, NUMBER_ROWS, NUMBER_DELETED_ROWS,
       DATA_SIZE/1048576 AS data_mb,
       INDEX_SIZE/1048576 AS index_mb
  FROM QSYS2.SYSTABLESTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
 ORDER BY DATA_SIZE DESC FETCH FIRST 20 ROWS ONLY;
```

Perbaikan bertahap

1. Pastikan kebijakan retensi tertulis dan disetujui, termasuk kewajiban retensi regulatoris
2. Arsipkan data lama ke tabel arsip atau media eksternal
3. Hapus dari tabel aktif
4. Reorganisasi tabel dengan `ALWCANCEL(*YES)`
5. Tinjau ulang index yang tidak terpakai

```
-- Index yang tidak pernah dipakai, kandidat penghapusan
SELECT TABLE_NAME, INDEX_NAME, LAST_QUERY_USE,
       NUMBER_KEY_COLUMNS, INDEX_SIZE/1048576 AS mb
  FROM QSYS2.SYSPARTITIONINDEXSTAT
 WHERE LAST_QUERY_USE IS NULL
   AND TABLE_SCHEMA = 'DTALIB'
 ORDER BY INDEX_SIZE DESC;
```

PERINGATAN. Index yang tidak dipakai kueri bisa jadi tetap dibutuhkan sebagai jalur akses program RPG native. Periksa `DSPPGMREF` sebelum menghapus.

💾 RB-015: Pemulihan setelah kehilangan partisi produksi

Pemicu. Partisi produksi tidak dapat diakses dan tidak dapat dipulihkan dalam RTO.

Dampak. Katastrofik. Ini deklarasi bencana.

Prosedur

1. Deklarasi bencana oleh pejabat berwenang, catat stempel waktu
2. Aktifkan tim tanggap bencana sesuai daftar kontak
3. Verifikasi status data di DR: seberapa jauh replikasi tertinggal
4. Konfirmasi keputusan failover dengan manajemen bisnis, karena ada potensi kehilangan data
5. Jalankan failover sesuai prosedur PowerHA atau replikasi
6. Verifikasi integritas: jumlah record, total saldo kontrol, transaksi terakhir
7. Arahkan seluruh kanal ke DR
8. Uji satu transaksi dari tiap kanal sebelum membuka layanan umum
9. Informasikan regulator sesuai ketentuan pelaporan insiden
10. Mulai dokumentasi insiden secara paralel, jangan menunggu selesai

Catatan penting. Prosedur ini hanya akan berjalan bila sudah pernah dilatih. Runbook yang belum pernah diuji dalam simulasi bukan runbook, melainkan harapan tertulis.

📖 Bagian X — Katalog IBM i Services

IBM i Services adalah kumpulan view dan fungsi tabel SQL yang menggantikan hampir seluruh perintah `DSP*` dan `WRK*` untuk keperluan otomasi. Bagian ini adalah referensi kueri siap pakai, dikelompokkan menurut tugas.

10.1 Kenapa beralih ke SQL

Perintah lama	Keterbatasan	Padanan SQL
`WRKACTJOB`	Hanya layar, sulit diotomasi	`QSYS2.ACTIVE_JOB_INFO`
`DSPUSRPRF`	Perlu outfile, format tetap	`QSYS2.USER_INFO`
`WRKSYSSTS`	Tidak bisa disimpan sebagai riwayat	`QSYS2.SYSTEM_STATUS_INFO`
`DSPJOBLOG`	Sulit dicari polanya	`QSYS2.JOBLOG_INFO`
`WRKOBJLCK`	Satu objek per kali	`QSYS2.RECORD_LOCK_INFO`
`DSPPTF`	Keluaran cetak	`QSYS2.PTF_INFO`

Keuntungan terbesar: hasilnya bisa disimpan ke tabel riwayat, dibandingkan antar waktu, dan dikirim ke sistem pemantauan luar.

📊 10.2 Status sistem dan kapasitas

```
-- Ringkasan status sistem, satu baris
SELECT TOTAL_JOBS_IN_SYSTEM,
       ACTIVE_JOBS_IN_SYSTEM,
       CURRENT_CPU_CAPACITY,
       AVERAGE_CPU_UTILIZATION,
       MAXIMUM_JOBS_IN_SYSTEM,
       SYSTEM_ASP_USED,
       TOTAL_AUXILIARY_STORAGE / 1024 AS total_gb,
       CURRENT_TEMPORARY_STORAGE,
       MAXIMUM_TEMPORARY_STORAGE_USED
  FROM TABLE(QSYS2.SYSTEM_STATUS(RESET_STATISTICS => 'NO'));

-- Pemakaian memori per pool
SELECT POOL_NAME, POOL_ID,
       CURRENT_SIZE / 1024 AS size_mb,
       DEFINED_SIZE / 1024 AS defined_mb,
       MAXIMUM_ACTIVE_THREADS,
       DATABASE_FAULTS,
       NON_DATABASE_FAULTS,
       ACTIVE_TO_WAIT, WAIT_TO_INELIGIBLE
  FROM TABLE(QSYS2.MEMORY_POOL(RESET_STATISTICS => 'NO'));

-- Riwayat kapasitas, jalankan terjadwal tiap jam
INSERT INTO APPWRK.KAPASITAS_LOG
SELECT CURRENT TIMESTAMP,
       AVERAGE_CPU_UTILIZATION,
       SYSTEM_ASP_USED,
       ACTIVE_JOBS_IN_SYSTEM,
       CURRENT_TEMPORARY_STORAGE
  FROM TABLE(QSYS2.SYSTEM_STATUS());
```

10.3 Job dan work management

```
-- Job aktif dengan detail lengkap
SELECT JOB_NAME, AUTHORIZATION_NAME, JOB_TYPE, SUBSYSTEM,
       JOB_STATUS, FUNCTION, ELAPSED_CPU_PERCENTAGE,
       ELAPSED_TOTAL_DISK_IO_COUNT, TEMPORARY_STORAGE,
       RUN_PRIORITY, THREAD_COUNT
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 WHERE JOB_TYPE <> 'SYS'
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC;
```

```
-- Job di setiap job queue, termasuk yang menunggu
SELECT JOB_QUEUE_LIBRARY, JOB_QUEUE_NAME, JOB_QUEUE_STATUS,
       NUMBER_OF_JOBS, SUBSYSTEM_NAME, MAXIMUM_ACTIVE_JOBS
  FROM QSYS2.JOB_QUEUE_INFO
 WHERE NUMBER_OF_JOBS > 0;

-- Job yang menunggu di job queue tertentu
SELECT JOB_NAME, JOB_QUEUE_NAME, JOB_QUEUE_STATUS,
       JOB_ENTERED_SYSTEM_TIME, SUBMITTER_JOB_NAME
  FROM TABLE(QSYS2.JOB_INFO(JOB_STATUS_FILTER => '*JOBQ',
                            JOB_USER_FILTER => '*ALL'))
 ORDER BY JOB_ENTERED_SYSTEM_TIME;

-- Subsystem yang aktif dan jumlah job di dalamnya
SELECT SUBSYSTEM_DESCRIPTION_LIBRARY, SUBSYSTEM_DESCRIPTION,
       STATUS, CURRENT_ACTIVE_JOBS, MAXIMUM_ACTIVE_JOBS
  FROM QSYS2.SUBSYSTEM_INFO
 WHERE STATUS = 'ACTIVE';

-- Job yang memakai temporary storage paling besar
SELECT JOB_NAME, AUTHORIZATION_NAME,
       TEMPORARY_STORAGE AS mb,
       FUNCTION, JOB_STATUS
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 WHERE TEMPORARY_STORAGE > 1000
 ORDER BY TEMPORARY_STORAGE DESC;
```

10.4 Penyimpanan dan objek

```
-- Statistik disk per unit
SELECT ASP_NUMBER, UNIT_NUMBER, UNIT_TYPE,
       UNIT_STORAGE_CAPACITY / 1073741824 AS kapasitas_gb,
       UNIT_SPACE_AVAILABLE / 1073741824 AS tersedia_gb,
       PERCENT_USED, UNIT_MEDIA_TYPE, HARDWARE_STATUS
  FROM QSYS2.SYSDISKSTAT
 ORDER BY ASP_NUMBER, PERCENT_USED DESC;

-- Objek terbesar dalam satu library
SELECT OBJNAME, OBJTYPE, OBJOWNER,
       OBJSIZE / 1048576 AS ukuran_mb,
       OBJCREATED, LAST_USED_TIMESTAMP, DAYS_USED_COUNT,
       OBJTEXT
  FROM TABLE(QSYS2.OBJECT_STATISTICS('APPLIB','*ALL'))
 ORDER BY OBJSIZE DESC FETCH FIRST 30 ROWS ONLY;

-- Ringkasan ukuran per library
SELECT OBJLONGSCHEMA AS library,
       COUNT(*) AS jumlah_objek,
       SUM(OBJSIZE) / 1048576 AS total_mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'))
 GROUP BY OBJLONGSCHEMA
 ORDER BY total_mb DESC;

-- Objek yang tidak pernah dipakai
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJCREATED,
       OBJSIZE / 1048576 AS mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('APPLIB','*ALL'))
 WHERE DAYS_USED_COUNT = 0
 ORDER BY OBJSIZE DESC;
```

```
-- Isi direktori IFS
SELECT PATH_NAME, OBJECT_TYPE,
       DATA_SIZE / 1048576 AS mb,
       OBJECT_OWNER, LAST_USED_TIMESTAMP
  FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
         START_PATH_NAME => '/interface',
         SUBTREE_DIRECTORIES => 'YES'))
 ORDER BY DATA_SIZE DESC;
```

🗄️ 10.5 Basis data

```
-- Statistik tabel
SELECT TABLE_SCHEMA, TABLE_NAME, TABLE_PARTITION,
       NUMBER_ROWS, NUMBER_DELETED_ROWS,
       DATA_SIZE / 1048576 AS data_mb,
       VARIABLE_LENGTH_SIZE / 1048576 AS varlen_mb,
       MAINTAINED_TEMPORARY_INDEX_SIZE / 1048576 AS idx_mb,
       LAST_CHANGE_TIMESTAMP
  FROM QSYS2.SYSTABLESTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
 ORDER BY DATA_SIZE DESC;

-- Index yang disarankan optimizer
SELECT TABLE_SCHEMA, TABLE_NAME, KEY_COLUMNS_ADVISED,
       LEADING_COLUMN_KEYS, INDEX_TYPE,
       TIMES_ADVISED, ESTIMATED_CREATION_TIME,
       LAST_ADVISED, MTI_USED, MTI_CREATED
  FROM QSYS2.SYSIXADV
 WHERE TABLE_SCHEMA = 'DTALIB'
 ORDER BY TIMES_ADVISED DESC;

-- Index yang ada dan seberapa sering dipakai
SELECT TABLE_NAME, INDEX_NAME,
       NUMBER_KEY_COLUMNS,
       INDEX_SIZE / 1048576 AS mb,
       LAST_QUERY_USE, LAST_STATISTICS_USE,
       QUERY_USE_COUNT, QUERY_STATISTICS_COUNT
  FROM QSYS2.SYSPARTITIONINDEXSTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
 ORDER BY QUERY_USE_COUNT;

-- Penguncian record aktif
SELECT JOB_NAME, LOCK_STATE, LOCK_STATUS, LOCK_SCOPE,
       TABLE_SCHEMA, TABLE_NAME, RRN
  FROM QSYS2.RECORD_LOCK_INFO
 ORDER BY TABLE_SCHEMA, TABLE_NAME;

-- Penguncian objek
SELECT OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE,
       JOB_NAME, LOCK_STATE, LOCK_SCOPE, LOCK_STATUS
  FROM QSYS2.OBJECT_LOCK_INFO
 WHERE OBJECT_SCHEMA = 'DTALIB';

-- Kueri SQL yang sedang berjalan
SELECT JOB_NAME, AUTHORIZATION_NAME,
       SQL_STATEMENT_TEXT, STATEMENT_START_TIMESTAMP,
       INTERFACE_TYPE, CLIENT_APPLICATION_NAME
  FROM QSYS2.ACTIVE_QUERY_INFO;
```

🔒 10.6 Keamanan

```
-- Profil dengan wewenang khusus
SELECT AUTHORIZATION_NAME, STATUS, USER_CLASS_NAME,
       SPECIAL_AUTHORITIES, GROUP_PROFILE_NAME,
       LIMIT_CAPABILITIES, INITIAL_MENU, INITIAL_PROGRAM,
       PREVIOUS_SIGNON, SIGN_ON_ATTEMPTS_NOT_VALID,
       PASSWORD_EXPIRATION_INTERVAL, DAYS_UNTIL_PASSWORD_EXPIRES
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES <> ''
 ORDER BY AUTHORIZATION_NAME;

-- Profil tidak aktif lebih dari 90 hari
SELECT AUTHORIZATION_NAME, STATUS, PREVIOUS_SIGNON,
       DAYS(CURRENT DATE) - DAYS(DATE(PREVIOUS_SIGNON)) AS hari_diam
  FROM QSYS2.USER_INFO
 WHERE STATUS = '*ENABLED'
   AND PREVIOUS_SIGNON < CURRENT TIMESTAMP - 90 DAYS
 ORDER BY PREVIOUS_SIGNON;

-- Wewenang objek yang terlalu terbuka
SELECT SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME, OBJECT_TYPE,
       AUTHORIZATION_NAME, OBJECT_AUTHORITY,
       DATA_READ, DATA_UPDATE, DATA_DELETE
  FROM QSYS2.OBJECT_PRIVILEGES
 WHERE SYSTEM_OBJECT_SCHEMA = 'DTALIB'
   AND AUTHORIZATION_NAME = '*PUBLIC'
   AND OBJECT_AUTHORITY <> '*EXCLUDE';

-- Exit program yang terpasang
SELECT EXIT_POINT_NAME, EXIT_PROGRAM_NUMBER,
       EXIT_PROGRAM_LIBRARY, EXIT_PROGRAM,
       EXIT_POINT_FORMAT
  FROM QSYS2.EXIT_PROGRAM_INFO
 ORDER BY EXIT_POINT_NAME;

-- Fungsi khusus dan siapa yang berwenang
SELECT FUNCTION_ID, USAGE, USER_NAME
  FROM QSYS2.FUNCTION_USAGE
 ORDER BY FUNCTION_ID;

-- Sertifikat yang mendekati kedaluwarsa
SELECT CERTIFICATE_LABEL, VALIDITY_START, VALIDITY_END,
       DAYS(VALIDITY_END) - DAYS(CURRENT DATE) AS sisa_hari,
       ISSUER_COMMON_NAME
  FROM QSYS2.CERTIFICATE_INFO(
         CERTIFICATE_STORE_PASSWORD => '*NOPWD')
 WHERE VALIDITY_END < CURRENT DATE + 90 DAYS;
```

🔒 10.7 Audit journal

```
-- Kegagalan wewenang 24 jam terakhir
SELECT ENTRY_TIMESTAMP, USER_NAME, JOB_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE,
       VIOLATION_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'AF',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 24 HOURS))
 ORDER BY ENTRY_TIMESTAMP DESC;
```

```
-- Perubahan profil pengguna
SELECT ENTRY_TIMESTAMP, USER_NAME AS pelaku,
       ENTRY_TYPE, OBJECT_NAME AS profil_diubah
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'CP',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS));

-- Objek yang dihapus dari produksi
SELECT ENTRY_TIMESTAMP, USER_NAME, JOB_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'DO',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 7 DAYS))
 WHERE OBJECT_LIBRARY IN ('APPPGM','DTALIB','APPPRM');

-- Restore objek, sering luput dari perhatian
SELECT ENTRY_TIMESTAMP, USER_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'RO,RU,RZ',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS));
```

🌐 10.8 Jaringan dan komunikasi

```
-- Koneksi jaringan aktif
SELECT REMOTE_ADDRESS, REMOTE_PORT, LOCAL_ADDRESS, LOCAL_PORT,
       PROTOCOL, TCP_STATE, IDLE_TIME,
       AUTHORIZATION_NAME, JOB_NAME
  FROM QSYS2.NETSTAT_JOB_INFO
 ORDER BY REMOTE_ADDRESS;

-- Port yang sedang mendengarkan
SELECT LOCAL_PORT, LOCAL_PORT_NAME, PROTOCOL,
       TCP_STATE, JOB_NAME
  FROM QSYS2.NETSTAT_INFO
 WHERE TCP_STATE = 'LISTEN'
 ORDER BY LOCAL_PORT;

-- Interface TCP/IP
SELECT INTERNET_ADDRESS, SUBNET_MASK, LINE_DESCRIPTION,
       INTERFACE_STATUS, ALIAS_NAME
  FROM QSYS2.NETSTAT_INTERFACE_INFO;

-- Server TCP dan statusnya
SELECT SERVER_NAME, SERVER_STATUS, AUTOSTART,
       ACTIVE_JOBS
  FROM QSYS2.SERVER_SHARE_INFO;
```

10.9 PTF dan pemeliharaan

```
-- PTF group dan levelnya
SELECT PTF_GROUP_NAME, PTF_GROUP_DESCRIPTION,
       PTF_GROUP_LEVEL, PTF_GROUP_TARGET_RELEASE,
       PTF_GROUP_STATUS
  FROM QSYS2.GROUP_PTF_INFO
 ORDER BY PTF_GROUP_NAME;

-- PTF yang menunggu IPL
SELECT PTF_IDENTIFIER, PTF_PRODUCT_ID, PTF_RELEASE_LEVEL,
       PTF_LOADED_STATUS, PTF_SAVE_FILE, PTF_IPL_ACTION,
       PTF_ACTION_PENDING, PTF_ACTION_REQUIRED
  FROM QSYS2.PTF_INFO
 WHERE PTF_ACTION_PENDING = 'YES'
    OR PTF_IPL_ACTION <> 'NONE';

-- Licensed program yang terpasang
SELECT PRODUCT_ID, PRODUCT_OPTION, RELEASE_LEVEL,
       LOAD_ID, SYMBOLIC_LOAD_STATE, LOAD_ERROR_INDICATOR
  FROM QSYS2.SOFTWARE_PRODUCT_INFO
 WHERE LOAD_ERROR_INDICATOR = 'YES'
    OR SYMBOLIC_LOAD_STATE <> 'INSTALLED';
```

10.10 Journaling

```
-- Journal dan receiver terpasang
SELECT JOURNAL_LIBRARY, JOURNAL_NAME,
       ATTACHED_JOURNAL_RECEIVER_LIBRARY,
       ATTACHED_JOURNAL_RECEIVER_NAME,
       JOURNAL_STATE, JOURNAL_TYPE,
       TOTAL_SIZE_JOURNAL_RECEIVERS / 1048576 AS total_mb
  FROM QSYS2.JOURNAL_INFO;

-- File yang sedang dijurnal
SELECT JOURNALED_OBJECT_LIBRARY, JOURNALED_OBJECT,
       OBJECT_TYPE, JOURNAL_IMAGES, OMIT_JOURNAL_ENTRY
  FROM QSYS2.JOURNALED_OBJECTS
 WHERE JOURNAL_NAME = 'JRNDTA';

-- File yang SEHARUSNYA dijurnal tetapi tidak
SELECT OBJNAME, OBJTYPE, OBJSIZE / 1048576 AS mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('DTALIB','*FILE')) o
 WHERE NOT EXISTS (
       SELECT 1 FROM QSYS2.JOURNALED_OBJECTS j
        WHERE j.JOURNALED_OBJECT = o.OBJNAME
          AND j.JOURNALED_OBJECT_LIBRARY = 'DTALIB');
```

Kueri terakhir itu layak dijadwalkan mingguan. Tabel produksi yang lolos dari journaling adalah lubang pemulihan yang tidak terlihat sampai dibutuhkan.

📊 10.11 Membangun pemantauan sendiri

Gabungkan kueri-kueri di atas menjadi satu program yang dijalankan terjadwal.

```
-- Tabel penampung
CREATE TABLE APPWRK.MONITOR_LOG (
    WAKTU        TIMESTAMP NOT NULL,
    METRIK       VARCHAR(40) NOT NULL,
    NILAI        DECIMAL(15,2),
    STATUS       CHAR(1),
    KETERANGAN   VARCHAR(200),
    PRIMARY KEY (WAKTU, METRIK)
);
```

```
PGM
   /* CPU */
   RUNSQL SQL('INSERT INTO APPWRK.MONITOR_LOG +
      SELECT CURRENT TIMESTAMP, ''CPU_PERSEN'', +
             AVERAGE_CPU_UTILIZATION, +
             CASE WHEN AVERAGE_CPU_UTILIZATION > 85 THEN ''K'' +
                  WHEN AVERAGE_CPU_UTILIZATION > 70 THEN ''W'' +
                  ELSE ''N'' END, '''' +
        FROM TABLE(QSYS2.SYSTEM_STATUS())') COMMIT(*NONE)

   /* ASP */
   RUNSQL SQL('INSERT INTO APPWRK.MONITOR_LOG +
      SELECT CURRENT TIMESTAMP, ''ASP_PERSEN'', SYSTEM_ASP_USED, +
             CASE WHEN SYSTEM_ASP_USED > 85 THEN ''K'' +
                  WHEN SYSTEM_ASP_USED > 75 THEN ''W'' +
                  ELSE ''N'' END, '''' +
        FROM TABLE(QSYS2.SYSTEM_STATUS())') COMMIT(*NONE)

   /* Job MSGW */
   RUNSQL SQL('INSERT INTO APPWRK.MONITOR_LOG +
      SELECT CURRENT TIMESTAMP, ''JOB_MSGW'', COUNT(*), +
             CASE WHEN COUNT(*) > 2 THEN ''K'' +
                  WHEN COUNT(*) > 0 THEN ''W'' +
                  ELSE ''N'' END, '''' +
        FROM TABLE(QSYS2.ACTIVE_JOB_INFO( +
             JOB_STATUS_FILTER => ''*MSGW''))') COMMIT(*NONE)

   /* Kirim peringatan bila ada status kritis */
   CALL PGM(APPPGM/KIRIMALERT)
ENDPGM
```

Jadwalkan tiap lima menit dengan `ADDJOBSCDE`. Setelah beberapa minggu Anda punya riwayat yang cukup untuk melihat tren, dan tren itulah yang memungkinkan Anda menangkap masalah sebelum menjadi insiden.

💻 Bagian XI — Primer RPG ILE untuk System Engineer

Anda tidak perlu menjadi programmer RPG. Tetapi Anda perlu bisa **membaca** RPG, karena setiap insiden aplikasi berakhir di depan kode sumber, dan seorang System Engineer yang hanya bisa mengangkat bahu di titik itu akan selalu menjadi penonton.

💻 Bab 44. Anatomi Program RPG Modern

44.1 Kerangka dasar

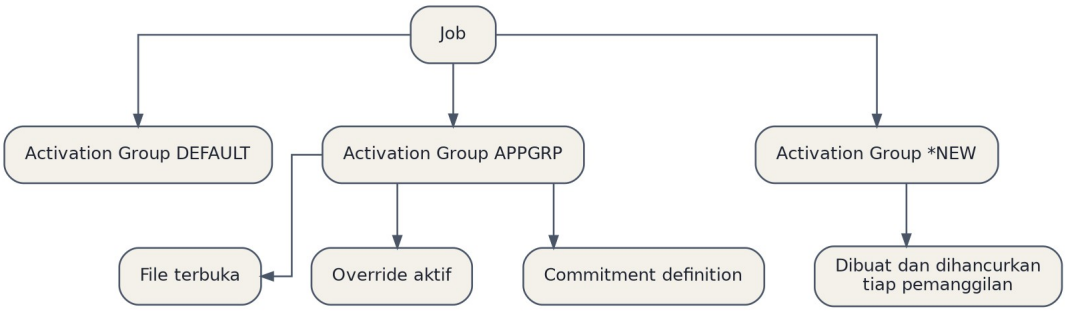
```
**free
```

```
// ---------- Control specification ----------
ctl-opt dftactgrp(*no)
        actgrp('APPGRP')
        option(*srcstmt: *nodebugio: *noshowcpy)
        bnddir('APPBND')
        datfmt(*iso)
        decedit('0,')
        main(prosesUtama);

// ---------- Deklarasi file ----------
dcl-f REKENING usage(*update: *delete) keyed;
dcl-f AUDITLOG usage(*output);

// ---------- Variabel global ----------
dcl-s gTglBuku date(*iso);
dcl-s gJmlProses int(10) inz(0);

// ---------- Konstanta ----------
dcl-c AKTIF   'A';
dcl-c TUTUP   'C';
dcl-c BLOKIR  'B';

// ---------- Prototipe ----------
dcl-pr ambilRate packed(9:6);
  pKdProduk char(4) const;
end-pr;

// ---------- Prosedur utama ----------
dcl-proc prosesUtama;
  dcl-pi *n end-pi;

  gTglBuku = %date();
  // logika di sini

end-proc;
```

44.2 Arti opsi yang paling penting

Opsi	Arti	Kenapa penting bagi Anda
`dftactgrp(*no)`	Bukan default activation group	Wajib untuk program ILE
`actgrp('NAMA')`	Nama activation group	Menentukan isolasi sumber daya dan scope commit
`option(*srcstmt)`	Nomor baris = nomor sumber	Job log menunjuk baris asli, bukan nomor internal
`option(*nodebugio)`	Breakpoint tidak berhenti tiap I/O	Debug jadi jauh lebih praktis
`bnddir('NAMA')`	Binding directory	Tempat mencari service program
`datfmt(*iso)`	Format tanggal	Menghindari kekacauan dd/mm vs mm/dd

Activation group adalah konsep yang wajib Anda pahami sebagai System Engineer. Ia menentukan di mana sumber daya program hidup: file yang terbuka, override, dan yang paling penting, **cakupan commitment control**. Dua program di activation group berbeda melakukan commit secara terpisah.



Gambar 53

`ACTGRP(*NEW)` membuat activation group baru tiap kali program dipanggil, lalu menghancurkannya saat selesai. Ini aman tetapi mahal — di dalam perulangan yang memanggil program jutaan kali, ini penyebab kinerja buruk yang sangat sering ditemui.

44.3 Deklarasi variabel

```
**free
// Skalar
dcl-s wNama      varchar(60);
dcl-s wKode      char(4);
dcl-s wSaldo     packed(17:2);
dcl-s wRate      packed(9:6);
dcl-s wHitung    int(10);
dcl-s wBesar     int(20);
dcl-s wTanggal   date(*iso);
dcl-s wWaktu     time;
dcl-s wStempel   timestamp;
dcl-s wBenar     ind;
dcl-s wPecahan   zoned(7:2);

// Nilai awal
dcl-s wCounter   int(10) inz(0);
dcl-s wStatus    char(1) inz('A');
dcl-s wHariIni   date inz(*sys);

// Array
dcl-s wBulan     char(20) dim(12);
dcl-s wTotal     packed(15:2) dim(12) inz(*zeros);

// Konstanta
dcl-c MAKS_COBA  3;
dcl-c PESAN_GAGAL 'Proses dihentikan karena kesalahan data';
```

44.4 Data structure

Data structure adalah cara RPG memetakan satu blok memori ke beberapa field. Ini penting saat memanggil API sistem.

```
**free
// Data structure sederhana
dcl-ds dsRekening qualified;
  noRek    char(16);
  nama     varchar(60);
  saldo    packed(17:2);
```

```
  produk   char(4);
  status   char(1);
end-ds;

// Penggunaan: dsRekening.noRek = '1234567890123456';

// Data structure dari format record file
dcl-ds dsRek likerec(REKENINGR: *all);

// Array of data structure
dcl-ds dsBatch qualified dim(500);
  noRek    char(16);
  jumlah   packed(15:2);
end-ds;

// Data structure dengan overlay, untuk mengurai field gabungan
dcl-ds dsTanggal qualified;
  penuh    char(8);
  tahun    char(4) overlay(penuh: 1);
  bulan    char(2) overlay(penuh: 5);
  hari     char(2) overlay(penuh: 7);
end-ds;

// Data structure untuk menerima informasi program status
dcl-ds dsPSDS psds qualified;
  namaProc   char(10) pos(1);
  statusKode zoned(5)  pos(11);
  namaRutin  char(8)  pos(22);
  nomorBaris char(8)  pos(21);
  namaPgm    char(10) pos(334);
  namaJob    char(10) pos(244);
  namaUser   char(10) pos(254);
  nomorJob   char(6)  pos(264);
end-ds;
```

PSDS (Program Status Data Structure) adalah cara program mengetahui konteksnya sendiri. Sangat berguna untuk menulis pesan error yang menyebutkan nama job dan baris kode.

44.5 Operasi file native

```
**free
dcl-f REKENING usage(*update: *delete) keyed;

// CHAIN: baca satu record berdasarkan kunci
chain wNoRek REKENING;
if %found(REKENING);
   // ditemukan, record terkunci untuk update
   SALDO = SALDO + wJumlah;
   update REKENINGR;
else;
   // tidak ditemukan
endif;

// SETLL + READE: baca berurutan dari posisi tertentu
setll wKdCabang REKENING;
reade wKdCabang REKENING;
dow not %eof(REKENING);
   // proses record
   reade wKdCabang REKENING;
enddo;
```

```
// READ: baca berurutan seluruh file
read REKENING;
dow not %eof(REKENING);
   // proses
   read REKENING;
enddo;

// Menulis record baru
clear REKENINGR;
NO_REK = wNoRek;
SALDO  = 0;
write REKENINGR;

// Menghapus
chain wNoRek REKENING;
if %found(REKENING);
   delete REKENINGR;
endif;

// Melepas kunci tanpa mengubah
chain wNoRek REKENING;
if %found(REKENING);
   unlock REKENING;
endif;
```

Yang wajib diperhatikan System Engineer. `CHAIN` pada file yang dibuka `usage(*update)` **mengunci record**. Bila program lupa melakukan `update`, `delete`, atau `unlock`, kunci itu bertahan sampai record berikutnya dibaca atau program berakhir. Inilah penyebab paling umum dari job berstatus LCKW.

44.6 Embedded SQL

```
**free
dcl-s wSaldo  packed(17:2);
dcl-s wStatus char(1);
dcl-s wNoRek  char(16);

// Satu baris
exec sql
  SELECT SALDO, STATUS
    INTO :wSaldo, :wStatus
    FROM DTALIB.REKENING
   WHERE NO_REK = :wNoRek;

if sqlcode = 0;
   // berhasil
elseif sqlcode = 100;
   // tidak ditemukan
else;
   // kesalahan, sqlcode negatif
endif;

// Banyak baris dengan cursor
exec sql
  DECLARE curRek CURSOR FOR
    SELECT NO_REK, SALDO, KD_PRODUK
      FROM DTALIB.REKENING
     WHERE STATUS = 'A'
       AND KD_CABANG = :wKdCabang
     ORDER BY NO_REK
```

```
     FOR UPDATE OF SALDO;

exec sql OPEN curRek;

dou sqlcode <> 0;
   exec sql
     FETCH curRek INTO :wNoRek, :wSaldo, :wProduk;

   if sqlcode <> 0;
      leave;
   endif;

   wBunga = hitungBunga(wSaldo: wProduk);

   exec sql
     UPDATE DTALIB.REKENING
        SET SALDO = SALDO + :wBunga
      WHERE CURRENT OF curRek;
enddo;

exec sql CLOSE curRek;
```

Pola blok untuk kinerja. Mengambil satu baris per kali dari jutaan record itu lambat. Ambil per blok:

```
**free
dcl-ds dsBlok qualified dim(500);
  noRek  char(16);
  saldo  packed(17:2);
  produk char(4);
end-ds;
dcl-s wJmlAmbil int(10);
dcl-s i int(10);

exec sql OPEN curRek;

dou wJmlAmbil < 500;
   exec sql
     FETCH curRek FOR 500 ROWS INTO :dsBlok;

   wJmlAmbil = sqler3;   // jumlah baris yang benar-benar terambil

   for i = 1 to wJmlAmbil;
      // proses dsBlok(i)
   endfor;
enddo;

exec sql CLOSE curRek;
```

Perbedaan kinerja antara fetch satu-satu dan fetch blok 500 pada tabel sepuluh juta record bisa mencapai lima sampai sepuluh kali lipat. Ini salah satu perbaikan termudah untuk mempercepat EOD.

44.7 Prosedur dan service program

```
**free
// ---- File: SVCBUNGA.sqlrpgle ----
ctl-opt nomain option(*srcstmt: *nodebugio);

/copy APPSRC/QRPGLEREF,SVCBUNGA_H

// Prosedur yang diekspor
dcl-proc hitungBunga export;
```

```
  dcl-pi *n packed(17:2);
    pSaldo  packed(17:2) const;
    pProduk char(4)      const;
    pHari   int(5)       const options(*nopass);
  end-pi;

  dcl-s wRate    packed(9:6);
  dcl-s wHari    int(5);
  dcl-s wHitung  packed(30:10);

  // Parameter opsional
  if %parms() >= 3;
     wHari = pHari;
  else;
     wHari = 365;
  endif;

  wRate = ambilRate(pProduk);

  if wRate = 0;
     kirimEscape('Rate tidak terdaftar untuk produk ' + pProduk);
  endif;

  // Kalikan dulu, bagi terakhir
  wHitung = pSaldo * wRate;
  return %dech(wHitung / wHari: 17: 2);

end-proc;

// Prosedur internal, tidak diekspor
dcl-proc ambilRate;
  dcl-pi *n packed(9:6);
    pProduk char(4) const;
  end-pi;

  dcl-s wRate packed(9:6) inz(0);

  exec sql
    SELECT RATE INTO :wRate
      FROM APPPRM.PRODUK_RATE
     WHERE KD_PRODUK = :pProduk
       AND TGL_BERLAKU <= CURRENT DATE
     ORDER BY TGL_BERLAKU DESC
     FETCH FIRST 1 ROW ONLY;

  if sqlcode <> 0;
     return 0;
  endif;

  return wRate;
end-proc;
```

File header yang di-copy oleh pemakainya:

```
**free
// ---- File: SVCBUNGA_H ----
dcl-pr hitungBunga packed(17:2) extproc('HITUNGBUNGA');
  pSaldo  packed(17:2) const;
  pProduk char(4)      const;
  pHari   int(5)       const options(*nopass);
end-pr;
```

Kompilasi dan binding:

```
/* Modul */
CRTSQLRPGI OBJ(APPPGM/SVCBUNGA) SRCSTMF('/src/svcbunga.sqlrpgle') +
           OBJTYPE(*MODULE) DBGVIEW(*SOURCE) COMMIT(*CHG)

/* Service program */
CRTSRVPGM SRVPGM(APPPGM/SVCBUNGA) MODULE(APPPGM/SVCBUNGA) +
          EXPORT(*SRCFILE) SRCFILE(APPSRC/QSRVSRC) +
          ACTGRP('APPGRP')

/* Binding directory agar program pemakai menemukannya */
CRTBNDDIR BNDDIR(APPPGM/APPBND)
ADDBNDDIRE BNDDIR(APPPGM/APPBND) OBJ((APPPGM/SVCBUNGA *SRVPGM))
```

44.8 Binder language — kenapa ini urusan System Engineer

Service program punya *signature*. Bila daftar prosedur yang diekspor berubah, signature ikut berubah, dan **semua program yang terikat padanya langsung gagal** dengan `CPF4218`.

```
/* File: QSRVSRC anggota SVCBUNGA */
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('BUNGA_V2')
  EXPORT SYMBOL('HITUNGBUNGA')
  EXPORT SYMBOL('HITUNGDENDA')
  EXPORT SYMBOL('HITUNGPINALTI')   /* tambahan baru, di akhir */
ENDPGMEXP

/* Versi lama dipertahankan agar program lama tetap jalan */
STRPGMEXP PGMLVL(*PRV) SIGNATURE('BUNGA_V1')
  EXPORT SYMBOL('HITUNGBUNGA')
  EXPORT SYMBOL('HITUNGDENDA')
ENDPGMEXP
```

Aturan yang menyelamatkan Anda dari insiden: **prosedur baru selalu ditambahkan di akhir daftar, jangan pernah di tengah, dan jangan pernah menghapus yang lama.** Melanggar ini berarti seluruh aplikasi harus dikompilasi ulang serentak.

44.9 Penanganan kesalahan

```
**free
// MONITOR: blok try-catch RPG
monitor;
   wHasil = wPembilang / wPenyebut;
on-error *all;
   wHasil = 0;
   kirimPesan('Pembagian gagal, hasil diset nol');
endmon;

// Menangkap kode kesalahan tertentu
monitor;
   chain wKunci REKENING;
on-error 1218;   // record terkunci job lain
   kirimPesan('Record sedang dipakai job lain');
on-error *all;
   kirimPesan('Kesalahan tak terduga saat membaca');
endmon;

// Indikator error pada operasi file
chain(e) wKunci REKENING;
```

```
if %error();
   // periksa %status() untuk detail
endif;
```

%status()	Arti
1211	File belum dibuka
1218	Record terkunci oleh job lain
1221	Update tanpa CHAIN sebelumnya
1222	Record terkunci, timeout
0100	Nilai di luar jangkauan
0102	Pembagian dengan nol
0112	Tanggal atau waktu tidak valid
0202	Pemanggilan gagal

44.10 Commitment control

Inilah yang menjadikan transaksi perbankan aman.

```
**free
ctl-opt dftactgrp(*no) actgrp('TRXGRP');

dcl-f REKENING usage(*update) keyed commit;
dcl-f GLTRX    usage(*output)        commit;

// Transfer antar rekening: dua sisi harus sukses bersama
monitor;

   // Debet rekening asal
   chain wRekAsal REKENING;
   if not %found(REKENING) or SALDO < wJumlah;
      exec sql ROLLBACK;
      kirimEscape('Saldo tidak mencukupi');
   endif;
   SALDO -= wJumlah;
   update REKENINGR;

   // Kredit rekening tujuan
   chain wRekTujuan REKENING;
   if not %found(REKENING);
      exec sql ROLLBACK;
      kirimEscape('Rekening tujuan tidak ditemukan');
   endif;
   SALDO += wJumlah;
   update REKENINGR;

   // Catat ke GL
   clear GLTRXR;
   GL_REF    = wNoRef;
   GL_DEBET  = wJumlah;
   GL_KREDIT = wJumlah;
   write GLTRXR;

   // Kedua sisi selesai, kunci hasilnya
   exec sql COMMIT;

on-error *all;
   exec sql ROLLBACK;
   kirimEscape('Transfer dibatalkan karena kesalahan sistem');
```

```
endmon;
```

```
/* Job harus dijalankan di bawah commitment control */
STRCMTCTL LCKLVL(*CHG) CMTSCOPE(*ACTGRP)
CALL PGM(APPPGM/TRANSFER)
ENDCMTCTL
```

Lock level	Perilaku	Kapan dipakai
`*CHG`	Kunci record yang diubah saja	Paling umum
`*CS`	Cursor stability, kunci saat dibaca	Perlu konsistensi baca
`*ALL`	Kunci semua yang dibaca sampai commit	Paling ketat, paling lambat

Yang sering menjadi masalah operasional. Program yang membuka commitment control tetapi tidak pernah commit atau rollback akan menahan kunci sampai job berakhir. Di EOD, ini terlihat sebagai job yang tampak berjalan tetapi menahan seluruh rantai.

```
-- Melihat transaksi yang belum di-commit
SELECT JOB_NAME, COMMITMENT_DEFINITION,
       STATE, LOCK_LEVEL, LOCK_COUNT,
       PENDING_CHANGES
  FROM QSYS2.DB_TRANSACTION_INFO
 WHERE STATE <> 'NONE';
```

💻 Bab 45. Membaca Kode RPG Warisan

45.1 Menerjemahkan fixed format

Banyak kode produksi masih fixed format. Peta cepat:

Kolom C-spec	Isi
6	Tipe spec, selalu `C`
7–8	Level break atau indikator
9–11	Indikator kondisi
12–25	Factor 1
26–35	Kode operasi
36–49	Factor 2
50–63	Result field
64–68	Panjang dan desimal
71–76	Indikator hasil

```
     C     WKUNCI        CHAIN     REKENING                 50
     C                   IF        *IN50 = *OFF
     C                   EVAL      WSALDO = SALDO
     C                   ENDIF
```

Terjemahannya:

```
**free
chain wKunci REKENING;
if %found(REKENING);
   wSaldo = SALDO;
endif;
```

Indikator `50` di kolom 71–72 menyala bila record **tidak** ditemukan — kebalikan dari `%found()`. Ini sumber kesalahpahaman paling sering saat membaca kode lama.

45.2 Indikator yang perlu dikenali

Indikator	Arti tradisional
`*IN01-*IN99`	Umum, dipakai bebas
`*INLR`	Last record; menyalakannya menutup file dan mengakhiri program
`*INRT`	Return tanpa menutup file
`*INKA-*INKY`	Tombol fungsi F1–F24 pada layar
`*IN1P`	First page

`*INLR = *ON` adalah baris terakhir hampir setiap program RPG. Bila lupa, program berakhir tetapi file tetap terbuka dan variabel mempertahankan nilai lama pada pemanggilan berikutnya — sumber bug yang sangat membingungkan.

45.3 Menelusuri ketergantungan program

```
/* Objek apa saja yang dipakai program ini */
DSPPGMREF PGM(APPPGM/HITBUNGA) OUTPUT(*PRINT)

/* Untuk seluruh library, ke outfile */
DSPPGMREF PGM(APPPGM/*ALL) OBJTYPE(*PGM) +
          OUTPUT(*OUTFILE) OUTFILE(QTEMP/PGMREF)
```

```
-- Program apa saja yang memakai tabel tertentu
SELECT WHPNAM AS program, WHLNAM AS library,
       WHFNAM AS objek_dipakai, WHOTYP AS tipe,
       WHUSAG AS cara_pakai
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'REKENING'
 ORDER BY WHPNAM;

-- Modul dan service program yang terikat pada satu program
SELECT * FROM TABLE(QSYS2.BOUND_MODULE_INFO('APPPGM','EODBUNGA'));

SELECT * FROM TABLE(QSYS2.BOUND_SRVPGM_INFO('APPPGM','EODBUNGA'));
```

Tiga kueri ini adalah alat utama Anda saat menjawab pertanyaan "kalau tabel ini diubah, apa saja yang rusak?"

45.4 Pertanyaan yang harus Anda ajukan saat membaca program produksi

1. Apakah dijalankan di bawah commitment control?
2. Activation group apa yang dipakai, dan apakah `*NEW` di dalam perulangan?
3. Apakah semua `CHAIN` diikuti `update`, `delete`, atau `unlock`?
4. Apakah ada penanganan error, atau kesalahan akan menjatuhkan program?
5. Apakah ada jalur kode yang diam-diam membukukan nol?
6. Berapa banyak I/O per record — apakah ada kueri di dalam perulangan?
7. Apakah program idempoten bila dijalankan dua kali?

Pertanyaan nomor tujuh adalah yang paling penting untuk lingkungan EOD, dan yang paling jarang bisa dijawab dengan yakin.

📖 Bagian XII — Katalog Message ID

Setiap insiden IBM i berbicara lewat message ID. Bagian ini adalah kamus kerja: apa artinya, apa penyebab yang paling sering, dan apa tindakan pertama yang benar.

Bab 46. Membaca Message ID

46.1 Struktur kode

Awalan	Sumber
`CPF`	Fungsi sistem OS/400, paling umum
`CPA`	Pesan yang menunggu jawaban operator
`CPC`	Penyelesaian sukses
`CPD`	Diagnostik perintah
`CPI`	Informasi
`MCH`	Machine check, level SLIC
`RNQ`	RPG runtime, menunggu jawaban
`RNX`	RPG runtime, escape
`SQL`	Db2 for i
`TCP`	Komunikasi TCP/IP
`SEC`	Keamanan

Selalu baca **teks tingkat kedua**. Itu bagian yang berisi penyebab dan pemulihan, dan sembilan dari sepuluh kali sudah menjawab pertanyaan Anda.

```
DSPMSGD RANGE(CPF4131) MSGF(QSYS/QCPFMSG) DETAIL(*FULL)
```

```
SELECT MESSAGE_ID, MESSAGE_TEXT, MESSAGE_SECOND_LEVEL_TEXT,
       SEVERITY, MESSAGE_DATA
  FROM QSYS2.MESSAGE_FILE_DATA
 WHERE MESSAGE_FILE = 'QCPFMSG'
   AND MESSAGE_ID = 'CPF4131';
```

📖 Bab 47. Katalog Pesan Operasional

47.1 Penyimpanan

ID	Arti	Penyebab lazim	Tindakan pertama
`CPF0907`	Ambang penyimpanan tercapai	ASP mendekati penuh	RB-001, bersihkan segera
`CPI0948`	Ambang ASP terlampaui	Idem	RB-001
`CPF0670`	Penyimpanan tambahan penuh	ASP 100%	Darurat, hentikan job penulis
`CPI099C`	Journal receiver ambang	Receiver tumbuh cepat	Simpan lalu lepas receiver

ID	Arti	Penyebab lazim	Tindakan pertama
CPF3782	File tidak dijurnal	Restore tanpa journaling	Mulai journaling ulang
CPF7013	Receiver tidak ditemukan	Receiver terhapus prematur	Pemulihan terbatas, eskalasi

47.2 Job dan work management

ID	Arti	Penyebab lazim	Tindakan pertama
CPF1240	Job berakhir abnormal	Program gagal	Baca job log job tersebut
CPF1241	Job berakhir normal	Informasi	Tidak ada
CPF1352	Job tidak bisa dimulai	Batas job tercapai	RB-012
CPF1269	Job tidak dimulai, error subsystem	Routing entry salah	Periksa DSPSBSD
CPF1338	Perintah tidak sah untuk job	Kesalahan tipe job	Periksa konteks eksekusi
CPA0700	Pesan escape dari program	Program gagal	Baca detail escape
CPF1002	Objek tidak bisa dialokasikan	Sudah dikunci job lain	RB-003
CPF1907	Job dihentikan	ENDJOB dijalankan	Periksa siapa dan kenapa

47.3 File dan basis data

ID	Arti	Penyebab lazim	Tindakan pertama
CPF4131	Level check	Struktur file berubah	RB-009, kompilasi ulang
CPF5029	Kesalahan data I/O	Data tidak sesuai definisi	Periksa record penyebab
CPF5027	Record dipakai job lain	Kontensi kunci	RB-003
CPF5032	Record tidak ditemukan	Kunci salah	Periksa logika program
CPF5116	Nilai kunci duplikat	Pelanggaran unique	Periksa data sumber
CPF4328	Anggota tidak ditemukan	Override salah	Periksa DSPOVR
CPF9801	Objek tidak ditemukan	Library list salah	Periksa DSPLIBL
CPF9810	Library tidak ditemukan	Lingkungan salah	Periksa library list
CPF9812	File tidak ditemukan	Idem	Idem
CPF9815	Anggota file tidak ditemukan	Member belum dibuat	ADDPFM

47.4 Keamanan

ID	Arti	Penyebab lazim	Tindakan pertama
CPF2189	Tidak berwenang atas objek	Wewenang kurang	Periksa DSPOBJAUT
CPF2217	Tidak berwenang atas profil	Wewenang kurang	Eskalasi ke Security
CPF1394	Profil tidak bisa login	Profil dinonaktifkan	RB-010
CPF1107	Password salah	Kesalahan input atau serangan	Periksa pola PW di audit
CPF1120	Pengguna tidak ada	Profil terhapus	Periksa audit DO
CPF2204	Profil tidak ditemukan	Idem	Idem

ID	Arti	Penyebab lazim	Tindakan pertama
`CPD0032`	Tidak berwenang atas perintah	Pembatasan perintah	Periksa `DSPOBJAUT` pada `*CMD`

47.5 Backup dan media

ID	Arti	Penyebab lazim	Tindakan pertama
`CPA5305`	Media berikutnya diminta	Tape penuh	Ganti media, jawab dengan benar
`CPF3793`	Media tidak mencukupi	Volume kurang	RB-008
`CPF3812`	Objek sedang dipakai	Lock saat save	Akhiri sesi penahan
`CPF3770`	Tidak ada objek yang disimpan	Library kosong atau salah	Periksa nama library
`CPF4257`	Perangkat tidak siap	Tape drive bermasalah	`VRYCFG` dan periksa fisik
`CPC3701`	Save selesai	Sukses	Verifikasi jumlah objek
`CPF3730`	Objek tidak disimpan	Sebagian gagal	Periksa daftar objek gagal

47.6 RPG runtime

ID	Arti	Penyebab lazim	Tindakan pertama
`RNQ0100`	Pembagian dengan nol	Parameter kosong	Periksa tabel parameter
`RNQ0103`	Hasil melebihi kapasitas field	Akumulasi terlalu besar	Periksa definisi field
`RNQ0112`	Tanggal atau waktu tidak valid	Data kosong atau format salah	Periksa data sumber
`RNQ0115`	Variabel di luar jangkauan	Index array salah	Periksa batas perulangan
`RNQ0202`	Pemanggilan gagal	Program tidak ditemukan	Periksa library list
`RNQ0211`	Program tidak ditemukan	Idem	Idem
`RNQ0221`	Parameter tidak dikirim	Jumlah parameter tidak cocok	Periksa `%parms()`
`RNQ0231`	Operasi pada file tertutup	Urutan open/close salah	Periksa logika program
`RNQ0401`	Kesalahan konversi data	Tipe tidak cocok	Periksa data masukan
`RNX0301`	Kesalahan dalam prosedur	Beragam	Baca call stack di dump

47.7 ILE dan binding

ID	Arti	Penyebab lazim	Tindakan pertama
`CPF4218`	Signature service program tidak cocok	Service program dikompilasi ulang	Kompilasi ulang program pemakai
`CPF5D03`	Prosedur tidak ditemukan	Export hilang dari binder source	Periksa `QSRVSRC`
`CPD5D03`	Kesalahan binding	Binding directory salah	Periksa `DSPBNDDIR`
`CPF2451`	Activation group tidak ditemukan	Konfigurasi `ACTGRP`	Periksa `ctl-opt`

47.8 SQL

SQLCODE	Arti	Tindakan
`0`	Berhasil	Lanjut
`100`	Tidak ada baris ditemukan	Tangani sebagai kondisi normal
`-104`	Token tidak sah	Perbaiki sintaks
`-204`	Objek tidak ditemukan	Periksa nama dan library list
`-206`	Kolom tidak ditemukan	Periksa nama kolom
`-302`	Nilai terlalu besar untuk host variable	Perbesar definisi variabel
`-407`	Null ke kolom NOT NULL	Periksa data sumber
`-501`	Cursor belum dibuka	Periksa urutan OPEN
`-502`	Cursor sudah dibuka	Tutup dulu
`-803`	Kunci duplikat	Periksa data atau logika
`-904`	Sumber daya tidak tersedia	Sering berarti tabel terkunci
`-911`	Deadlock atau timeout, rollback terjadi	Coba ulang transaksi
`-913`	Deadlock atau timeout, tanpa rollback	Tangani lalu rollback
`-7008`	Objek tidak valid untuk operasi ini	Sering: tabel tidak dijurnal padahal commit aktif

`SQLCODE -7008` layak diingat khusus: ia muncul ketika program berjalan di bawah commitment control tetapi tabel yang disentuh belum dijurnal. Di lingkungan perbankan, ini biasanya berarti ada tabel yang lolos dari konfigurasi journaling.

47.9 Komunikasi dan TCP/IP

ID	Arti	Tindakan pertama
`TCP1A04`	Server sudah aktif	Tidak ada
`TCP2665`	Interface tidak aktif	`STRTCPIFC`
`CPF9899`	Kesalahan umum aplikasi	Baca detail di job log
`CPE3425`	Koneksi ditolak	Periksa firewall dan port
`CPE3426`	Koneksi diputus pihak lain	Periksa sisi klien
`CPDBC01`	Kesalahan koneksi database	Periksa server host

Bab 48. Membangun Kamus Pesan Internal

Setiap institusi punya pola pesan khasnya sendiri. Bangun kamus Anda sendiri dari data nyata.

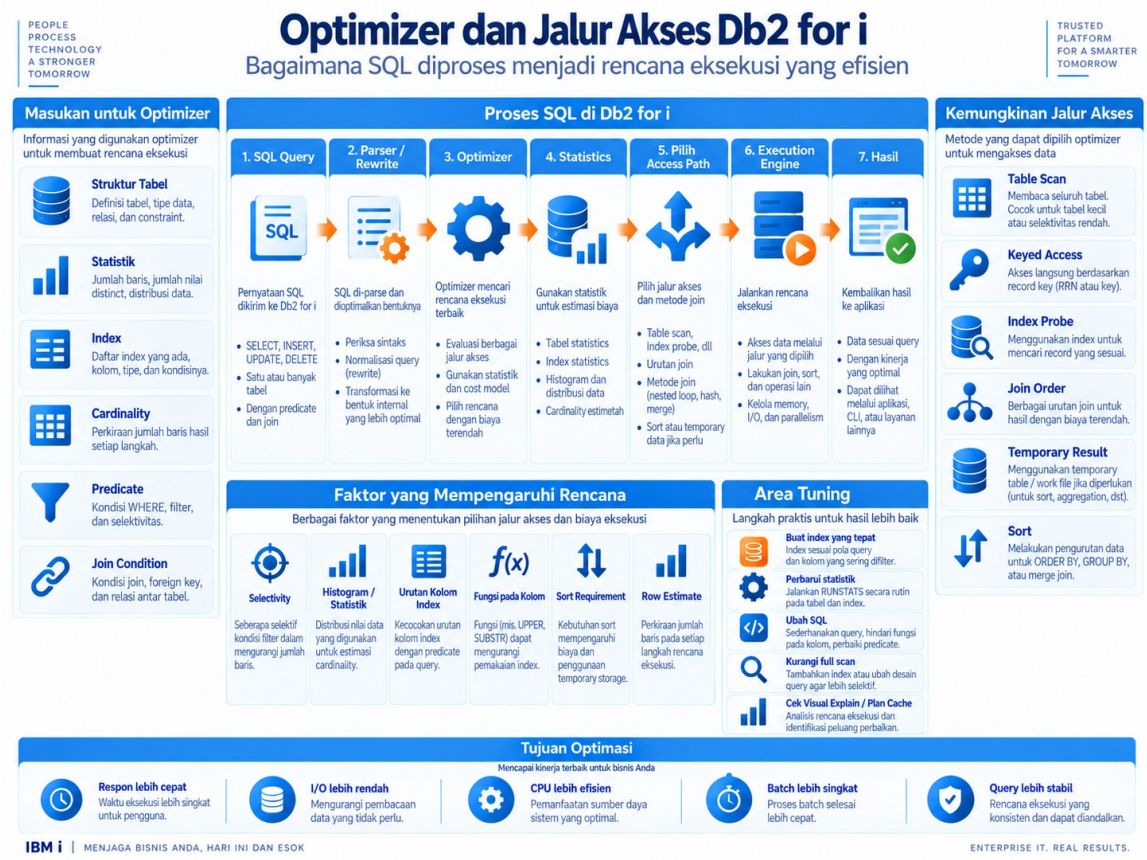
```
-- Tabel penampung riwayat pesan
CREATE TABLE APPWRK.PESAN_RIWAYAT (
    WAKTU       TIMESTAMP NOT NULL,
    MESSAGE_ID  CHAR(7)   NOT NULL,
    SEVERITY    SMALLINT,
    JOB_NAME    VARCHAR(50),
    PROGRAM     VARCHAR(30),
    TEKS        VARCHAR(500)
);

-- Diisi harian dari history log
INSERT INTO APPWRK.PESAN_RIWAYAT
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, SEVERITY,
       FROM_JOB, FROM_PROGRAM, MESSAGE_TEXT
  FROM TABLE(QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 24 HOURS))
 WHERE SEVERITY >= 30;
```

```
-- Setelah beberapa bulan: pesan apa yang paling sering mengganggu
SELECT MESSAGE_ID,
       COUNT(*) AS frekuensi,
       COUNT(DISTINCT DATE(WAKTU)) AS jumlah_hari,
       MIN(WAKTU) AS pertama,
       MAX(WAKTU) AS terakhir,
       MAX(TEKS) AS contoh_teks
  FROM APPWRK.PESAN_RIWAYAT
 WHERE WAKTU > CURRENT DATE - 180 DAYS
 GROUP BY MESSAGE_ID
 ORDER BY frekuensi DESC
 FETCH FIRST 30 ROWS ONLY;

-- Pesan yang muncul hanya saat EOD
SELECT MESSAGE_ID, COUNT(*) AS jml
  FROM APPWRK.PESAN_RIWAYAT
 WHERE HOUR(WAKTU) BETWEEN 17 AND 6
   AND SEVERITY >= 40
 GROUP BY MESSAGE_ID
 ORDER BY jml DESC;
```

Daftar tiga puluh pesan teratas dari data Anda sendiri lebih berharga daripada seluruh katalog di bab ini, karena ia mencerminkan sistem Anda, bukan sistem umum. Jadikan daftar itu sebagai dasar prioritas penulisan runbook berikutnya: pesan yang paling sering muncul dan paling sering butuh penanganan manual adalah kandidat otomasi pertama.

🗄️ Bagian XIII — Db2 for i Mendalam

👥 Bab 49. Optimizer dan Jalur Akses



Gambar 54 — Optimizer dan pemilihan jalur akses Db2 for i

Optimizer Db2 for i memutuskan bagaimana sebuah kueri dijalankan. Memahami keputusannya adalah keterampilan tunggal yang paling berdampak untuk mempercepat EOD.

49.1 Dua mesin kueri

	CQE (Classic)	SQE (SQL Query Engine)
Asal	Warisan lama	Modern, terus dikembangkan
Statistik	Terbatas	Statistik kolom penuh
Rencana disimpan	Tidak	Plan cache
Paralelisme	Terbatas	SMP penuh
Dipakai untuk	Kasus khusus, logical file DDS lama	Hampir semua kueri modern

Sebagian besar kueri kini melewati SQE. Yang masih jatuh ke CQE biasanya karena memakai logical file DDS sebagai tabel atau fitur lama tertentu. Bila kinerja terasa aneh dan Anda melihat CQE dipakai, itu petunjuk pertama.

49.2 Melihat keputusan optimizer

```
-- Aktifkan pesan debug untuk job saat ini
CALL QSYS2.OVERRIDE_QAQQINI(1, 'MESSAGES_DEBUG', '*YES');

-- Jalankan kueri yang diselidiki
SELECT COUNT(*) FROM DTALIB.TRXHIST WHERE TGL_TRX = CURRENT DATE;

-- Baca pesan optimizer di job log
SELECT MESSAGE_ID, MESSAGE_TEXT, MESSAGE_SECOND_LEVEL_TEXT
  FROM TABLE(QSYS2.JOBLOG_INFO('*'))
 WHERE MESSAGE_ID LIKE 'CPI43%'
 ORDER BY ORDINAL_POSITION DESC
 FETCH FIRST 30 ROWS ONLY;
```

Pesan	Arti	Tindakan
`CPI4321`	Index sementara dibangun	Buat index permanen
`CPI4322`	Table scan	Cari index yang cocok
`CPI4324`	File sementara dibangun	Sering karena `ORDER BY` atau `GROUP BY`
`CPI4326`	Index dipakai untuk akses	Baik
`CPI4327`	Index dipakai untuk seleksi	Baik
`CPI4328`	Index dipakai untuk urutan	Baik
`CPI432F`	Index disarankan	Catat, lalu timbang

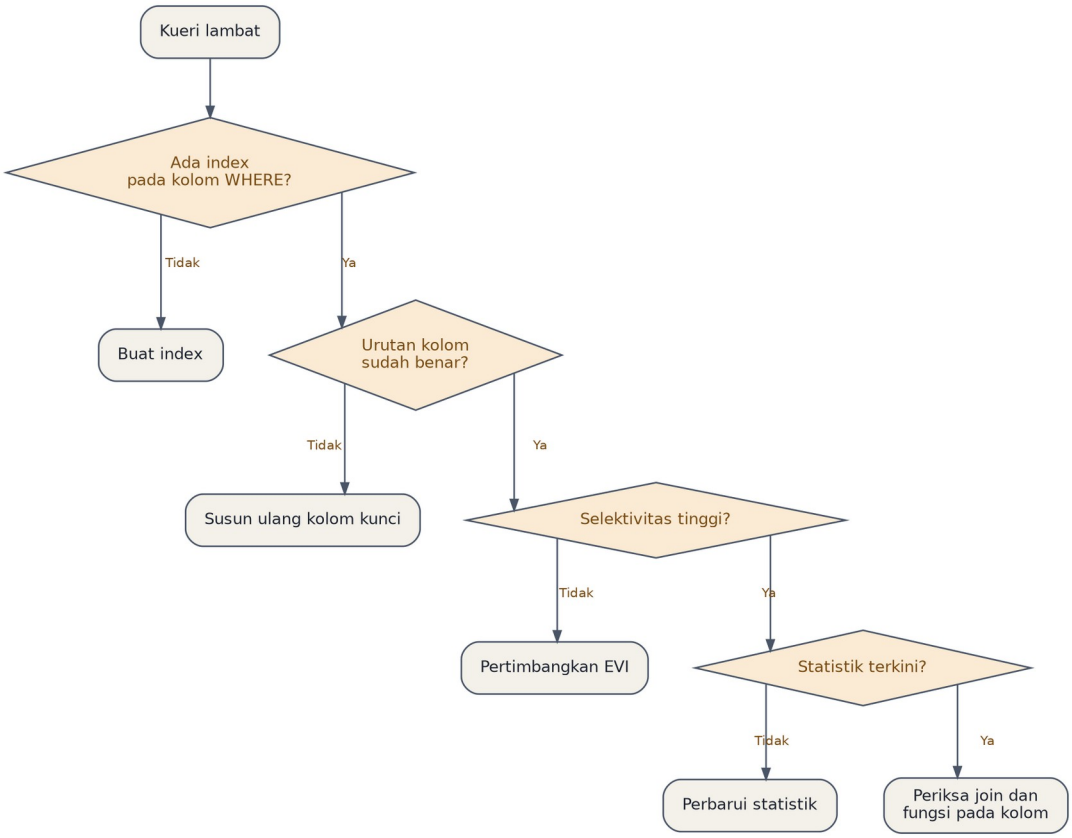
49.3 Plan cache

Rencana eksekusi disimpan dan dipakai ulang. Ini tambang informasi untuk menemukan kueri mahal.

```
-- Buang isi plan cache ke tabel untuk dianalisis
CALL QSYS2.DUMP_PLAN_CACHE('APPWRK', 'PLANCACHE');

-- Statement paling boros
SELECT QQ1000 AS teks,
       QQI2   AS jumlah_eksekusi,
       QQI6   AS total_waktu_mikro,
       QQI6 / NULLIF(QQI2,0) / 1000 AS rata_ms,
       QVC3001 AS index_disarankan
  FROM APPWRK.PLANCACHE
 WHERE QQRID = 3006
 ORDER BY QQI6 DESC
 FETCH FIRST 25 ROWS ONLY;
```

Jalankan ini sebulan sekali dan simpan hasilnya. Perbandingan antar bulan menunjukkan kueri mana yang memburuk seiring pertumbuhan data — jauh sebelum penggunanya mengeluh.

49.4 Strategi index



Gambar 55

Aturan urutan kolom kunci. Kolom dengan kondisi kesetaraan lebih dulu, lalu kolom rentang, lalu kolom untuk pengurutan.

```
-- Untuk: WHERE KD_CABANG = ? AND TGL_TRX BETWEEN ? AND ? ORDER BY NO_REK
CREATE INDEX DTALIB.TRXHIST_IX1
    ON DTALIB.TRXHIST (KD_CABANG, TGL_TRX, NO_REK);
```

Encoded Vector Index cocok untuk kolom dengan sedikit nilai berbeda, seperti status atau kode produk.

```
CREATE ENCODED VECTOR INDEX DTALIB.REKENING_EVI1
    ON DTALIB.REKENING (STATUS)
    WITH 10 DISTINCT VALUES;

-- EVI dengan agregat yang sudah dihitung sebelumnya
CREATE ENCODED VECTOR INDEX DTALIB.REKENING_EVI2
    ON DTALIB.REKENING (KD_PRODUK)
    INCLUDE (SUM(SALDO), COUNT(*));
```

EVI dengan `INCLUDE` menyimpan agregat, sehingga `SELECT KD_PRODUK, SUM(SALDO) ... GROUP BY KD_PRODUK` dijawab tanpa membaca tabel sama sekali. Untuk laporan neraca harian, ini bisa mengubah kueri lima menit menjadi di bawah satu detik.

Kapan index justru merugikan

Situasi	Alasan
Tabel yang ditulis jutaan kali saat batch	Tiap insert memelihara tiap index
Selektivitas rendah tanpa EVI	Optimizer mengabaikannya
Banyak index tumpang tindih	Ruang terbuang, pemeliharaan ganda
Index yang tidak pernah dipakai	Biaya murni

```
-- Beban pemeliharaan index per tabel
SELECT TABLE_NAME,
       COUNT(*) AS jumlah_index,
       SUM(INDEX_SIZE) / 1048576 AS total_index_mb,
       SUM(CASE WHEN LAST_QUERY_USE IS NULL THEN 1 ELSE 0 END) AS tak_terpakai
  FROM QSYS2.SYSPARTITIONINDEXSTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
 GROUP BY TABLE_NAME
 ORDER BY jumlah_index DESC;
```

49.5 QAQQINI — menyetel perilaku optimizer

Parameter	Nilai	Efek
`QUERY_TIME_LIMIT`	angka detik	Tolak kueri yang diperkirakan lebih lama
`PARALLEL_DEGREE`	`*OPTIMIZE`, `*MAX`, `*NONE`	Kendali SMP
`MEMORY_POOL_PREFERENCE`	nama pool	Arahkan kueri ke pool tertentu
`ALLOW_TEMPORARY_INDEXES`	`*YES` / `*NO`	Cegah pembangunan index sementara
`MESSAGES_DEBUG`	`*YES` / `*NO`	Pesan optimizer ke job log

```
INSERT INTO QUSRSYS.QAQQINI (QQPARM, QQVAL, QQTEXT)
VALUES ('QUERY_TIME_LIMIT', '300', 'Batas 5 menit untuk kueri pelaporan');
```

`QUERY_TIME_LIMIT` layak dipertimbangkan untuk lingkungan pelaporan. Ia menolak kueri sembrono sebelum dijalankan, alih-alih membiarkannya menghabiskan CPU selama dua jam.

49.6 Statistik kolom

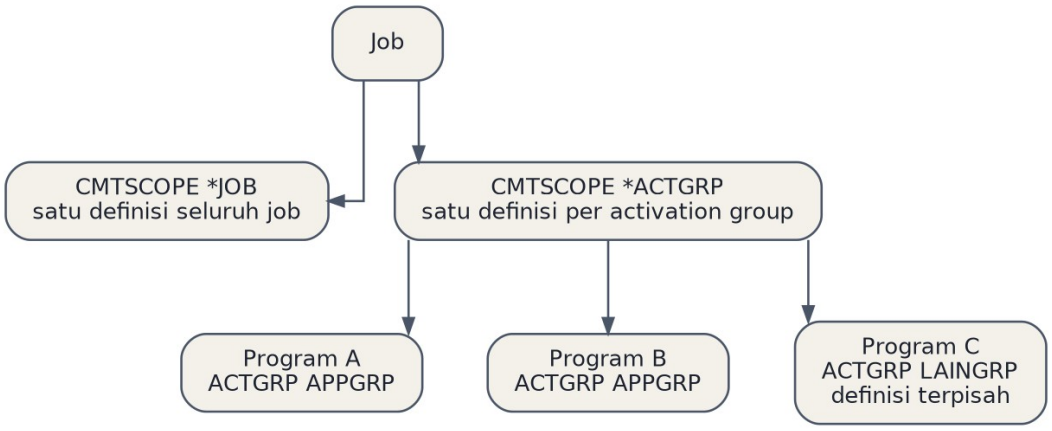
```
-- Statistik yang ada dan kapan terakhir diperbarui
SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAMES,
       LAST_STATISTICS_USE, LAST_REFRESH_TIME,
       NUMBER_DISTINCT_VALUES
  FROM QSYS2.SYSCOLUMNSTAT
 WHERE TABLE_SCHEMA = 'DTALIB';

-- Memaksa pembaruan statistik untuk satu tabel
CALL QSYS2.ENSURE_STATISTICS('DTALIB', 'TRXHIST');
```

Setelah pemuatan data besar atau reorganisasi, statistik bisa usang dan optimizer mengambil keputusan berdasarkan gambaran lama. Perbarui statistik sebagai langkah penutup setiap migrasi data besar.

Bab 50. Commitment Control dan Integritas Transaksi

Gambar 56 — Commitment control dan integritas transaksi

50.1 Cakupan commit



Gambar 57

Bahaya yang sering luput: dua program di activation group berbeda melakukan commit secara **terpisah**. Program A bisa commit sementara program B rollback, dan Anda mendapat transaksi setengah jadi tanpa satu pun pesan kesalahan.

50.2 Savepoint untuk transaksi bertingkat

```
-- Pola: sebagian boleh gagal tanpa membatalkan seluruh transaksi
SAVEPOINT sp_biaya ON ROLLBACK RETAIN CURSORS;

UPDATE DTALIB.REKENING
   SET SALDO = SALDO - :wBiaya
 WHERE NO_REK = :wNoRek;

-- Bila pembebanan biaya gagal, batalkan hanya bagian ini
ROLLBACK TO SAVEPOINT sp_biaya;

RELEASE SAVEPOINT sp_biaya;
```

Berguna untuk EOD di mana kegagalan pada satu rekening tidak boleh membatalkan seluruh batch.

50.3 Memantau transaksi terbuka

```
SELECT JOB_NAME, AUTHORIZATION_NAME,
       COMMITMENT_DEFINITION,
       STATE, LOCK_LEVEL, LOCK_COUNT,
       PENDING_CHANGES,
       LAST_COMMIT_TIMESTAMP
  FROM QSYS2.DB_TRANSACTION_INFO
 WHERE PENDING_CHANGES > 0
 ORDER BY PENDING_CHANGES DESC;
```

Job dengan `PENDING_CHANGES` besar dan `LAST_COMMIT_TIMESTAMP` yang sudah lama adalah tanda bahaya: ia menahan kunci pada ribuan record. Di tengah EOD, satu job semacam ini bisa menghentikan segalanya.

Batas praktis. Commit setiap beberapa ribu record, bukan sekali di akhir. Transaksi yang mencakup jutaan record menahan kunci terlalu lama, dan rollback-nya bisa berlangsung berjam-jam bila gagal.

```
**free
dcl-s wCounter int(10) inz(0);

dow not %eof(REKENING);
   // proses record
   update REKENINGR;

   wCounter += 1;
   if %rem(wCounter: 5000) = 0;
      exec sql COMMIT;
   endif;

   read REKENING;
enddo;

exec sql COMMIT;
```

50.4 Constraint sebagai pengaman terakhir

```
ALTER TABLE DTALIB.REKENING
  ADD CONSTRAINT PK_REKENING PRIMARY KEY (NO_REK);

ALTER TABLE DTALIB.TRXHIST
  ADD CONSTRAINT FK_TRX_REKENING
  FOREIGN KEY (NO_REK) REFERENCES DTALIB.REKENING (NO_REK)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT;

ALTER TABLE DTALIB.REKENING
  ADD CONSTRAINT CK_SALDO CHECK (SALDO >= 0);

ALTER TABLE DTALIB.REKENING
  ADD CONSTRAINT CK_STATUS CHECK (STATUS IN ('A','C','B','D'));
```

```
-- Constraint yang sedang check pending
SELECT CONSTRAINT_SCHEMA, CONSTRAINT_NAME, TABLE_NAME,
       CONSTRAINT_TYPE, ENABLED, CHECK_PENDING
  FROM QSYS2.SYSCST
 WHERE CHECK_PENDING = 'YES';
```

`CHECK_PENDING` menyala setelah restore atau pemuatan data yang melewati validasi. Selama status itu aktif, constraint tidak menjaga apa pun. Periksa kondisi ini sebagai bagian dari verifikasi pasca-restore.

50.5 Trigger dan biayanya

```
CREATE TRIGGER DTALIB.TRG_AUDIT_SALDO
  AFTER UPDATE OF SALDO ON DTALIB.REKENING
  REFERENCING OLD AS o NEW AS n
  FOR EACH ROW MODE DB2SQL
BEGIN
  IF o.SALDO <> n.SALDO THEN
    INSERT INTO DTALIB.AUDIT_SALDO
      (WAKTU, NO_REK, SALDO_LAMA, SALDO_BARU, PENGGUNA)
    VALUES (CURRENT TIMESTAMP, n.NO_REK, o.SALDO, n.SALDO, SESSION_USER);
  END IF;
END;
```

Peringatan kinerja. Trigger baris berjalan sekali per record. Pada fase akrual yang menyentuh dua juta rekening, trigger ini dijalankan dua juta kali. Untuk pencatatan perubahan massal, journaling jauh lebih murah dan sudah tersedia tanpa biaya tambahan.

```
SELECT TRIGGER_NAME, EVENT_OBJECT_TABLE, ACTION_TIMING,
       EVENT_MANIPULATION, ACTION_ORIENTATION, ENABLED
  FROM QSYS2.SYSTRIGGERS
 WHERE EVENT_OBJECT_SCHEMA = 'DTALIB';
```

🗄️ Bab 51. Fitur Db2 yang Sering Terlewat

51.1 Temporal table

Menyimpan riwayat perubahan secara otomatis, tanpa trigger dan tanpa kode aplikasi.

```
CREATE TABLE DTALIB.PRODUK_RATE (
    KD_PRODUK   CHAR(4)       NOT NULL,
    RATE        DECIMAL(9,6)  NOT NULL,
    MULAI       TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
    SELESAI     TIMESTAMP(12) GENERATED ALWAYS AS ROW END NOT NULL,
    TRX_ID      TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
    PERIOD SYSTEM_TIME (MULAI, SELESAI),
    PRIMARY KEY (KD_PRODUK)
);

CREATE TABLE DTALIB.PRODUK_RATE_H LIKE DTALIB.PRODUK_RATE;

ALTER TABLE DTALIB.PRODUK_RATE
  ADD VERSIONING USE HISTORY TABLE DTALIB.PRODUK_RATE_H;
```

```
-- Berapa rate produk ini pada tanggal tertentu?
SELECT KD_PRODUK, RATE
  FROM DTALIB.PRODUK_RATE
   FOR SYSTEM_TIME AS OF TIMESTAMP('2026-06-30 23:59:59')
 WHERE KD_PRODUK = 'DP01';

-- Seluruh riwayat perubahan
SELECT KD_PRODUK, RATE, MULAI, SELESAI
  FROM DTALIB.PRODUK_RATE
   FOR SYSTEM_TIME FROM '2026-01-01' TO '2026-12-31'
 ORDER BY MULAI;
```

Untuk tabel parameter perbankan — rate, biaya, kurs, kode produk — temporal table menjawab pertanyaan auditor "berapa nilainya pada tanggal itu" tanpa pekerjaan tambahan.

51.2 Fungsi analitik

```
-- Saldo berjalan dan peringkat dalam satu lintasan
SELECT NO_REK, TGL_TRX, JUMLAH,
       SUM(JUMLAH) OVER (PARTITION BY NO_REK
                         ORDER BY TGL_TRX
                         ROWS BETWEEN UNBOUNDED PRECEDING
                                  AND CURRENT ROW) AS saldo_berjalan,
       ROW_NUMBER() OVER (PARTITION BY NO_REK
                          ORDER BY TGL_TRX DESC) AS urutan_terbaru,
       LAG(JUMLAH, 1) OVER (PARTITION BY NO_REK
                            ORDER BY TGL_TRX) AS trx_sebelumnya
  FROM DTALIB.TRXHIST
 WHERE TGL_TRX >= CURRENT DATE - 30 DAYS;

-- Lima transaksi terbesar per cabang
SELECT * FROM (
  SELECT KD_CABANG, NO_REK, JUMLAH,
         RANK() OVER (PARTITION BY KD_CABANG ORDER BY JUMLAH DESC) AS peringkat
    FROM DTALIB.TRXHIST
   WHERE TGL_TRX = CURRENT DATE
) x
 WHERE peringkat <= 5;
```

Fungsi analitik sering menggantikan program RPG berpuluh baris dengan satu kueri, dan biasanya jauh lebih cepat karena optimizer bisa memparalelkannya.

51.3 Common Table Expression rekursif

```
-- Menelusuri hierarki unit kerja
WITH RECURSIVE hirarki (kd_unit, nm_unit, kd_induk, tingkat, jalur) AS (
  SELECT KD_UNIT, NM_UNIT, KD_INDUK, 1,
         CAST(KD_UNIT AS VARCHAR(200))
    FROM APPPRM.UNIT_KERJA
   WHERE KD_INDUK IS NULL
  UNION ALL
  SELECT u.KD_UNIT, u.NM_UNIT, u.KD_INDUK, h.tingkat + 1,
         h.jalur || ' > ' || u.KD_UNIT
    FROM APPPRM.UNIT_KERJA u
    JOIN hirarki h ON u.KD_INDUK = h.kd_unit
   WHERE h.tingkat < 10
)
SELECT tingkat, kd_unit, nm_unit, jalur
  FROM hirarki
 ORDER BY jalur;
```

Batas `tingkat < 10` bukan hiasan. Tanpa itu, data yang mengandung lingkaran referensi membuat kueri berjalan tanpa akhir.

51.4 MERGE untuk sinkronisasi data

```
MERGE INTO DTALIB.SALDO_HARIAN AS t
USING (SELECT NO_REK, SUM(JUMLAH) AS total
         FROM APPWRK.STAGING_TRX
        WHERE TGL_PROSES = CURRENT DATE
        GROUP BY NO_REK) AS s
   ON t.NO_REK = s.NO_REK AND t.TGL = CURRENT DATE
 WHEN MATCHED THEN
      UPDATE SET t.SALDO = t.SALDO + s.total,
                 t.DIUBAH = CURRENT TIMESTAMP
 WHEN NOT MATCHED THEN
      INSERT (NO_REK, TGL, SALDO, DIUBAH)
      VALUES (s.NO_REK, CURRENT DATE, s.total, CURRENT TIMESTAMP);
```

Satu `MERGE` menggantikan pola baca-periksa-tulis yang lazim ditulis sebagai perulangan RPG, dan berjalan jauh lebih cepat karena dioptimasi sebagai operasi set.

51.5 Fungsi tabel buatan sendiri

```
CREATE FUNCTION DTALIB.MUTASI_REKENING (
    p_norek  CHAR(16),
    p_dari   DATE,
    p_sampai DATE
)
RETURNS TABLE (
    tgl        DATE,
    keterangan VARCHAR(100),
    debet      DECIMAL(17,2),
    kredit     DECIMAL(17,2)
)
LANGUAGE SQL
READS SQL DATA
RETURN
  SELECT TGL_TRX, KETERANGAN,
         CASE WHEN JENIS = 'D' THEN JUMLAH ELSE 0 END,
         CASE WHEN JENIS = 'K' THEN JUMLAH ELSE 0 END
    FROM DTALIB.TRXHIST
```

```
   WHERE NO_REK = p_norek
     AND TGL_TRX BETWEEN p_dari AND p_sampai;

-- Pemakaian
SELECT * FROM TABLE(DTALIB.MUTASI_REKENING(
        '1234567890123456', '2026-09-01', '2026-09-30'));
```

Membungkus logika bisnis sebagai fungsi tabel memungkinkan kanal digital dan pelaporan memakai definisi yang sama persis, sehingga tidak ada lagi selisih angka antara laporan cabang dan mobile banking.

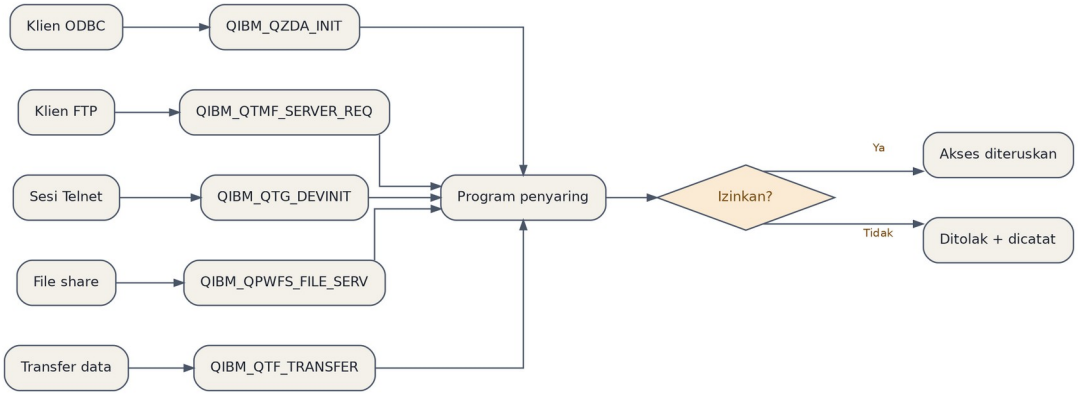
🔒 Bagian XIV — Keamanan Mendalam

Bab 52. Exit Program

Gambar 58 — Exit point dan exit program sebagai kendali akses

Wewenang objek melindungi objek. Ia tidak tahu apa-apa tentang siapa yang menyambung dari mana, jam berapa, dan dengan aplikasi apa. Exit program mengisi celah itu.

52.1 Peta titik masuk



Gambar 59

52.2 Exit program ODBC

Ini yang paling penting di lingkungan perbankan, karena ODBC adalah jalur paling sering dipakai untuk mengambil data dalam jumlah besar.

```
**free
ctl-opt dftactgrp(*no) actgrp('SECGRP');

dcl-pr odbcExit extpgm('ODBCEXIT');
  pIzinkan char(1);
  pInfo    likeds(dsZdai) const;
end-pr;

dcl-ds dsZdai qualified template;
  namaUser   char(10);
  namaSrv    char(10);
  format     char(8);
  fungsi     char(10);
  alamatIP   char(15);
end-ds;

dcl-proc odbcExit;
  dcl-pi *n;
    pIzinkan char(1);
    pInfo    likeds(dsZdai) const;
  end-pi;

  dcl-s wJam  int(5);
  dcl-s wJml  int(10);

  pIzinkan = '0';   // default menolak

  // 1. Apakah pengguna terdaftar boleh memakai ODBC?
  exec sql
    SELECT COUNT(*) INTO :wJml
      FROM SECLIB.ODBC_IZIN
     WHERE USERID   = :pInfo.namaUser
       AND AKTIF    = 'Y';
  if wJml = 0;
     catatPenolakan(pInfo: 'Pengguna tidak terdaftar');
     return;
```

```
  endif;

  // 2. Apakah alamat IP termasuk yang diizinkan?
  exec sql
    SELECT COUNT(*) INTO :wJml
      FROM SECLIB.ODBC_IP
     WHERE USERID = :pInfo.namaUser
       AND IP_ADDR = :pInfo.alamatIP;
  if wJml = 0;
     catatPenolakan(pInfo: 'Alamat IP tidak dikenal');
     return;
  endif;

  // 3. Apakah dalam jam yang diizinkan?
  wJam = %subdt(%time(): *hours);
  if wJam >= 17 or wJam < 7;
     catatPenolakan(pInfo: 'Di luar jam yang diizinkan');
     return;
  endif;

  // Lolos semua pemeriksaan
  pIzinkan = '1';
  catatAkses(pInfo);
end-proc;
```

```
/* Pendaftaran */
ADDEXITPGM EXITPNT(QIBM_QZDA_INIT) FORMAT(ZDAI0100) +
           PGMNBR(1) PGM(SECLIB/ODBCEXIT) +
           TEXT('Kendali akses ODBC')

/* Wajib: server harus di-restart agar exit aktif */
ENDTCPSVR SERVER(*DATABASE)
STRTCPSVR SERVER(*DATABASE)
```

Tiga hal yang sering keliru saat memasang exit program.

1. Lupa me-restart server, sehingga exit terdaftar tetapi tidak aktif
2. Default `pIzinkan` diset `'1'`, sehingga kegagalan program berarti akses terbuka
3. Exit program sendiri gagal dan tidak ada penanganan, sehingga seluruh akses terblokir mendadak

Poin kedua adalah kesalahan desain keamanan yang paling berbahaya. **Default harus menolak.** Bila exit program gagal, sistem harus menutup, bukan membuka.

52.3 Menguji exit program dengan aman

```
/* Pasang dalam mode pencatatan saja lebih dulu */
/* Program mengembalikan '1' selalu, tetapi mencatat apa yang AKAN ditolak */
```

Jalankan dalam mode ini minimal dua minggu, analisis catatannya, dan baru aktifkan penolakan sesungguhnya setelah Anda yakin tidak ada akses sah yang terjaring.

```
-- Apa yang akan ditolak bila mode penegakan diaktifkan
SELECT ALASAN, COUNT(*) AS jumlah,
       COUNT(DISTINCT USERID) AS jml_pengguna,
       MIN(WAKTU) AS pertama, MAX(WAKTU) AS terakhir
  FROM SECLIB.ODBC_LOG
 WHERE HASIL = 'AKAN_DITOLAK'
 GROUP BY ALASAN
 ORDER BY jumlah DESC;
```

Bab 53. Enkripsi dan Sertifikat

Gambar 60 — Enkripsi, sertifikat, dan TLS pada IBM i

53.1 Digital Certificate Manager

DCM adalah tempat seluruh sertifikat sistem dikelola. Aksesnya lewat antarmuka web, tetapi pemantauannya sebaiknya lewat SQL.

```
-- Sertifikat yang mendekati kedaluwarsa
SELECT CERTIFICATE_LABEL,
       SUBJECT_COMMON_NAME,
       ISSUER_COMMON_NAME,
       VALIDITY_START, VALIDITY_END,
       DAYS(VALIDITY_END) - DAYS(CURRENT DATE) AS sisa_hari,
       PUBLIC_KEY_SIZE, SIGNATURE_ALGORITHM
  FROM QSYS2.CERTIFICATE_INFO(
         CERTIFICATE_STORE_PASSWORD => '*NOPWD')
 WHERE VALIDITY_END < CURRENT DATE + 120 DAYS
 ORDER BY VALIDITY_END;
```

Jadwalkan kueri ini mingguan dengan peringatan otomatis. Sertifikat kedaluwarsa yang tidak terpantau

adalah penyebab pemadaman layanan yang sepenuhnya bisa dicegah, dan selalu terjadi pada akhir pekan.

53.2 Mengeraskan TLS

```
/* Protokol yang diizinkan sistem */
DSPSYSVAL SYSVAL(QSSLPCL)
CHGSYSVAL SYSVAL(QSSLPCL) VALUE('*TLSV1.3 *TLSV1.2')

/* Cipher suite yang diizinkan */
DSPSYSVAL SYSVAL(QSSLCSL)

/* Kendali daftar cipher */
CHGSYSVAL SYSVAL(QSSLCSLCTL) VALUE('*USRDFN')
```

System value	Fungsi	Nilai untuk perbankan
`QSSLPCL`	Protokol TLS yang aktif	`*TLSV1.3 *TLSV1.2` saja
`QSSLCSL`	Daftar cipher suite	Hapus semua yang memakai RC4, 3DES, MD5
`QSSLCSLCTL`	Siapa mengendalikan daftar	`*USRDFN` agar Anda yang menentukan

SSLv3 dan TLS 1.0/1.1 harus dimatikan. Ini temuan audit yang hampir pasti muncul bila dibiarkan.

53.3 Enkripsi data saat diam

Tingkat	Cara	Cocok untuk
ASP	Enkripsi seluruh ASP	Perlindungan menyeluruh terhadap pencurian media
Kolom	`FIELDPROC` atau fungsi enkripsi	Data sangat sensitif seperti nomor kartu
Backup	Enkripsi pada BRMS atau perangkat tape	Media yang keluar dari data center

```
-- Enkripsi tingkat kolom dengan fungsi bawaan
SELECT DECRYPT_CHAR(NO_KARTU_ENC, :wKunci) AS no_kartu
  FROM DTALIB.KARTU
 WHERE ID_KARTU = :wId;

-- Menyimpan terenkripsi
UPDATE DTALIB.KARTU
   SET NO_KARTU_ENC = ENCRYPT_AES(:wNoKartu, :wKunci)
 WHERE ID_KARTU = :wId;
```

Manajemen kunci adalah masalah sesungguhnya, bukan algoritmanya. Kunci yang disimpan di dalam program, di data area tanpa perlindungan, atau di file konfigurasi, meniadakan seluruh manfaat enkripsi. Gunakan penyimpanan kunci terkelola, dan pastikan rotasi kunci sudah direncanakan sebelum enkripsi diaktifkan.

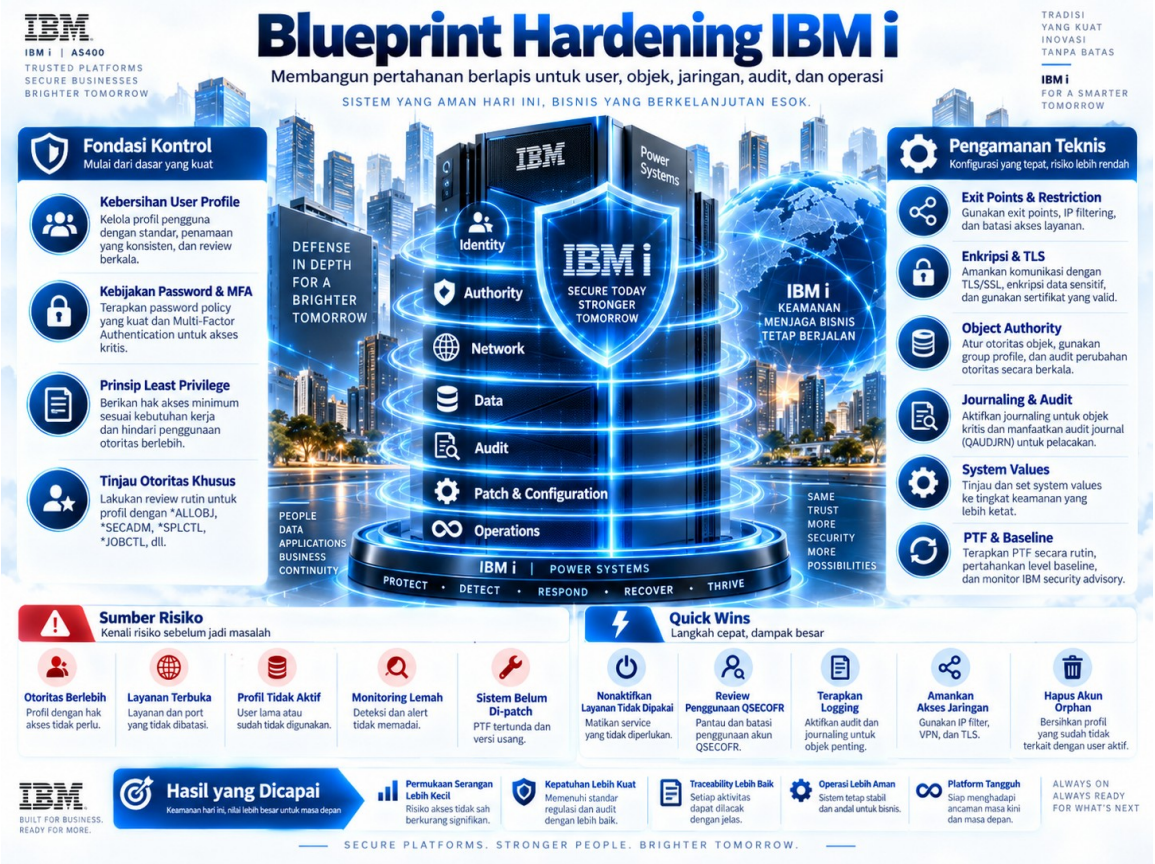
53.4 FIELDPROC — enkripsi transparan

```
-- Kolom dienkripsi otomatis saat ditulis, didekripsi saat dibaca
ALTER TABLE DTALIB.NASABAH
  ALTER COLUMN NIK
  SET FIELDPROC SECLIB.FPNIK;
```

Keunggulannya: program aplikasi tidak perlu diubah sama sekali. Kekurangannya: kolom yang dienkripsi tidak

bisa dipakai untuk pencarian rentang, dan index padanya hanya berguna untuk pencocokan persis.

⚙️ Bab 54. Pengerasan Sistem



Gambar 61 — Blueprint pengerasan sistem IBM i

54.1 Daftar periksa pengerasan

Layanan jaringan

- ☐ Matikan FTP bila tidak dipakai, atau paksa lewat TLS saja
- ☐ Matikan TFTP tanpa kecuali
- ☐ Matikan Telnet tanpa TLS; pakai SSH atau Telnet aman
- ☐ Matikan REXEC
- ☐ Batasi akses NetServer dan matikan share yang tidak perlu
- ☐ Tinjau seluruh port yang mendengarkan

```
-- Daftar port yang terbuka, bandingkan dengan daftar yang disetujui
SELECT LOCAL_PORT, LOCAL_PORT_NAME, PROTOCOL, JOB_NAME
  FROM QSYS2.NETSTAT_INFO
 WHERE TCP_STATE = 'LISTEN'
 ORDER BY LOCAL_PORT;
```

Profil dan wewenang

- ☐ Tidak ada profil dengan password default
- ☐ Profil IBM bawaan berstatus `*DISABLED` kecuali yang memang dipakai

- ☐ Jumlah profil `*ALLOBJ` di bawah lima, seluruhnya terkendali
- ☐ `*PUBLIC` pada library data diset `*EXCLUDE`
- ☐ Tidak ada objek yang dimiliki `QSECOFR` di library aplikasi
- ☐ Program adopsi wewenang ditinjau satu per satu

```
-- Profil IBM bawaan yang masih aktif
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       PREVIOUS_SIGNON
  FROM QSYS2.USER_INFO
 WHERE AUTHORIZATION_NAME LIKE 'Q%'
   AND STATUS = '*ENABLED'
 ORDER BY AUTHORIZATION_NAME;

-- Program yang mengadopsi wewenang pemiliknya
SELECT PROGRAM_LIBRARY, PROGRAM_NAME, PROGRAM_OWNER,
       USER_PROFILE, USE_ADOPTED_AUTHORITY
  FROM QSYS2.PROGRAM_INFO
 WHERE USER_PROFILE = '*OWNER'
   AND PROGRAM_LIBRARY NOT LIKE 'Q%'
 ORDER BY PROGRAM_OWNER;
```

Program yang mengadopsi wewenang pemiliknya adalah pintu belakang yang sah tetapi berbahaya. Bila pemiliknya punya `*ALLOBJ`, siapa pun yang bisa memanggil program itu untuk sesaat memiliki `*ALLOBJ` juga. Tinjau setiap satu dari mereka.

Audit

- ☐ `QAUDCTL` aktif dengan `*AUDLVL` dan `*OBJAUD`
- ☐ Audit journal receiver disimpan sebelum dihapus
- ☐ Objek sensitif diberi `CHGOBJAUD` tingkat `*ALL`
- ☐ Ekstraksi audit dikirim ke SIEM di luar sistem

```
/* Audit khusus untuk tabel sangat sensitif */
CHGOBJAUD OBJ(DTALIB/NASABAH) OBJTYPE(*FILE) OBJAUD(*ALL)
CHGOBJAUD OBJ(DTALIB/KARTU) OBJTYPE(*FILE) OBJAUD(*ALL)

/* Audit untuk aktivitas pengguna tertentu */
CHGUSRAUD USRPRF(VENDOR01) AUDLVL(*CMD *CREATE *DELETE *OBJMGT)
```

`CHGUSRAUD` layak dipakai untuk akun vendor dan konsultan: setiap perintah yang mereka jalankan tercatat, tanpa memperlambat pengguna lain.

54.2 Membangun laporan kepatuhan otomatis

```
CREATE OR REPLACE VIEW SECLIB.V_KEPATUHAN AS
SELECT 'Profil ALLOBJ aktif' AS kontrol,
       COUNT(*) AS nilai,
       CASE WHEN COUNT(*) <= 5 THEN 'LULUS' ELSE 'GAGAL' END AS hasil
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%' AND STATUS = '*ENABLED'
UNION ALL
SELECT 'Profil diam lebih 90 hari',
       COUNT(*),
       CASE WHEN COUNT(*) = 0 THEN 'LULUS' ELSE 'GAGAL' END
  FROM QSYS2.USER_INFO
 WHERE STATUS = '*ENABLED'
   AND PREVIOUS_SIGNON < CURRENT TIMESTAMP - 90 DAYS
```

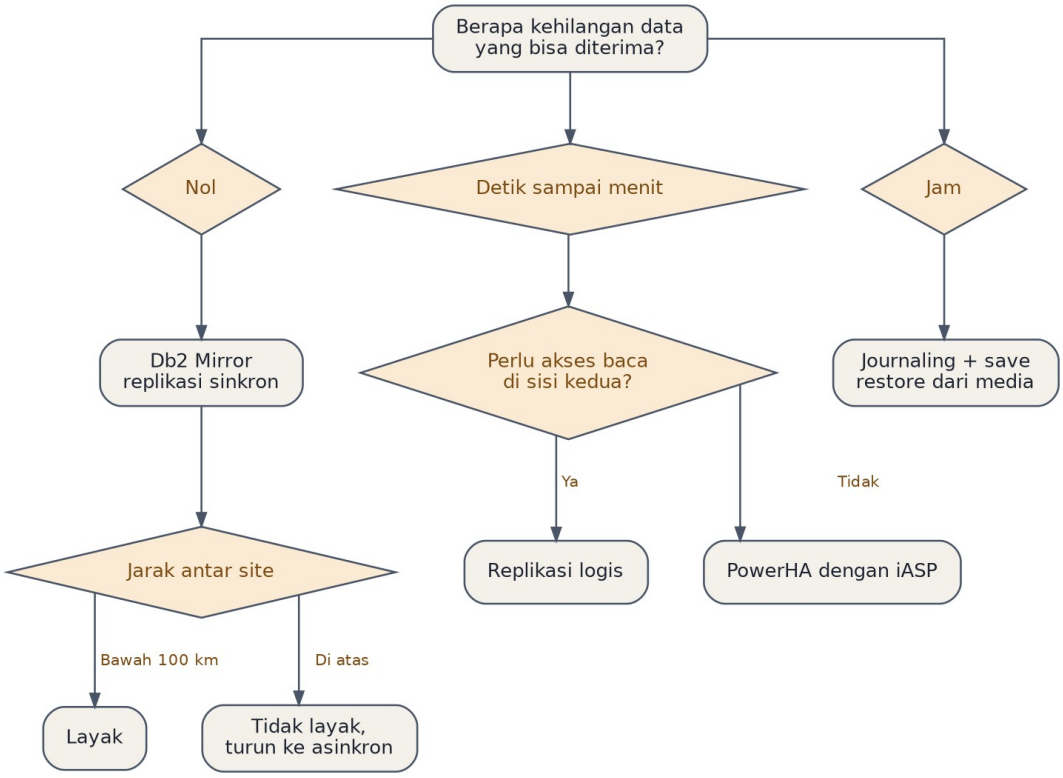
```
UNION ALL
SELECT 'Tingkat keamanan sistem',
       CURRENT_NUMERIC_VALUE,
       CASE WHEN CURRENT_NUMERIC_VALUE >= 40 THEN 'LULUS' ELSE 'GAGAL' END
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME = 'QSECURITY'
UNION ALL
SELECT 'Tabel produksi tanpa journaling',
       COUNT(*),
       CASE WHEN COUNT(*) = 0 THEN 'LULUS' ELSE 'GAGAL' END
  FROM TABLE(QSYS2.OBJECT_STATISTICS('DTALIB','*FILE')) o
 WHERE NOT EXISTS (
       SELECT 1 FROM QSYS2.JOURNALED_OBJECTS j
        WHERE j.JOURNALED_OBJECT = o.OBJNAME);

-- Dipanggil sebelum audit, atau dijadwalkan bulanan
SELECT * FROM SECLIB.V_KEPATUHAN;
```

View semacam ini mengubah persiapan audit dari pekerjaan dua minggu menjadi satu kueri. Kembangkan daftarnya mengikuti temuan audit sebelumnya — setiap temuan yang pernah muncul layak menjadi baris permanen di sini.

🛟 Bagian XV — High Availability dan Disaster Recovery Mendalam

🏗️ Bab 55. Memilih Arsitektur Ketersediaan

55.1 Pohon keputusan



Gambar 62

55.2 Perbandingan lengkap

Aspek	PowerHA iASP	Replikasi logis	Db2 Mirror
RPO	Mendekati nol	Detik sampai menit	Nol
RTO	Menit	Menit	Mendekati nol
Objek di luar iASP	Butuh admin domain	Direplikasi terpisah	Perlu penanganan
Sisi kedua bisa dibaca	Tidak saat aktif	Ya	Ya, keduanya aktif
Beban CPU	Rendah	Menengah sampai tinggi	Tinggi
Batas jarak	Tergantung mode mirror	Praktis tanpa batas	Ketat
Kompleksitas operasi	Menengah	Tinggi	Sangat tinggi
Biaya lisensi	Tinggi	Menengah	Sangat tinggi

55.3 Yang tidak ikut berpindah

Ini penyebab kegagalan uji DR yang paling sering, dan hampir selalu ditemukan terlambat.

Objek	Ada di iASP?	Cara sinkronisasi
Tabel dan program aplikasi	Ya	Otomatis
User profile	Tidak	Administrative domain
System value	Tidak	Administrative domain
Konfigurasi perangkat	Tidak	Manual atau skrip
Job scheduler entry	Tidak	Skrip sinkronisasi
Job description	Tergantung penempatan	Periksa satu per satu
Output queue	Tidak	Manual
Exit program terdaftar	Tidak	Skrip
Sertifikat DCM	Tidak	Ekspor dan impor manual
Objek di QGPL dan QUSRSYS	Tidak	Skrip sinkronisasi
Konfigurasi TCP/IP	Tidak	Manual, alamat berbeda

```
/* Administrative domain untuk objek di luar iASP */
ADDCADMRE CLUSTER(CLSTR1) CRG(ADMDOMAIN) +
          MONOBJ((*USRPRF QSYS BUDI001)) +
          AUTO(*YES)

/* Memantau apa yang dipantau domain */
WRKCADMRE CLUSTER(CLSTR1) CRG(ADMDOMAIN)
```

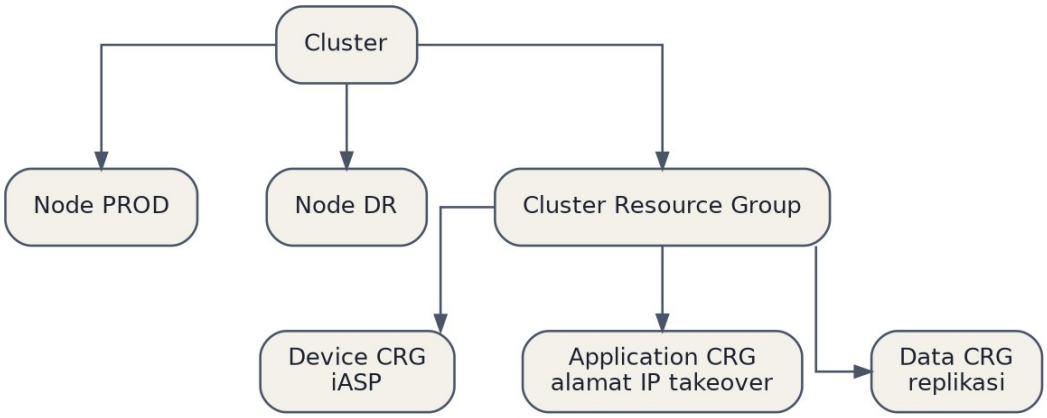
Daftar verifikasi sebelum uji DR. Bandingkan kedua sisi objek demi objek.

```
-- Jalankan di kedua partisi, bandingkan hasilnya
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       GROUP_PROFILE_NAME
  FROM QSYS2.USER_INFO
 WHERE AUTHORIZATION_NAME NOT LIKE 'Q%'
 ORDER BY AUTHORIZATION_NAME;

SELECT SYSTEM_VALUE_NAME, CURRENT_CHARACTER_VALUE,
       CURRENT_NUMERIC_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 ORDER BY SYSTEM_VALUE_NAME;

SELECT SCHEDULED_JOB_NAME, STATUS, FREQUENCY,
       SCHEDULE_DATE, SCHEDULE_TIME, COMMAND_STRING
  FROM QSYS2.SCHEDULED_JOB_INFO
 ORDER BY SCHEDULED_JOB_NAME;
```

Simpan keluaran ketiganya dari kedua sisi sebagai berkas, lalu bandingkan dengan `diff`. Setiap perbedaan yang tidak bisa dijelaskan adalah risiko DR yang terbuka.

⚙️ Bab 56. Operasi Cluster

56.1 Struktur cluster



Gambar 63

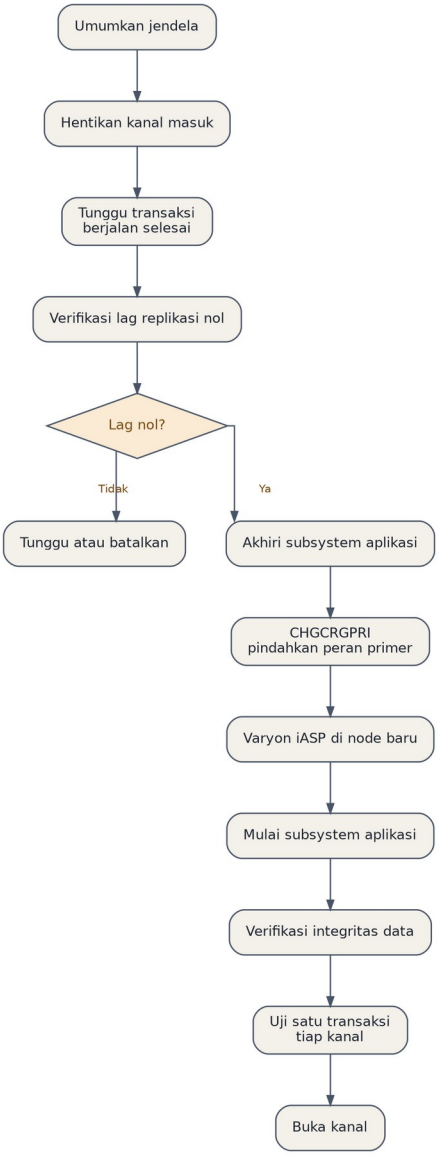
56.2 Perintah harian

```
/* Status cluster secara keseluruhan */
WRKCLU

/* Status tiap node */
DSPCLUINF CLUSTER(CLSTR1)

/* Status CRG */
WRKCRG CLUSTER(CLSTR1)
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA)

/* Status iASP */
WRKCFGSTS CFGTYPE(*DEV) CFGD(IASP01)
```

```
-- Informasi cluster lewat SQL, lebih mudah dipantau
SELECT NODE_ID, NODE_STATUS, DEVICE_DOMAIN,
       CURRENT_CLUSTER_VERSION
  FROM QSYS2.CLUSTER_INFO;

SELECT CLUSTER_RESOURCE_GROUP, CRG_TYPE, CRG_STATUS,
       PRIMARY_NODE, RECOVERY_DOMAIN
  FROM QSYS2.CLUSTER_RESOURCE_GROUP_INFO;
```

56.3 Switchover terencana

Berbeda dari failover: switchover dilakukan dengan sengaja, dengan kedua sisi sehat, biasanya untuk pemeliharaan.



Gambar 64

```
/* Verifikasi lag sebelum switchover */
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA) DETAIL(*FULL)

/* Eksekusi switchover */
CHGCRGPRI CLUSTER(CLSTR1) CRG(CRGDTA)

/* Verifikasi iASP sudah tersedia di node baru */
WRKCFGSTS CFGTYPE(*DEV) CFGD(IASP01)
VRYCFG CFGOBJ(IASP01) CFGTYPE(*DEV) STATUS(*ON)
```

Aturan yang tidak boleh dilanggar. Jangan pernah melakukan switchover saat lag replikasi belum nol. Anda akan kehilangan data sebesar lag itu, dan di lingkungan perbankan kehilangan lima menit transaksi berarti ratusan transaksi nasabah hilang tanpa jejak.

56.4 Failover tak terencana

Failover terjadi ketika node primer hilang. Sistem bisa melakukannya otomatis, tetapi di lingkungan perbankan **failover otomatis sering justru berbahaya**.

Skenario	Failover otomatis	Alasan
Node primer mati total	Bermanfaat	Jelas, tidak ambigu
Jaringan antar site putus	Berbahaya	Risiko split brain
Node melambat tetapi hidup	Berbahaya	Kedua sisi bisa aktif
Storage bermasalah	Bergantung	Perlu penilaian manusia

Split brain adalah kondisi kedua node mengira dirinya primer, keduanya menerima transaksi, dan datanya menyimpang. Pemulihannya menuntut rekonsiliasi manual yang bisa memakan berhari-hari. Untuk core banking, saya sarankan failover memerlukan konfirmasi manusia, dengan prosedur yang sudah dilatih agar konfirmasi itu bisa diberikan dalam hitungan menit.

56.5 Skrip verifikasi pasca-failover

```
PGM
   DCL VAR(&PESAN) TYPE(*CHAR) LEN(200)

   /* 1. iASP tersedia? */
   RTVCFGSTS CFGD(IASP01) CFGTYPE(*DEV) STSVAR(&STATUS)

   /* 2. Verifikasi jumlah record tabel kunci */
   RUNSQL SQL('INSERT INTO APPWRK.VERIFIKASI_DR +
       SELECT CURRENT TIMESTAMP, ''REKENING'', COUNT(*), SUM(SALDO) +
         FROM DTALIB.REKENING') COMMIT(*NONE)

   RUNSQL SQL('INSERT INTO APPWRK.VERIFIKASI_DR +
       SELECT CURRENT TIMESTAMP, ''GLTRX'', COUNT(*), SUM(DEBET) +
         FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE') COMMIT(*NONE)

   /* 3. Transaksi terakhir yang tercatat */
   RUNSQL SQL('INSERT INTO APPWRK.VERIFIKASI_DR +
       SELECT CURRENT TIMESTAMP, ''TRX_TERAKHIR'', 0, 0 +
         FROM SYSIBM.SYSDUMMY1') COMMIT(*NONE)

   /* 4. Subsystem aplikasi aktif? */
   /* 5. Server TCP yang dibutuhkan aktif? */

   SNDMSG MSG('Verifikasi pasca-failover selesai, periksa VERIFIKASI_DR') +
          TOUSR(SYSENG)
ENDPGM
```

```
-- Bandingkan dengan nilai terakhir sebelum insiden
SELECT NAMA_TABEL,
       JUMLAH_RECORD,
       TOTAL_NILAI,
       WAKTU
  FROM APPWRK.VERIFIKASI_DR
 ORDER BY WAKTU DESC
 FETCH FIRST 20 ROWS ONLY;
```

Angka-angka inilah yang menjadi dasar keputusan membuka layanan. Tanpa perbandingan kuantitatif,

"sistem sudah naik" hanyalah pernyataan tentang proses, bukan tentang data.

✍️ Bab 57. Merancang Latihan DR yang Bermakna

Gambar 65 — Latihan DR: switchover terencana dan failover darurat

57.1 Tingkatan latihan

Tingkat	Bentuk	Frekuensi	Gangguan operasi
1	Tinjauan dokumen di meja	Triwulanan	Tidak ada
2	Simulasi lisan berbasis skenario	Semesteran	Tidak ada
3	Switchover terencana	Tahunan	Terjadwal
4	Failover tanpa pemberitahuan waktu	Dua tahunan	Nyata

Sebagian besar institusi berhenti di tingkat 3. Tingkat 4 adalah satu-satunya yang benar-benar menguji apakah prosedur bekerja ketika orang tidak siap, dan itulah kondisi bencana yang sebenarnya.

57.2 Skenario yang layak diuji

1. Kehilangan total data center utama
2. Kerusakan data, bukan kehilangan perangkat — replikasi ikut menyalin kerusakan
3. Serangan ransomware pada sistem terhubung
4. Kehilangan personel kunci bersamaan dengan insiden teknis
5. Kegagalan saat EOD sedang berjalan

6. Kegagalan pada akhir bulan, saat beban tertinggi

Skenario kedua layak mendapat perhatian khusus. **Replikasi bukan backup.** Bila data rusak di sisi primer, kerusakan itu direplikasi ke DR dalam hitungan detik. Satu-satunya perlindungan adalah backup dengan titik waktu, dan itulah sebabnya journaling serta save terjadwal tetap wajib meskipun Anda sudah punya PowerHA.

57.3 Ukuran keberhasilan

Yang diukur	Cara mengukur	Target
RTO aktual	Dari deklarasi sampai layanan buka	Sesuai komitmen
RPO aktual	Selisih transaksi terakhir	Sesuai komitmen
Kelengkapan prosedur	Langkah yang harus diimprovisasi	Nol
Kesiapan personel	Langkah yang butuh bantuan luar tim	Nol
Integritas data	Selisih saldo kontrol	Nol
Kesiapan kanal	Kanal yang gagal terhubung	Nol

Catat setiap langkah yang harus diimprovisasi. Itulah daftar pekerjaan perbaikan dokumentasi Anda untuk tahun berikutnya, dan nilainya jauh melebihi laporan yang menyatakan semuanya berjalan lancar.

🏦 Bagian XVI — Modul Core Banking

Bagian ini membedah tiap modul dari sudut pandang System Engineer: apa karakteristik bebannya, di mana titik kegagalannya, dan apa yang harus Anda pantau. Seluruh nama tabel dan kolom adalah rekaan generik.

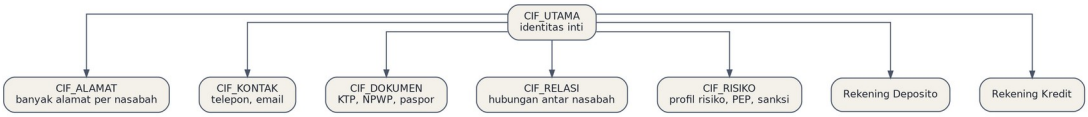
Bab 58. Customer Information File

CIF adalah pusat gravitasi seluruh sistem. Setiap modul lain menunjuk ke sini.

58.1 Karakteristik beban

Sifat	Nilai khas	Implikasi teknis
Rasio baca terhadap tulis	500:1 atau lebih	Optimasi berat di sisi baca
Pola akses	Pencarian berdasarkan nama, NIK, nomor rekening	Butuh banyak index
Ukuran record	Besar, banyak kolom opsional	Pertimbangkan pemisahan tabel
Pertumbuhan	Lambat tetapi tidak pernah menyusut	Data nasabah tutup tetap disimpan
Sensitivitas	Tertinggi di seluruh sistem	Wajib RCAC dan audit objek

58.2 Struktur yang lazim



Gambar 66

58.3 Titik kegagalan yang khas

Pencarian nama yang lambat. Pencarian nasabah berdasarkan nama parsial adalah keluhan nomor satu dari cabang.

```
-- Pola yang membunuh kinerja: wildcard di depan
SELECT * FROM DTALIB.CIF_UTAMA WHERE NAMA LIKE '%SANTOSO%';
```

Index biasa tidak bisa dipakai untuk pola ini. Solusi yang lazim:

```
-- 1. Kolom turunan yang sudah dinormalisasi
ALTER TABLE DTALIB.CIF_UTAMA
  ADD COLUMN NAMA_CARI VARCHAR(80)
  GENERATED ALWAYS AS (UPPER(TRANSLATE(NAMA, '     ', '.,-/\')));

CREATE INDEX DTALIB.CIF_NAMA_IX ON DTALIB.CIF_UTAMA (NAMA_CARI);

-- 2. Paksa pencarian berawalan, bukan mengandung
SELECT ID_NASABAH, NAMA, NIK
  FROM DTALIB.CIF_UTAMA
 WHERE NAMA_CARI LIKE 'SANTOSO%'
 FETCH FIRST 50 ROWS ONLY;

-- 3. Batasi hasil, selalu
```

Aturan yang perlu disepakati dengan bisnis: **pencarian nama wajib berawalan dan wajib dibatasi jumlah hasilnya.** Tanpa batas, satu pencarian "BUDI" bisa memindai seluruh tabel dan mengembalikan ratusan ribu baris.

Duplikasi nasabah. Satu orang terdaftar berkali-kali dengan ejaan nama berbeda.

```
-- Deteksi kandidat duplikat berdasarkan NIK
SELECT NIK, COUNT(*) AS jumlah,
       LISTAGG(ID_NASABAH, ', ') AS id_terkait,
       LISTAGG(NAMA, ' | ') AS variasi_nama
  FROM DTALIB.CIF_UTAMA
 WHERE NIK IS NOT NULL AND NIK <> ''
 GROUP BY NIK
HAVING COUNT(*) > 1
 ORDER BY jumlah DESC;

-- Kandidat duplikat berdasarkan kemiripan tanggal lahir dan nama
SELECT a.ID_NASABAH, a.NAMA, b.ID_NASABAH, b.NAMA, a.TGL_LAHIR
  FROM DTALIB.CIF_UTAMA a
  JOIN DTALIB.CIF_UTAMA b
    ON a.TGL_LAHIR = b.TGL_LAHIR
   AND a.ID_NASABAH < b.ID_NASABAH
   AND SUBSTR(a.NAMA_CARI,1,6) = SUBSTR(b.NAMA_CARI,1,6)
 WHERE a.TGL_LAHIR IS NOT NULL
 FETCH FIRST 200 ROWS ONLY;
```

Penggabungan nasabah duplikat adalah operasi berisiko tinggi yang menyentuh seluruh modul hilir. Perlakukan sebagai perubahan terkendali, bukan perbaikan data rutin.

58.4 Perlindungan data nasabah

```
-- Masking NIK kecuali untuk peran tertentu
CREATE MASK DTALIB.MASK_NIK ON DTALIB.CIF_UTAMA
  FOR COLUMN NIK RETURN
    CASE
      WHEN VERIFY_GROUP_FOR_USER(SESSION_USER,'GRPCS','GRPCOMPLIANCE') = 1
        THEN NIK
      ELSE SUBSTR(NIK,1,6) || 'XXXXXXXX' || SUBSTR(NIK,15,2)
    END
  ENABLE;

-- Pembatasan baris: petugas cabang hanya melihat nasabah cabangnya
CREATE PERMISSION DTALIB.PERM_CIF ON DTALIB.CIF_UTAMA
  FOR ROWS WHERE
    VERIFY_GROUP_FOR_USER(SESSION_USER,'GRPPUSAT') = 1
    OR KD_CABANG IN (SELECT KD_CABANG FROM APPPRM.PEGAWAI_CABANG
                     WHERE USERID = SESSION_USER)
  ENFORCED FOR ALL ACCESS
  ENABLE;

ALTER TABLE DTALIB.CIF_UTAMA ACTIVATE ROW ACCESS CONTROL;
ALTER TABLE DTALIB.CIF_UTAMA ACTIVATE COLUMN ACCESS CONTROL;
```

```
/* Audit objek tingkat penuh untuk tabel nasabah */
CHGOBJAUD OBJ(DTALIB/CIF_UTAMA) OBJTYPE(*FILE) OBJAUD(*ALL)
```

58.5 Yang dipantau System Engineer

```
-- Siapa membaca data nasabah dalam jumlah besar
SELECT ENTRY_TIMESTAMP, USER_NAME, JOB_NAME,
       OBJECT_NAME, ENTRY_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'ZR',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 24 HOURS))
 WHERE OBJECT_NAME IN ('CIF_UTAMA','CIF_DOKUMEN')
 ORDER BY ENTRY_TIMESTAMP DESC;
```

Lonjakan pembacaan CIF di luar jam layanan, atau dari profil yang biasanya tidak menyentuhnya, adalah sinyal yang harus diselidiki, bukan diabaikan sebagai anomali statistik.

🏦 Bab 59. Modul Deposito

59.1 Karakteristik beban

Sifat	Implikasi
Volume transaksi tertinggi di sistem	Titik kontensi utama
Saldo diperbarui setiap transaksi	Kunci record sering
Semua kanal menyentuhnya	Kegagalan terlihat seluruh nasabah
Akrual bunga harian atau bulanan	Fase EOD terberat

59.2 Anatomi posting transaksi

```
sequenceDiagram
    participant K as Kanal
    participant V as Validasi
    participant R as Rekening
    participant G as General Ledger
    participant J as Journal
    K->>V: Permintaan debet
    V->>V: Periksa status rekening
    V->>R: CHAIN + kunci record
    R-->>V: Saldo saat ini
    V->>V: Periksa kecukupan saldo
    V->>R: UPDATE saldo
    R->>J: Entri journal otomatis
    V->>G: Tulis pasangan debet kredit
    V->>K: Konfirmasi + commit
```

59.3 Kontensi saldo — masalah teknis paling nyata

Rekening dengan volume transaksi sangat tinggi, misalnya rekening perantara atau rekening penampungan biller, menjadi titik panas. Setiap transaksi mengunci baris yang sama.

```
-- Menemukan rekening yang menjadi titik panas
SELECT NO_REK, COUNT(*) AS jml_transaksi
  FROM DTALIB.TRXHIST
 WHERE TGL_TRX = CURRENT DATE
 GROUP BY NO_REK
 ORDER BY jml_transaksi DESC
 FETCH FIRST 20 ROWS ONLY;

-- Kunci yang sedang terjadi pada rekening tersebut
SELECT JOB_NAME, LOCK_STATE, RRN
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'DTALIB' AND TABLE_NAME = 'REKENING';
```

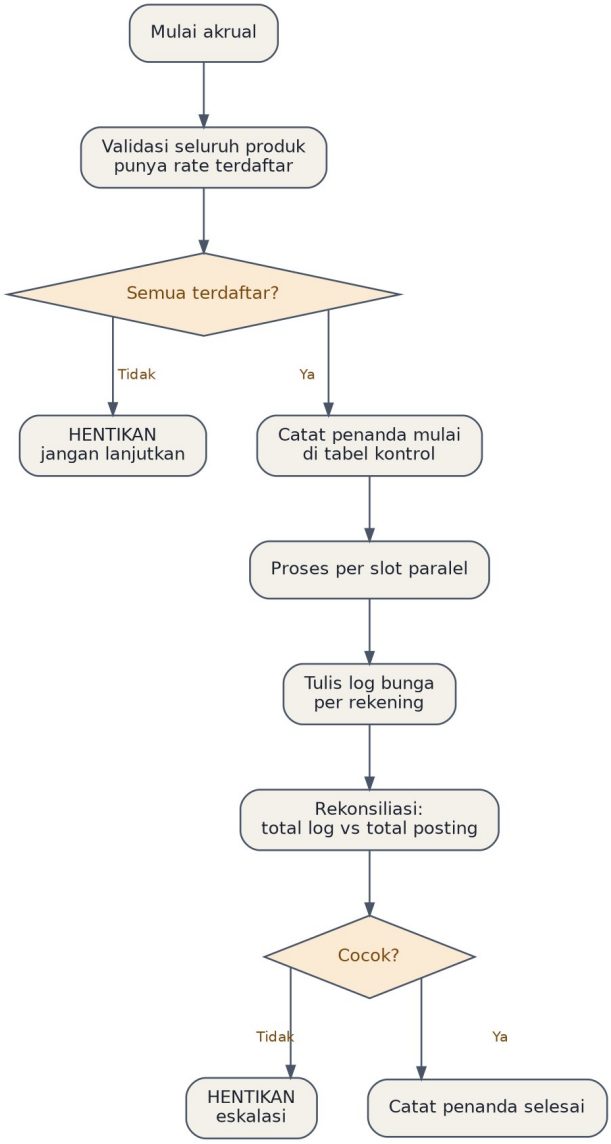
Pola pemecahan yang lazim. Alih-alih memperbarui satu baris saldo, tulis mutasi ke tabel terpisah dan hitung saldo sebagai agregat, atau pecah saldo menjadi beberapa baris sub-saldo yang dijumlahkan.

```
-- Pola sub-saldo untuk rekening bervolume tinggi
CREATE TABLE DTALIB.SALDO_PECAH (
    NO_REK    CHAR(16)      NOT NULL,
    SLOT      SMALLINT      NOT NULL,
    SALDO     DECIMAL(17,2) NOT NULL DEFAULT 0,
    PRIMARY KEY (NO_REK, SLOT)
);

-- Transaksi menulis ke slot berdasarkan hash, mengurangi kontensi
UPDATE DTALIB.SALDO_PECAH
   SET SALDO = SALDO + :wJumlah
 WHERE NO_REK = :wNoRek
   AND SLOT = MOD(:wNoUrut, 16);

-- Saldo sesungguhnya adalah jumlah seluruh slot
SELECT SUM(SALDO) FROM DTALIB.SALDO_PECAH WHERE NO_REK = :wNoRek;
```

Ini menukar kesederhanaan dengan throughput. Pakai hanya untuk rekening yang terbukti menjadi titik panas, bukan sebagai pola umum.

59.4 Fase akrual bunga

Inilah bagian EOD yang paling berat dan paling berbahaya.



Gambar 67

Validasi pra-akrual yang wajib ada.

```
-- Produk aktif yang belum punya rate berlaku hari ini
SELECT p.KD_PRODUK, p.NM_PRODUK
  FROM APPPRM.PRODUK p
 WHERE p.STATUS = 'A'
   AND NOT EXISTS (
       SELECT 1 FROM APPPRM.PRODUK_RATE r
        WHERE r.KD_PRODUK = p.KD_PRODUK
          AND r.TGL_BERLAKU <= CURRENT DATE
          AND (r.TGL_AKHIR IS NULL OR r.TGL_AKHIR >= CURRENT DATE));
```

Kueri ini harus mengembalikan nol baris sebelum akrual dimulai. Bila tidak, hentikan — inilah penyebab kasus "bunga tidak terbukukan" di Bab 42.

Rekonsiliasi pasca-akrual.

```
-- Total yang dicatat di log harus sama dengan perubahan saldo
WITH log_bunga AS (
  SELECT SUM(LOG_BUNGA) AS total_log, COUNT(*) AS jml_log
    FROM DTALIB.BUNGALOG WHERE LOG_TGL = CURRENT DATE
),
posting_gl AS (
  SELECT SUM(KREDIT) AS total_gl, COUNT(*) AS jml_gl
    FROM DTALIB.GLTRX
   WHERE TGL_BUKU = CURRENT DATE AND KD_GL = 'BUNGA_DEP'
)
SELECT l.total_log, p.total_gl,
       l.total_log - p.total_gl AS selisih,
       l.jml_log, p.jml_gl
  FROM log_bunga l, posting_gl p;
```

Selisih bukan nol berarti akrual tidak konsisten. Jangan melanjutkan ke fase berikutnya.

59.5 Idempotensi

Pertanyaan yang harus bisa dijawab untuk setiap program akrual: **apa yang terjadi bila ini dijalankan dua kali?**

```
-- Pola penanda idempoten
CREATE TABLE APPWRK.AKRUAL_PENANDA (
    TGL_BUKU  DATE     NOT NULL,
    KD_FASE   CHAR(10) NOT NULL,
    STATUS    CHAR(1)  NOT NULL,
    JAM       TIMESTAMP,
    JML_REK   BIGINT,
    TOTAL     DECIMAL(19,2),
    PRIMARY KEY (TGL_BUKU, KD_FASE)
);
```

```
**free
// Di awal program akrual
dcl-s wSudah int(5);

exec sql
  SELECT COUNT(*) INTO :wSudah
    FROM APPWRK.AKRUAL_PENANDA
   WHERE TGL_BUKU = :wTglBuku
     AND KD_FASE  = 'AKRUAL'
     AND STATUS   = 'S';

if wSudah > 0;
   kirimEscape('Akrual untuk tanggal ini SUDAH dijalankan. ' +
               'Rerun tidak diizinkan tanpa reversal.');
endif;
```

Enam baris ini mencegah kelas kesalahan yang paling mahal di perbankan. Pastikan setiap program fase akrual memilikinya.

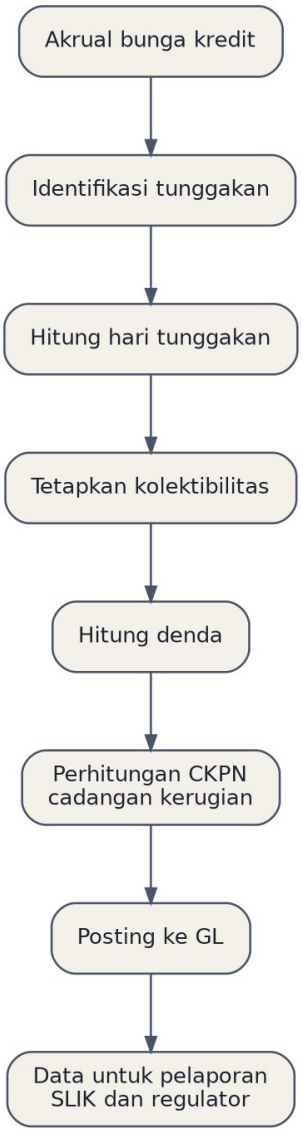
🏦 Bab 60. Modul Kredit

60.1 Yang membedakan dari deposito

Aspek	Deposito	Kredit
Volume transaksi harian	Sangat tinggi	Rendah
Kompleksitas perhitungan	Sedang	Tinggi
Jumlah tabel terkait	Sedikit	Banyak
Perhitungan EOD	Akrual bunga	Akrual, kolektibilitas, CKPN
Sensitivitas regulatoris	Menengah	Tinggi

Modul kredit memproses jauh lebih sedikit record tetapi dengan perhitungan yang jauh lebih rumit. Konsekuensinya: bebannya CPU-intensif, bukan I/O-intensif, dan pola optimasinya berbeda.

60.2 Rantai perhitungan EOD kredit



Gambar 68

Setiap anak panah adalah ketergantungan keras. Menjalankan perhitungan kolektibilitas sebelum hari tunggakan selesai dihitung menghasilkan klasifikasi yang salah, dan klasifikasi yang salah masuk ke laporan regulator.

60.3 Klasifikasi kolektibilitas

```
-- Contoh logika klasifikasi berdasarkan hari tunggakan
UPDATE DTALIB.KREDIT k
   SET KOLEKTIBILITAS =
       CASE
         WHEN k.HARI_TUNGGAK = 0               THEN 1
         WHEN k.HARI_TUNGGAK BETWEEN 1 AND 90  THEN 2
         WHEN k.HARI_TUNGGAK BETWEEN 91 AND 120 THEN 3
         WHEN k.HARI_TUNGGAK BETWEEN 121 AND 180 THEN 4
         ELSE 5
       END,
       TGL_UPDATE_KOL = CURRENT DATE
 WHERE k.STATUS = 'A';

-- Verifikasi: distribusi kolektibilitas, bandingkan dengan kemarin
SELECT KOLEKTIBILITAS,
       COUNT(*) AS jml_rekening,
       SUM(BAKI_DEBET) AS total_baki
  FROM DTALIB.KREDIT
 WHERE STATUS = 'A'
 GROUP BY KOLEKTIBILITAS
 ORDER BY KOLEKTIBILITAS;
```

Pemeriksaan yang wajib. Perpindahan kolektibilitas massal dalam satu malam hampir selalu berarti kesalahan perhitungan, bukan perubahan nyata kualitas portofolio.

```
-- Rekening yang kolektibilitasnya melompat lebih dari satu tingkat
SELECT k.NO_KREDIT, h.KOLEKTIBILITAS AS kemarin,
       k.KOLEKTIBILITAS AS hari_ini,
       k.HARI_TUNGGAK, k.BAKI_DEBET
  FROM DTALIB.KREDIT k
  JOIN DTALIB.KREDIT_HIST h
    ON k.NO_KREDIT = h.NO_KREDIT
   AND h.TGL_BUKU = CURRENT DATE - 1 DAY
 WHERE ABS(k.KOLEKTIBILITAS - h.KOLEKTIBILITAS) > 1
 ORDER BY k.BAKI_DEBET DESC;
```

Bila kueri ini mengembalikan ratusan baris, hentikan EOD dan selidiki sebelum data masuk ke pelaporan.

60.4 Kinerja perhitungan kredit

Karena bebannya CPU-intensif, paralelisasi sangat efektif di sini — dan aman, karena perhitungan per rekening kredit umumnya independen.

```
-- Pembagian kerja berdasarkan modulo
SELECT NO_KREDIT, BAKI_DEBET, KD_PRODUK, TGL_JATUH_TEMPO
  FROM DTALIB.KREDIT
 WHERE STATUS = 'A'
   AND MOD(INT(SUBSTR(NO_KREDIT, 10, 5)), :pTotalJob) = :pNomorJob - 1
  FOR UPDATE OF BUNGA_AKRUAL, DENDA;
```

Pengecualian: perhitungan CKPN yang memakai pendekatan portofolio kolektif tidak bisa diparalelkan per rekening, karena hasilnya bergantung pada agregat seluruh kelompok.

60.5 Jejak audit untuk kredit

Regulator menanyakan riwayat, bukan keadaan saat ini. Simpan snapshot harian.

```
-- Snapshot harian setelah EOD, sebelum date roll
INSERT INTO DTALIB.KREDIT_HIST
SELECT CURRENT DATE, NO_KREDIT, ID_NASABAH, BAKI_DEBET,
       KOLEKTIBILITAS, HARI_TUNGGAK, BUNGA_AKRUAL,
       DENDA, CKPN, STATUS
  FROM DTALIB.KREDIT
 WHERE STATUS IN ('A','R');
```

Tabel riwayat ini tumbuh cepat. Terapkan partisi atau kebijakan arsip sejak awal, bukan setelah ia mencapai ratusan juta baris.

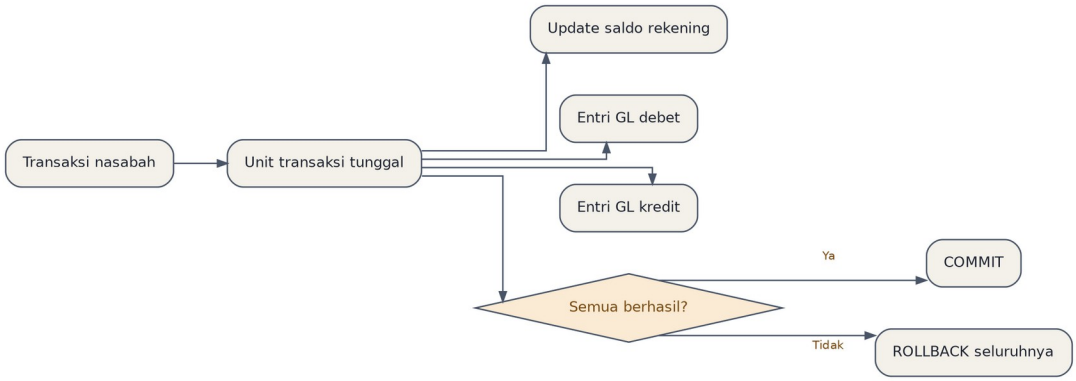
```
-- Proyeksi pertumbuhan tabel riwayat
SELECT COUNT(*) AS baris_per_hari,
       COUNT(*) * 365 AS proyeksi_setahun,
       COUNT(*) * 365 * 250 / 1048576 AS estimasi_mb_setahun
  FROM DTALIB.KREDIT WHERE STATUS IN ('A','R');
```

🏦 Bab 61. General Ledger

GL adalah tempat seluruh modul bertemu. Bila GL tidak seimbang, tidak ada satu pun angka di bank yang bisa dipercaya.

61.1 Prinsip yang menentukan desain teknis

Setiap transaksi menghasilkan sepasang entri yang jumlahnya harus sama. Konsekuensi teknisnya: **entri GL harus ditulis dalam satu unit transaksi yang sama dengan perubahan saldo yang memicunya.** Menulis saldo di satu commit dan GL di commit berikutnya adalah cacat desain, bukan optimasi.



Gambar 69

61.2 Pemeriksaan keseimbangan

```
-- Keseimbangan harian secara keseluruhan
SELECT TGL_BUKU,
       SUM(DEBET) AS total_debet,
       SUM(KREDIT) AS total_kredit,
       SUM(DEBET) - SUM(KREDIT) AS selisih,
       COUNT(*) AS jml_entri
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU = CURRENT DATE
 GROUP BY TGL_BUKU;

-- Per nomor referensi: tiap transaksi harus berpasangan
SELECT NO_REF, COUNT(*) AS jml_leg,
```

```
       SUM(DEBET) AS d, SUM(KREDIT) AS k,
       SUM(DEBET) - SUM(KREDIT) AS selisih
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU = CURRENT DATE
 GROUP BY NO_REF
HAVING SUM(DEBET) <> SUM(KREDIT)
 ORDER BY ABS(SUM(DEBET) - SUM(KREDIT)) DESC;

-- Rekonsiliasi saldo buku besar terhadap buku pembantu
WITH buku_besar AS (
  SELECT KD_GL, SUM(KREDIT) - SUM(DEBET) AS saldo_gl
    FROM DTALIB.GLTRX
   WHERE TGL_BUKU <= CURRENT DATE
   GROUP BY KD_GL
),
buku_pembantu AS (
  SELECT KD_GL_TERKAIT AS KD_GL, SUM(SALDO) AS saldo_sub
    FROM DTALIB.REKENING
   WHERE STATUS = 'A'
   GROUP BY KD_GL_TERKAIT
)
SELECT b.KD_GL, b.saldo_gl, p.saldo_sub,
       b.saldo_gl - COALESCE(p.saldo_sub, 0) AS selisih
  FROM buku_besar b
  LEFT JOIN buku_pembantu p ON b.KD_GL = p.KD_GL
 WHERE b.saldo_gl <> COALESCE(p.saldo_sub, 0);
```

Kueri ketiga adalah yang paling penting. Selisih antara buku besar dan buku pembantu berarti ada transaksi yang memperbarui salah satu tanpa memperbarui yang lain — hampir selalu akibat commit yang tidak terkoordinasi atau proses koreksi manual.

61.3 Beban teknis GL

Tabel GL adalah tabel yang tumbuh paling cepat di seluruh sistem, karena setiap transaksi menulis minimal dua baris ke sana.

Beban harian	Baris GL per hari	Setahun
1 juta transaksi	2 juta	730 juta
5 juta transaksi	10 juta	3,65 miliar

Strategi yang perlu disiapkan sejak awal:

1. **Pemisahan tabel aktif dan arsip.** Tabel aktif hanya menyimpan periode berjalan
2. **Index minimal.** Setiap index memperlambat penulisan, dan GL didominasi penulisan
3. **Tabel ringkasan.** Saldo GL per akun per hari dihitung sekali, bukan diagregasi ulang tiap kali
4. **Reorganisasi terjadwal.** Setelah pemindahan ke arsip

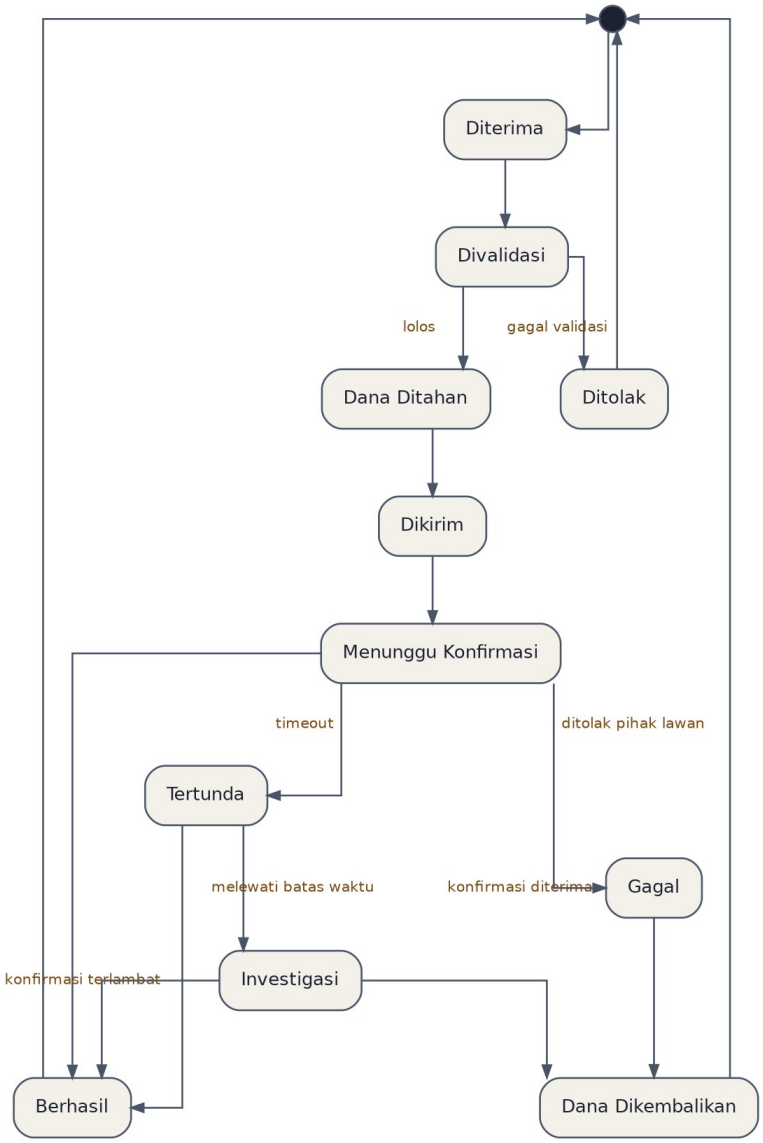
```
-- Tabel ringkasan saldo GL harian
CREATE TABLE DTALIB.GL_SALDO_HARIAN (
    TGL_BUKU  DATE          NOT NULL,
    KD_GL     CHAR(10)      NOT NULL,
    SALDO_AWAL DECIMAL(19,2) NOT NULL,
    MUTASI_D  DECIMAL(19,2) NOT NULL,
    MUTASI_K  DECIMAL(19,2) NOT NULL,
    SALDO_AKHIR DECIMAL(19,2) NOT NULL,
    PRIMARY KEY (TGL_BUKU, KD_GL)
);

-- Diisi sebagai bagian dari fase GL balancing
INSERT INTO DTALIB.GL_SALDO_HARIAN
SELECT CURRENT DATE, t.KD_GL,
       COALESCE(y.SALDO_AKHIR, 0),
       SUM(t.DEBET), SUM(t.KREDIT),
       COALESCE(y.SALDO_AKHIR, 0) + SUM(t.KREDIT) - SUM(t.DEBET)
  FROM DTALIB.GLTRX t
  LEFT JOIN DTALIB.GL_SALDO_HARIAN y
    ON y.KD_GL = t.KD_GL AND y.TGL_BUKU = CURRENT DATE - 1 DAY
 WHERE t.TGL_BUKU = CURRENT DATE
 GROUP BY t.KD_GL, y.SALDO_AKHIR;
```

Setelah tabel ini ada, neraca harian dijawab dari satu tabel kecil, bukan dari agregasi miliaran baris.

🏦 Bab 62. Remittance dan Sistem Pembayaran

62.1 Yang membedakan dari transaksi internal

Transaksi internal selesai dalam satu unit transaksi. Transaksi antar bank tidak — ia melibatkan pihak luar yang bisa lambat, gagal, atau menjawab belakangan.



Gambar 70

Status `Tertunda` adalah sumber masalah operasional terbesar. Dana sudah keluar dari rekening nasabah tetapi belum dipastikan sampai. Setiap transaksi dalam status ini menuntut penyelesaian, dan penumpukannya adalah keluhan nasabah yang menunggu.

62.2 Yang dipantau System Engineer

```
-- Transaksi yang menggantung lebih dari ambang waktu
SELECT KD_KANAL, STATUS,
       COUNT(*) AS jumlah,
       SUM(NOMINAL) AS total_nominal,
       MIN(WAKTU_KIRIM) AS tertua,
       TIMESTAMPDIFF(4, CHAR(CURRENT TIMESTAMP - MIN(WAKTU_KIRIM)))
         AS menit_tertua
  FROM DTALIB.REMIT_TRX
 WHERE STATUS IN ('PENDING','SENT')
   AND WAKTU_KIRIM < CURRENT TIMESTAMP - 30 MINUTES
 GROUP BY KD_KANAL, STATUS
```

```
 ORDER BY menit_tertua DESC;

-- Tingkat kegagalan per kanal, per jam
SELECT KD_KANAL, HOUR(WAKTU_KIRIM) AS jam,
       COUNT(*) AS total,
       SUM(CASE WHEN STATUS = 'FAILED' THEN 1 ELSE 0 END) AS gagal,
       DECIMAL(SUM(CASE WHEN STATUS = 'FAILED' THEN 1 ELSE 0 END)
               * 100.0 / COUNT(*), 5, 2) AS persen_gagal
  FROM DTALIB.REMIT_TRX
 WHERE TGL_TRX = CURRENT DATE
 GROUP BY KD_KANAL, HOUR(WAKTU_KIRIM)
HAVING COUNT(*) > 50
 ORDER BY persen_gagal DESC;
```

Lonjakan tingkat kegagalan pada satu kanal biasanya berarti masalah di sisi lawan atau di jaringan, bukan di IBM i. Tetapi Andalah yang pertama melihatnya, dan kecepatan Anda memberi tahu pihak yang tepat menentukan berapa lama gangguan berlangsung.

62.3 Rekonsiliasi antar sistem

```
-- Transaksi yang tercatat di core tetapi tidak di file settlement
SELECT c.NO_REF, c.NOMINAL, c.WAKTU_KIRIM, c.STATUS
  FROM DTALIB.REMIT_TRX c
 WHERE c.TGL_TRX = CURRENT DATE
   AND c.STATUS = 'SUCCESS'
   AND NOT EXISTS (
       SELECT 1 FROM APPWRK.SETTLEMENT_FILE s
        WHERE s.NO_REF = c.NO_REF);

-- Sebaliknya: ada di settlement tetapi tidak di core
SELECT s.NO_REF, s.NOMINAL
  FROM APPWRK.SETTLEMENT_FILE s
 WHERE NOT EXISTS (
       SELECT 1 FROM DTALIB.REMIT_TRX c
        WHERE c.NO_REF = s.NO_REF);
```

Kedua arah harus diperiksa. Arah kedua lebih berbahaya: dana keluar dari bank tanpa catatan internal.

62.4 Ketahanan terhadap kegagalan pihak lawan

Pola	Kegunaan
Timeout eksplisit di setiap panggilan keluar	Mencegah job menggantung selamanya
Idempotency key per transaksi	Mencegah dobel kirim saat retry
Circuit breaker	Berhenti mencoba saat lawan jelas mati
Antrean dengan retry bertingkat	Menahan lonjakan tanpa kehilangan transaksi
Rekonsiliasi terjadwal	Menangkap yang lolos dari semua di atas

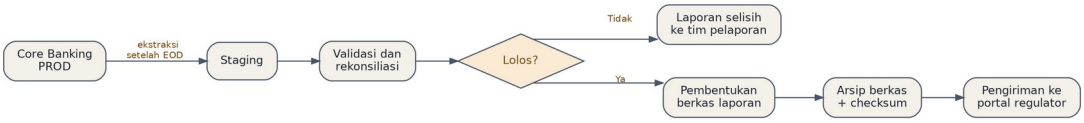
```
**free
// Idempotency: jangan kirim ulang transaksi yang sudah terkirim
dcl-s wAda int(5);

exec sql
  SELECT COUNT(*) INTO :wAda
    FROM DTALIB.REMIT_TRX
   WHERE IDEMPOTENCY_KEY = :wKey
     AND STATUS IN ('SENT','SUCCESS');

if wAda > 0;
   // sudah pernah dikirim, kembalikan status yang ada
   return;
endif;
```

Bab 63. Pelaporan Regulator

63.1 Sifat khas beban pelaporan

Sifat	Implikasi teknis
Tenggat keras, tidak bisa ditawar	Kegagalan berarti sanksi
Agregasi lintas seluruh modul	Kueri berat, menyentuh banyak tabel
Harus konsisten dengan laporan sebelumnya	Butuh data historis, bukan keadaan saat ini
Format ditentukan pihak luar	Perubahan format datang dari luar kendali Anda
Diperiksa oleh pihak ketiga	Selisih apa pun harus bisa dijelaskan

63.2 Arsitektur yang disarankan



Gambar 71

Prinsip yang menyelamatkan Anda. Pembentukan laporan tidak boleh berjalan langsung di atas tabel produksi. Ekstraksi ke staging memberi tiga hal: snapshot yang konsisten, ruang untuk mengulang tanpa mengganggu produksi, dan bukti apa yang dilaporkan bila kemudian dipertanyakan.

63.3 Arsip laporan sebagai bukti

```
CREATE TABLE APPWRK.LAPORAN_ARSIP (
    ID_LAPORAN   BIGINT GENERATED ALWAYS AS IDENTITY,
    KD_LAPORAN   VARCHAR(30)   NOT NULL,
    PERIODE      DATE          NOT NULL,
    VERSI        SMALLINT      NOT NULL DEFAULT 1,
    WAKTU_BENTUK TIMESTAMP     NOT NULL,
    DIBENTUK_OLEH VARCHAR(30)  NOT NULL,
    JML_RECORD   BIGINT,
    TOTAL_NILAI  DECIMAL(21,2),
    CHECKSUM     VARCHAR(64),
    PATH_BERKAS  VARCHAR(255),
    STATUS_KIRIM CHAR(1),
    PRIMARY KEY (ID_LAPORAN)
);
```

Kolom `VERSI` penting. Laporan yang diperbaiki dan dikirim ulang harus tersimpan sebagai versi baru, bukan menimpa versi lama. Ketika regulator bertanya mengapa angkanya berubah, Anda bisa menunjukkan keduanya.

```
-- Riwayat pengiriman satu jenis laporan
SELECT KD_LAPORAN, PERIODE, VERSI, WAKTU_BENTUK,
       JML_RECORD, TOTAL_NILAI, STATUS_KIRIM
  FROM APPWRK.LAPORAN_ARSIP
 WHERE KD_LAPORAN = 'LAP_KREDIT_BULANAN'
 ORDER BY PERIODE DESC, VERSI DESC;

-- Selisih antar versi, untuk menjelaskan perubahan
SELECT a.PERIODE, a.VERSI, a.TOTAL_NILAI,
       b.VERSI AS versi_sebelum, b.TOTAL_NILAI AS nilai_sebelum,
       a.TOTAL_NILAI - b.TOTAL_NILAI AS selisih
  FROM APPWRK.LAPORAN_ARSIP a
  JOIN APPWRK.LAPORAN_ARSIP b
    ON a.KD_LAPORAN = b.KD_LAPORAN
   AND a.PERIODE = b.PERIODE
   AND a.VERSI = b.VERSI + 1
 WHERE a.KD_LAPORAN = 'LAP_KREDIT_BULANAN';
```

63.4 Validasi sebelum kirim

```
-- Kerangka validasi berlapis
CREATE OR REPLACE VIEW APPWRK.V_VALIDASI_LAPORAN AS
SELECT 'Jumlah record nol' AS pemeriksaan,
       CASE WHEN COUNT(*) = 0 THEN 'GAGAL' ELSE 'LULUS' END AS hasil,
       COUNT(*) AS nilai
  FROM APPWRK.STG_LAPORAN
UNION ALL
SELECT 'Nilai negatif tidak wajar',
       CASE WHEN COUNT(*) = 0 THEN 'LULUS' ELSE 'GAGAL' END,
       COUNT(*)
  FROM APPWRK.STG_LAPORAN WHERE NOMINAL < 0
UNION ALL
SELECT 'Kolom wajib kosong',
       CASE WHEN COUNT(*) = 0 THEN 'LULUS' ELSE 'GAGAL' END,
       COUNT(*)
  FROM APPWRK.STG_LAPORAN
 WHERE ID_NASABAH IS NULL OR KD_PRODUK IS NULL
UNION ALL
SELECT 'Selisih terhadap GL',
       CASE WHEN ABS(COALESCE(SUM(s.NOMINAL),0)
                   - COALESCE((SELECT SUM(SALDO_AKHIR)
                                FROM DTALIB.GL_SALDO_HARIAN
                               WHERE TGL_BUKU = CURRENT DATE
                                 AND KD_GL LIKE 'KREDIT%'),0)) < 1
            THEN 'LULUS' ELSE 'GAGAL' END,
       0
  FROM APPWRK.STG_LAPORAN s;
```

Jalankan view ini sebagai gerbang wajib. Tidak ada berkas yang boleh dikirim selama masih ada baris berstatus `GAGAL`.

63.5 Menangani perubahan format dari regulator

Perubahan format datang dari luar dengan tenggat yang sering ketat. Yang bisa Anda siapkan sebelumnya:

1. **Pisahkan logika ekstraksi dari logika pemformatan.** Perubahan format seharusnya menyentuh satu program, bukan sepuluh
2. **Simpan definisi format sebagai data, bukan kode.** Panjang kolom dan urutannya di tabel parameter
3. **Simpan berkas contoh dari regulator** bersama dengan versi program yang menghasilkannya
4. **Uji dengan data produksi yang di-mask di UAT**, karena masalah format hampir selalu muncul dari data pinggiran yang tidak ada di data uji sintetis

```
-- Definisi format sebagai data
CREATE TABLE APPPRM.FORMAT_LAPORAN (
    KD_LAPORAN VARCHAR(30) NOT NULL,
    VERSI_FMT  SMALLINT    NOT NULL,
    URUTAN     SMALLINT    NOT NULL,
    NM_KOLOM   VARCHAR(40) NOT NULL,
    POSISI     SMALLINT    NOT NULL,
    PANJANG    SMALLINT    NOT NULL,
    TIPE       CHAR(1)     NOT NULL,
    WAJIB      CHAR(1)     NOT NULL DEFAULT 'N',
    BERLAKU_DARI DATE      NOT NULL,
    PRIMARY KEY (KD_LAPORAN, VERSI_FMT, URUTAN)
);
```

Dengan struktur ini, perubahan panjang kolom dari regulator menjadi perubahan data dengan persetujuan, bukan perubahan program dengan siklus pengembangan penuh.

⚙️ Bab 64. Interface Antar Sistem

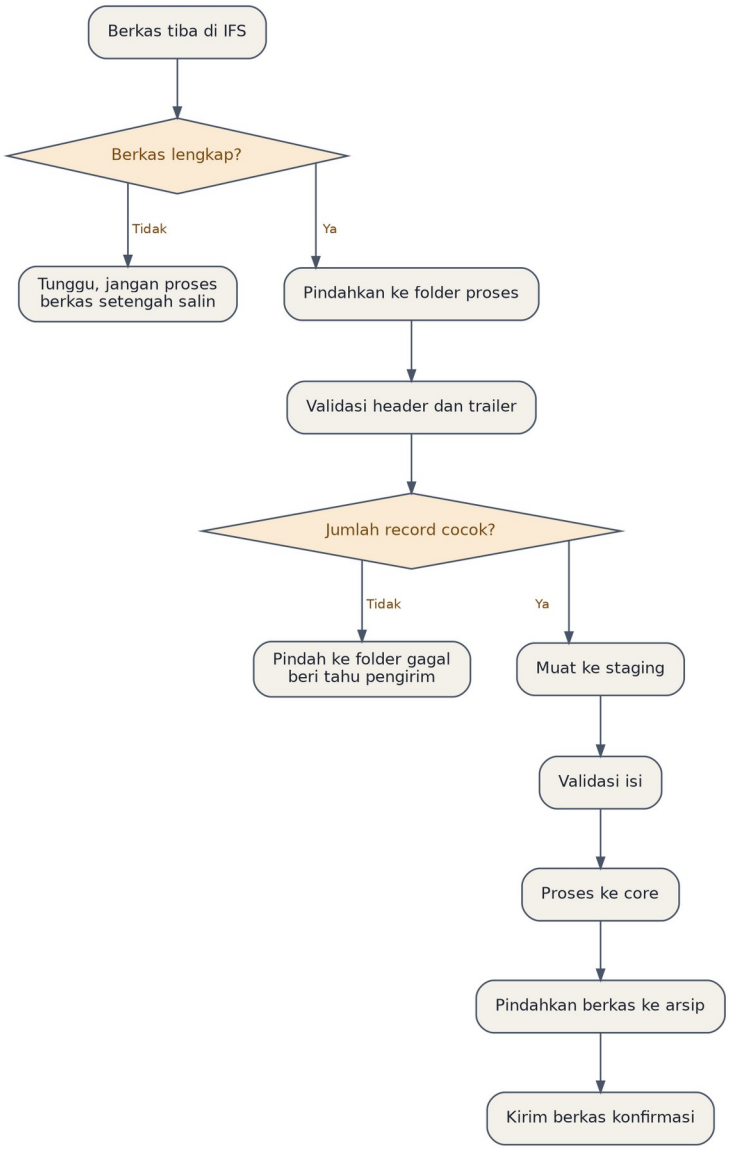
Core banking jarang berdiri sendiri. Di sekelilingnya ada puluhan interface, dan sebagian besar insiden operasional lahir di sana, bukan di dalam core.

64.1 Peta interface yang khas

Interface	Arah	Pola	Sifat kegagalan
Switching ATM	Dua arah	Real time, pesan	Timeout, dobel debet
Mobile dan internet banking	Dua arah	Real time, API	Lonjakan trafik
Sistem pembayaran nasional	Dua arah	Real time dan batch	Ketergantungan pihak luar
Data warehouse	Keluar	Batch harian	Menahan sumber daya
Sistem kartu	Dua arah	Batch dan real time	Rekonsiliasi rumit
Payroll korporat	Masuk	Batch berkas	Format berubah tanpa pemberitahuan
Biller dan e-commerce	Dua arah	Real time	Volume tidak terduga
Sistem AML	Keluar	Batch harian	Volume sangat besar

64.2 Pola interface berbasis berkas

Masih sangat lazim, dan punya kegagalan khasnya sendiri.



Gambar 72

Masalah berkas setengah salin adalah jebakan klasik. Program pemroses membaca berkas yang masih sedang ditulis pengirim, lalu memproses data tidak lengkap. Pencegahan yang andal:

```
PGM
   DCL VAR(&UKURAN1) TYPE(*DEC) LEN(15 0)
   DCL VAR(&UKURAN2) TYPE(*DEC) LEN(15 0)

   /* Baca ukuran, tunggu, baca lagi. Sama berarti selesai disalin. */
   CALL PGM(APPPGM/GETSIZE) PARM('/interface/in/data.txt' &UKURAN1)
   DLYJOB DLY(30)
   CALL PGM(APPPGM/GETSIZE) PARM('/interface/in/data.txt' &UKURAN2)

   IF COND(&UKURAN1 *NE &UKURAN2) THEN(DO)
      SNDMSG MSG('Berkas masih ditulis, proses ditunda') TOUSR(SYSENG)
      RETURN
   ENDDO

   /* Lebih baik lagi: pengirim membuat berkas penanda setelah selesai */
   CHKOBJ OBJ('/interface/in/data.txt.done') OBJTYPE(*STMF)
   MONMSG MSGID(CPFA0A9) EXEC(RETURN)
ENDPGM
```

Berkas penanda adalah solusi yang lebih baik daripada pemeriksaan ukuran. Sepakati ini dengan setiap mitra interface sejak awal.

64.3 Header dan trailer sebagai kontrak

```
-- Validasi jumlah dan total dari trailer berkas
WITH isi AS (
  SELECT COUNT(*) AS jml, SUM(NOMINAL) AS total
    FROM APPWRK.STG_INTERFACE
   WHERE ID_BATCH = :wIdBatch
     AND TIPE_BARIS = 'D'
),
trailer AS (
  SELECT JML_RECORD, TOTAL_NOMINAL
    FROM APPWRK.STG_INTERFACE
   WHERE ID_BATCH = :wIdBatch
     AND TIPE_BARIS = 'T'
)
SELECT i.jml, t.JML_RECORD,
       i.total, t.TOTAL_NOMINAL,
       CASE WHEN i.jml = t.JML_RECORD
             AND i.total = t.TOTAL_NOMINAL
            THEN 'LULUS' ELSE 'GAGAL' END AS hasil
  FROM isi i, trailer t;
```

Berkas yang gagal validasi trailer **tidak boleh diproses sebagian**. Tolak seluruhnya dan kembalikan ke pengirim.

64.4 Memantau kesehatan interface

```
CREATE TABLE APPWRK.INTERFACE_LOG (
    WAKTU       TIMESTAMP NOT NULL,
    KD_INTERFACE VARCHAR(20) NOT NULL,
    ARAH        CHAR(1)   NOT NULL,
    ID_BATCH    VARCHAR(40),
    JML_RECORD  BIGINT,
    TOTAL_NILAI DECIMAL(21,2),
    STATUS      CHAR(1)   NOT NULL,
    DURASI_DETIK INTEGER,
    PESAN       VARCHAR(500)
);
```

```
-- Interface yang seharusnya sudah jalan tetapi belum
SELECT j.KD_INTERFACE, j.NM_INTERFACE, j.JAM_HARAPAN,
       CURRENT TIME AS sekarang
  FROM APPPRM.INTERFACE_JADWAL j
 WHERE j.AKTIF = 'Y'
   AND j.JAM_HARAPAN < CURRENT TIME
   AND NOT EXISTS (
       SELECT 1 FROM APPWRK.INTERFACE_LOG l
        WHERE l.KD_INTERFACE = j.KD_INTERFACE
          AND DATE(l.WAKTU) = CURRENT DATE
          AND l.STATUS = 'S');

-- Tren durasi: interface yang melambat
SELECT KD_INTERFACE,
       DATE(WAKTU) AS tgl,
       AVG(DURASI_DETIK) AS rata_detik,
       MAX(JML_RECORD) AS record_terbanyak
  FROM APPWRK.INTERFACE_LOG
 WHERE WAKTU > CURRENT DATE - 30 DAYS
   AND STATUS = 'S'
 GROUP BY KD_INTERFACE, DATE(WAKTU)
 ORDER BY KD_INTERFACE, tgl;
```

Kueri pertama adalah salah satu pemantauan paling berharga yang bisa Anda pasang: ia mendeteksi interface yang **tidak berjalan sama sekali**. Kegagalan diam jauh lebih berbahaya daripada kegagalan yang berteriak, karena tidak ada yang tahu sampai seseorang menanyakan datanya.

64.5 Kontrak yang harus disepakati untuk tiap interface

1. Jam kirim yang disepakati dan toleransi keterlambatan
2. Format berkas atau pesan, dengan nomor versi
3. Mekanisme penandaan selesai
4. Isi header dan trailer
5. Apa yang terjadi bila validasi gagal — siapa memperbaiki, dalam berapa lama
6. Batas ukuran dan jumlah record
7. Prosedur kirim ulang dan aturan idempotensi
8. Kontak teknis kedua pihak, dengan cadangan
9. Jadwal pemeliharaan masing-masing pihak
10. Prosedur saat salah satu pihak sedang DR

Poin sepuluh hampir selalu terlupakan. Ketika Anda gagal ke DR, alamat IP Anda berubah, dan setiap mitra interface yang memakai daftar putih alamat IP akan menolak koneksi Anda. Ini ditemukan saat uji DR, bukan saat bencana nyata — kalau Anda benar-benar menguji.

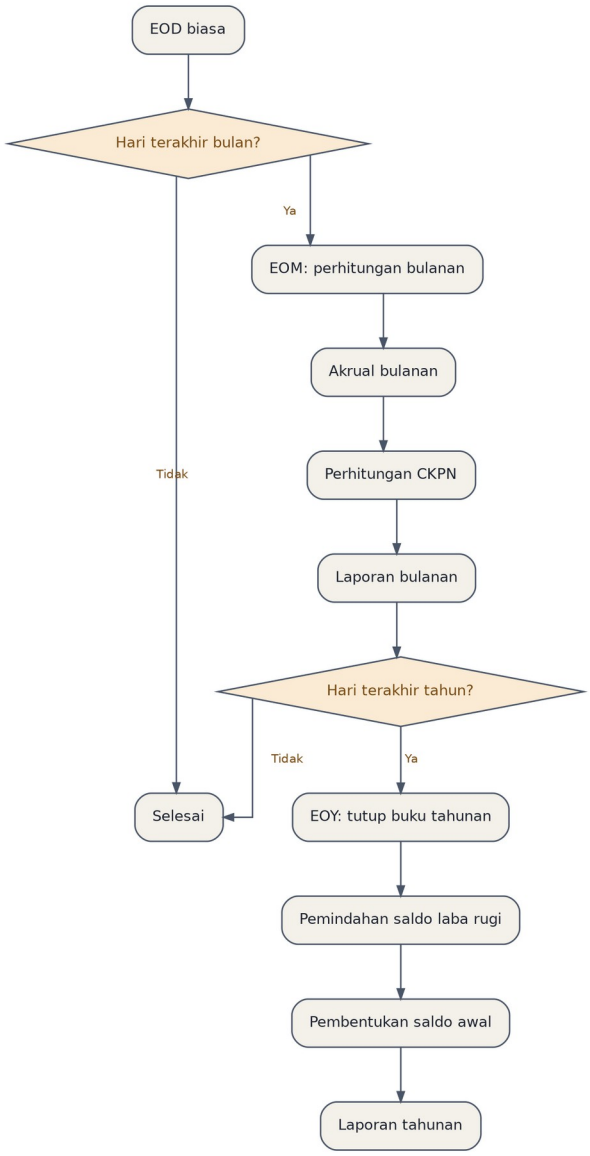
🌙 Bab 65. Batch di Luar EOD

Tidak semua batch berjalan malam hari. Sebagian besar institusi punya lapisan batch lain yang kurang diperhatikan tetapi tetap bisa merusak layanan.

65.1 Jenis batch dan risikonya

Jenis	Waktu	Risiko utama
End of Day	Malam	Menahan seluruh layanan besok
End of Month	Akhir bulan	Volume berlipat, jendela sama
End of Year	Akhir tahun	Jarang diuji, paling berisiko
Ekstraksi data warehouse	Setelah EOD	Memperpanjang jendela
Pemrosesan payroll	Siang hari	Beban mendadak di jam layanan
Kliring	Beberapa kali sehari	Tenggat keras dari luar
Pembersihan data	Akhir pekan	Kunci berkepanjangan
Pembentukan laporan	Setelah EOD	Kueri berat

65.2 End of Month dan End of Year



Gambar 73

Kenapa EOY paling berbahaya. Ia berjalan sekali setahun. Artinya program itu tidak diuji dalam produksi

selama dua belas bulan, sementara di sekelilingnya ada perubahan PTF, perubahan aplikasi, dan pertumbuhan data. Perlakukan EOY sebagai proyek tersendiri dengan uji lengkap di UAT memakai salinan data produksi terkini, dua bulan sebelum pelaksanaan.

65.3 Perencanaan jendela End of Month

```
-- Berapa lama EOD biasa, dibanding EOM, 12 bulan terakhir
SELECT YEAR(TGL_BUKU) AS thn, MONTH(TGL_BUKU) AS bln,
       CASE WHEN DAY(TGL_BUKU) =
                 DAY(LAST_DAY(TGL_BUKU)) THEN 'EOM' ELSE 'EOD' END AS jenis,
       AVG(TIMESTAMPDIFF(4, CHAR(MAX(JAM_SELESAI) - MIN(JAM_MULAI))))
         AS rata_menit
  FROM APPWRK.EOD_KONTROL
 WHERE TGL_BUKU > CURRENT DATE - 12 MONTHS
   AND STATUS = 'S'
 GROUP BY YEAR(TGL_BUKU), MONTH(TGL_BUKU),
          CASE WHEN DAY(TGL_BUKU) = DAY(LAST_DAY(TGL_BUKU))
               THEN 'EOM' ELSE 'EOD' END
 ORDER BY thn, bln, jenis;
```

Bila EOM memakan dua sampai tiga kali durasi EOD biasa, dan jendela Anda hanya cukup untuk satu setengah kali, Anda sedang menunggu insiden yang jadwalnya sudah diketahui.

65.4 Batch di jam layanan

Batch yang berjalan saat nasabah bertransaksi menuntut kehati-hatian berbeda.

Aturan	Alasan
Prioritas run lebih rendah dari interaktif	Batch mengalah pada layanan
Pool memori terpisah	Tidak mengusir halaman sesi interaktif
Commit sering, batch kecil	Kunci tidak berkepanjangan
Jeda terjadwal di antara blok	Memberi ruang bagi transaksi
Batas waktu keras dengan penghentian mandiri	Tidak menabrak jam sibuk

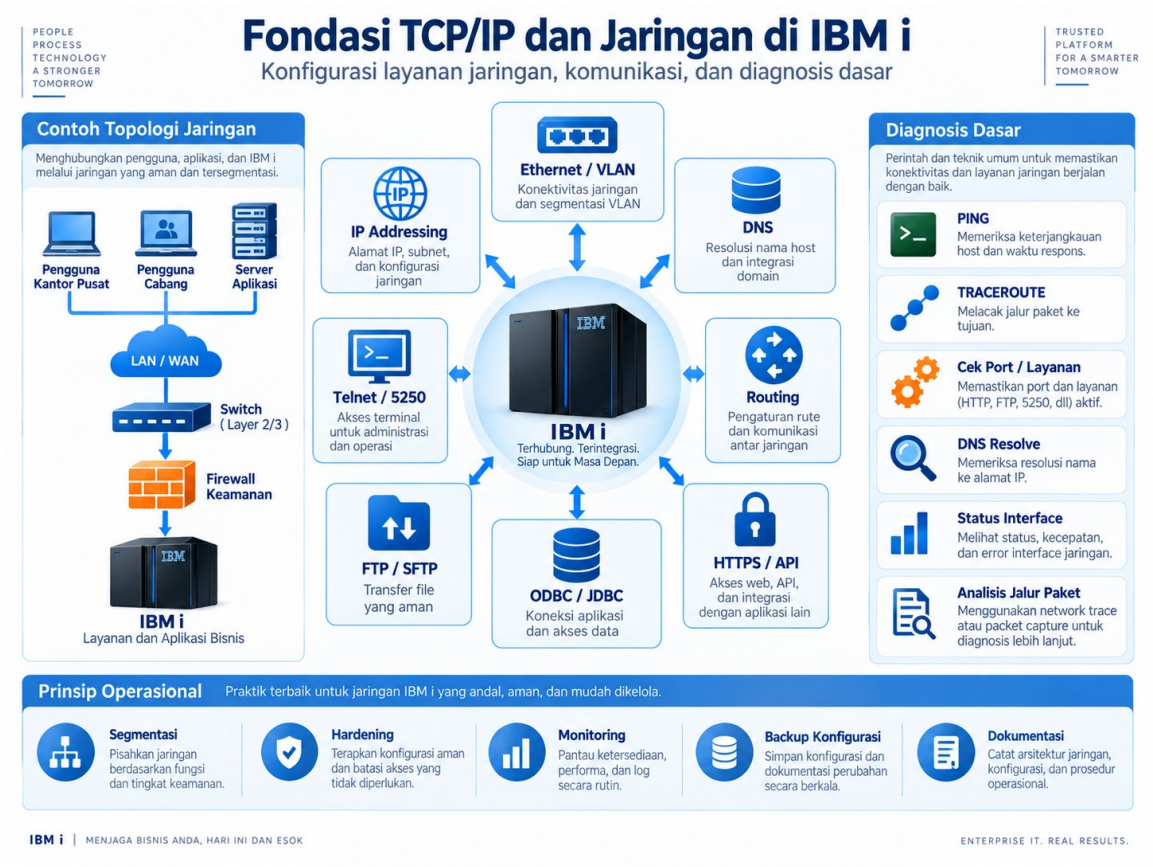
```
/* Batch siang hari yang tahu diri */
PGM
   DCL VAR(&JAM) TYPE(*CHAR) LEN(6)

   CHGJOB RUNPTY(50) TIMESLICE(2000)

 LOOP:
   RTVSYSVAL SYSVAL(QTIME) RTNVAR(&JAM)

   /* Berhenti sendiri saat mendekati jam sibuk */
   IF COND(%SST(&JAM 1 2) *GE '09' *AND %SST(&JAM 1 2) *LT '11') THEN(DO)
      SNDMSG MSG('Batch ditunda, memasuki jam sibuk') TOUSR(SYSENG)
      RETURN
   ENDDO

   CALL PGM(APPPGM/PROSESBLOK)
   DLYJOB DLY(5)
   GOTO CMDLBL(LOOP)
ENDPGM
```

Program batch yang tahu kapan harus berhenti sendiri jauh lebih baik daripada program yang harus dihentikan operator saat telepon keluhan mulai masuk.

🌐 Bagian XVII — Jaringan dan TCP/IP

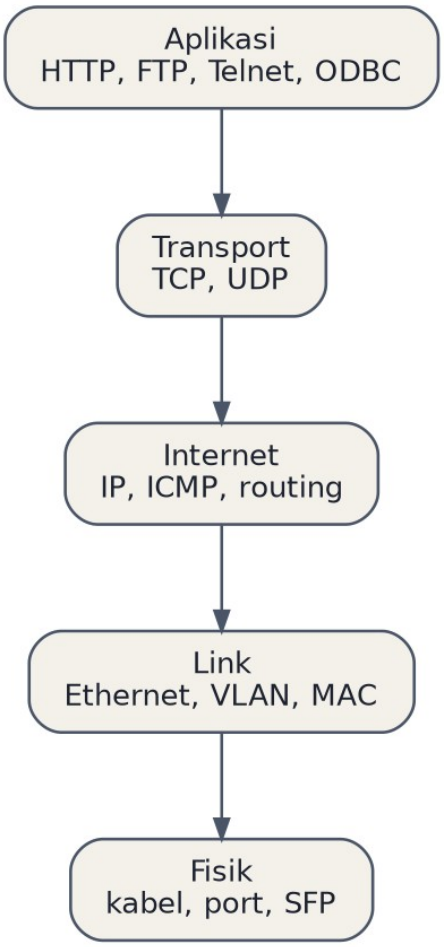
Sebagian besar System Engineer IBM i berasal dari latar belakang aplikasi, dan jaringan menjadi titik buta yang paling sering. Padahal ketika layanan terganggu, pertanyaan pertama hampir selalu: apakah ini masalah sistem atau masalah jaringan? Anda harus bisa menjawabnya sendiri.

🌐 Bab 66. Fondasi TCP/IP



Gambar 74 — Fondasi TCP/IP dan jaringan di IBM i

66.1 Lapisan dan apa yang bisa gagal di tiap lapisan



Gambar 75

Lapisan	Gejala kegagalan	Alat diagnosis
Fisik	Interface tidak naik sama sekali	`WRKCFGSTS`, lampu port switch
Link	Interface naik tetapi tidak ada lalu lintas	`NETSTAT`, ARP, VLAN
Internet	Bisa ping tetangga, tidak bisa ping jauh	`PING`, `TRACEROUTE`, tabel rute
Transport	Ping berhasil, port ditolak	`NETSTAT`, telnet ke port
Aplikasi	Port terbuka tetapi layanan menolak	Log aplikasi, exit program

Diagnosis yang benar selalu dari bawah ke atas. Memeriksa konfigurasi aplikasi sementara kabelnya lepas adalah pemborosan waktu yang sangat umum.

66.2 Alamat IP dan subnet

Subnet mask menentukan bagian mana dari alamat yang menunjuk jaringan dan bagian mana yang menunjuk host.

Prefix	Subnet mask	Jumlah host	Kegunaan khas
/30	255.255.255.252	2	Tautan titik ke titik antar

Prefix	Subnet mask	Jumlah host	Kegunaan khas
			router
/29	255.255.255.248	6	Segmen kecil perangkat
/28	255.255.255.240	14	Kelompok server kecil
/27	255.255.255.224	30	Segmen server
/26	255.255.255.192	62	Segmen server menengah
/25	255.255.255.128	126	Setengah blok C
/24	255.255.255.0	254	Satu blok C, paling lazim
/23	255.255.254.0	510	Dua blok C
/22	255.255.252.0	1022	Segmen cabang besar
/16	255.255.0.0	65534	Blok B, jarang dipakai utuh

Cara cepat menghitung. Jumlah host adalah 2 pangkat (32 dikurangi prefix), dikurangi 2 untuk alamat jaringan dan alamat broadcast.

Contoh: alamat 10.20.30.45 dengan mask 255.255.255.192 (/26).

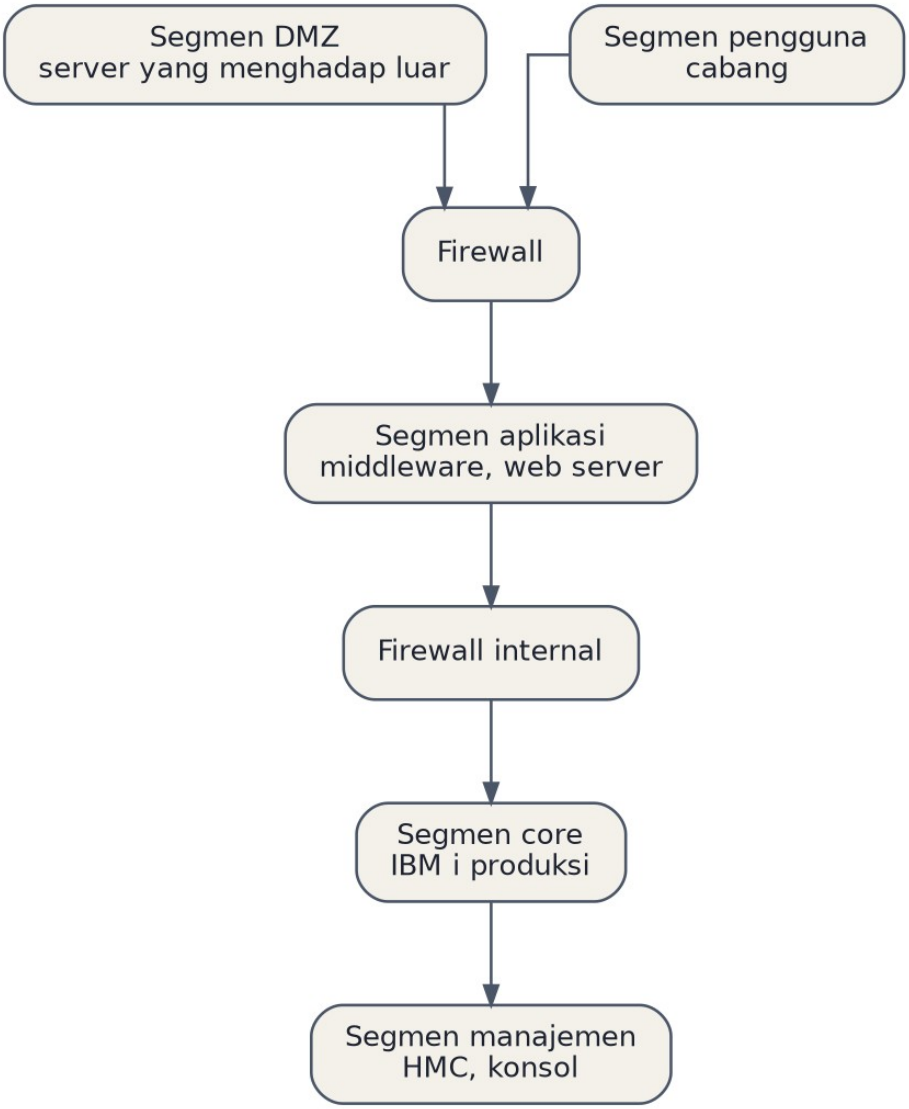
Unsur	Nilai	Cara
Ukuran blok	64	256 dikurangi 192
Alamat jaringan	10.20.30.0	Kelipatan 64 terdekat di bawah 45
Host pertama	10.20.30.1	Jaringan tambah 1
Host terakhir	10.20.30.62	Broadcast kurang 1
Broadcast	10.20.30.63	Jaringan tambah 63
Blok berikutnya	10.20.30.64	Jaringan tambah 64

Mengapa ini penting bagi Anda: kesalahan subnet mask menyebabkan gejala yang membingungkan — mesin bisa menghubungi sebagian alamat dan tidak bisa menghubungi sebagian lain, tanpa pola yang jelas bagi orang yang tidak menghitung.

66.3 Alamat privat dan perencanaan

Blok	Rentang	Jumlah alamat
10.0.0.0/8	10.0.0.0 sampai 10.255.255.255	16,7 juta
172.16.0.0/12	172.16.0.0 sampai 172.31.255.255	1 juta
192.168.0.0/16	192.168.0.0 sampai 192.168.255.255	65 ribu

Dalam lingkungan perbankan, pemisahan segmen adalah kontrol keamanan, bukan sekadar kerapian:



Gambar 76

IBM i produksi tidak pernah diletakkan di segmen yang sama dengan pengguna akhir, dan tidak pernah bisa dihubungi langsung dari internet.

66.4 Gateway dan routing

Gateway adalah alamat yang dipakai untuk mencapai jaringan di luar subnet lokal. Kesalahan paling umum: gateway tidak berada dalam subnet yang sama dengan interface.

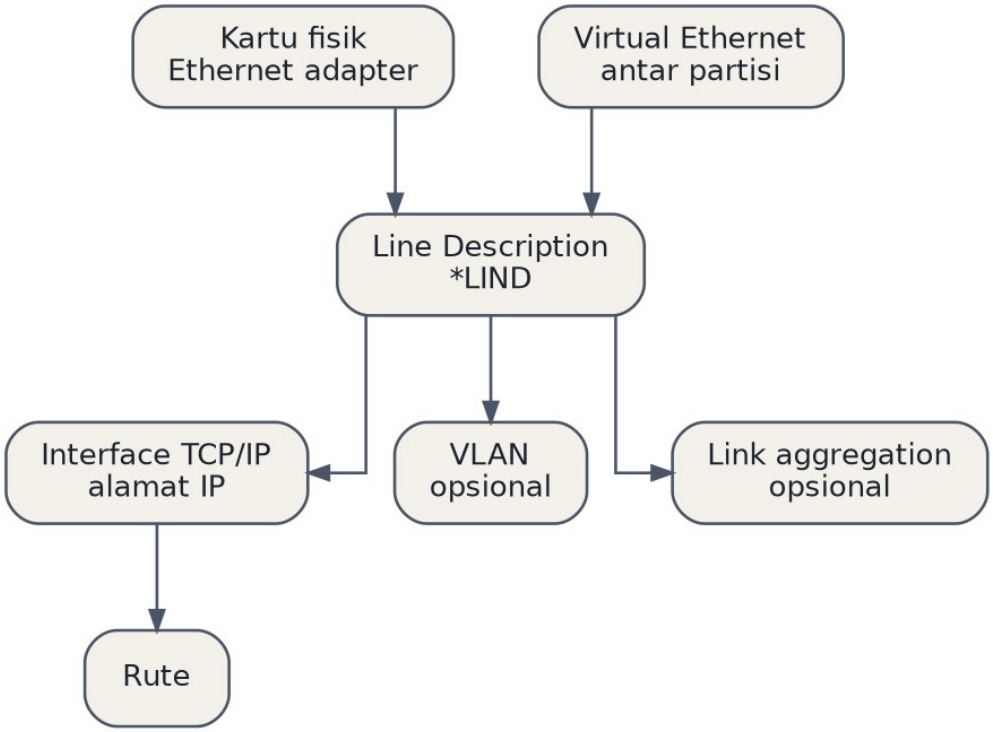
```
/* Melihat tabel rute */
NETSTAT OPTION(*RTE)

/* Menambah rute default */
ADDTCPRTE RTEDEST(*DFTROUTE) SUBNETMASK(*NONE) +
          NEXTHOP('10.20.30.1') +
          BINDIFC('10.20.30.50')

/* Rute statis ke jaringan tertentu lewat gateway lain */
ADDTCPRTE RTEDEST('192.168.50.0') SUBNETMASK('255.255.255.0') +
          NEXTHOP('10.20.30.254') +
          MTU(*IFC)

/* Mengubah dan menghapus */
CHGTCPRTE RTEDEST('192.168.50.0') SUBNETMASK('255.255.255.0') +
          NEXTHOP('10.20.30.253')
RMVTCPRTE RTEDEST('192.168.50.0') SUBNETMASK('255.255.255.0') +
          NEXTHOP('10.20.30.253')
```

```
-- Tabel rute lewat SQL, lebih mudah diarsipkan sebagai baseline
SELECT ROUTE_DESTINATION, SUBNET_MASK, NEXT_HOP,
       ROUTE_TYPE, ROUTE_PRECEDENCE, MAXIMUM_TRANSMISSION_UNIT,
       ROUTE_STATUS
  FROM QSYS2.NETSTAT_ROUTE_INFO
 ORDER BY ROUTE_DESTINATION;
```

Urutan pemilihan rute. Sistem memilih rute paling spesifik lebih dulu. Rute /32 mengalahkan /24, dan /24 mengalahkan rute default. Bila lalu lintas ke satu tujuan berperilaku aneh sementara tujuan lain normal, periksa apakah ada rute statis yang lebih spesifik yang terlupakan.

🌐 Bab 67. Konfigurasi Jaringan di IBM i

67.1 Susunan objek jaringan



Gambar 77

Berbeda dari Linux atau Windows di mana kartu jaringan dan alamat IP menyatu, IBM i memisahkan keduanya. Satu line description bisa membawa banyak alamat IP.

67.2 Membuat line description Ethernet

```
/* Melihat sumber daya perangkat keras yang tersedia */
WRKHDWRSC TYPE(*CMN)

/* Membuat line description */
CRTLINETH LIND(ETHLINE01) +
          RSRCNAME(CMN03) +
          LINESPEED(*AUTO) +
          DUPLEX(*AUTO) +
          MAXFRAME(1496) +
          TEXT('Jaringan produksi utama')

/* Line description dengan VLAN tagging */
CRTLINETH LIND(ETHVLAN20) +
          RSRCNAME(CMN03) +
          VLANID(20) +
          TEXT('VLAN 20 segmen core')

/* Mengaktifkan */
VRYCFG CFGOBJ(ETHLINE01) CFGTYPE(*LIN) STATUS(*ON)
```

```
/* Memeriksa status */
WRKCFGSTS CFGTYPE(*LIN) CFGD(ETHLINE01)
DSPLIND LIND(ETHLINE01)
```

Catatan tentang kecepatan dan duplex. `*AUTO` hampir selalu benar untuk perangkat modern. Menyetel manual di satu sisi dan otomatis di sisi lain menghasilkan duplex mismatch: interface tampak naik, ping berhasil, tetapi throughput anjlok dan terjadi banyak error. Ini gejala yang sangat membingungkan dan sangat sering terjadi.

```
-- Statistik kesalahan pada line description
SELECT LINE_DESCRIPTION, LINE_TYPE, LINE_STATUS,
       TOTAL_PACKETS_RECEIVED, TOTAL_PACKETS_TRANSMITTED,
       RECEIVE_ERRORS, TRANSMIT_ERRORS
  FROM QSYS2.NETSTAT_INTERFACE_INFO;
```

Jumlah error yang naik terus adalah tanda masalah fisik atau duplex mismatch, bukan masalah konfigurasi IP.

67.3 Interface TCP/IP

```
/* Menambah interface */
ADDTCPIFC INTNETADR('10.20.30.50') +
          LIND(ETHLINE01) +
          SUBNETMASK('255.255.255.0') +
          MTU(*LIND) +
          AUTOSTART(*YES) +
          TEXT('Alamat produksi utama')

/* Alamat kedua pada line yang sama, untuk layanan terpisah */
ADDTCPIFC INTNETADR('10.20.30.51') +
          LIND(ETHLINE01) +
          SUBNETMASK('255.255.255.0') +
          AUTOSTART(*YES) +
          TEXT('Alamat khusus web service')

/* Mengaktifkan dan menonaktifkan */
STRTCPIFC INTNETADR('10.20.30.50')
ENDTCPIFC INTNETADR('10.20.30.50')

/* Mengubah */
CHGTCPIFC INTNETADR('10.20.30.50') SUBNETMASK('255.255.255.128')

/* Menghapus */
RMVTCPIFC INTNETADR('10.20.30.51')

/* Melihat semua */
NETSTAT OPTION(*IFC)
CFGTCP  /* menu konfigurasi lengkap, opsi 1 sampai 12 */
```

Padanan perintah antar platform. Bagi Anda yang terbiasa dengan platform lain:

Tugas	Windows	Linux	IBM i
Lihat alamat IP	`ipconfig /all`	`ip addr show`	`NETSTAT OPTION(*IFC)`
Lihat tabel rute	`route print`	`ip route`	`NETSTAT OPTION(*RTE)`
Lihat koneksi aktif	`netstat -ano`	`ss -tunap`	`NETSTAT OPTION(*CNN)`
Uji jangkauan	`ping`	`ping`	`PING`

Tugas	Windows	Linux	IBM i
Telusuri jalur	`tracert`	`traceroute`	`TRACEROUTE`
Lihat cache ARP	`arp -a`	`ip neigh`	`NETSTAT OPTION(*IFC)` lalu opsi detail
Lepas dan perbarui DHCP	`ipconfig /renew`	`dhclient`	Jarang dipakai; IBM i umumnya statis
Resolusi nama	`nslookup`	`dig`	`NSLOOKUP` di PASE
Konfigurasi kartu	Panel kontrol	`nmcli`	`CRTLINETH`, `ADDTCPIFC`

IBM i produksi hampir selalu memakai alamat statis. DHCP untuk server core banking adalah pilihan yang sulit dipertahankan di depan auditor.

67.4 Virtual IP dan takeover alamat

Virtual IP tidak terikat pada satu kartu fisik, sehingga bisa berpindah saat kartu gagal atau saat failover.

```
/* Interface virtual, tanpa line description fisik */
ADDTCPIFC INTNETADR('10.20.30.100') +
          LIND(*VIRTUALIP) +
          SUBNETMASK('255.255.255.255') +
          TEXT('Alamat layanan yang berpindah')

/* Mengaitkan dengan interface fisik untuk redundansi */
CHGTCPIFC INTNETADR('10.20.30.50') +
          PREFIFC('10.20.30.100')
```

Dalam konfigurasi PowerHA, alamat inilah yang berpindah ke node DR saat switchover, sehingga aplikasi klien tidak perlu diubah konfigurasinya.

67.5 Agregasi tautan

```
/* Menggabungkan beberapa port untuk redundansi dan throughput */
CRTLINETH LIND(ETHAGG01) +
          RSRCNAME(*AGGREGATE) +
          AGGPCY(*LNKAGG) +
          AGGRSRCL(CMN03 CMN04) +
          TEXT('Agregasi dua port')
```

Sisi switch harus dikonfigurasi sepadan. Agregasi yang hanya dikonfigurasi di satu sisi menyebabkan lalu lintas hilang secara acak — salah satu masalah jaringan yang paling sulit didiagnosis karena gejalanya tidak konsisten.

67.6 Resolusi nama

```
/* Konfigurasi domain dan server DNS */
CHGTCPDMN HOSTNAME(PRDIBMI) +
          DMNNAME(bank.co.id) +
          DMNSCHLIST(bank.co.id dc.bank.co.id) +
          INTNETADR('10.20.10.5' '10.20.10.6') +
          HOSTSCHPTY(*LOCAL)

/* Tabel host lokal */
ADDTCPHTE INTNETADR('10.20.40.10') +
          HOSTNAME(('midware01') ('midware01.bank.co.id'))

CHGTCPHTE INTNETADR('10.20.40.10') +
          HOSTNAME(('midware01') ('mw01'))

RMVTCPHTE INTNETADR('10.20.40.10')

/* Melihat seluruh tabel host */
CFGTCP OPTION(10)
```

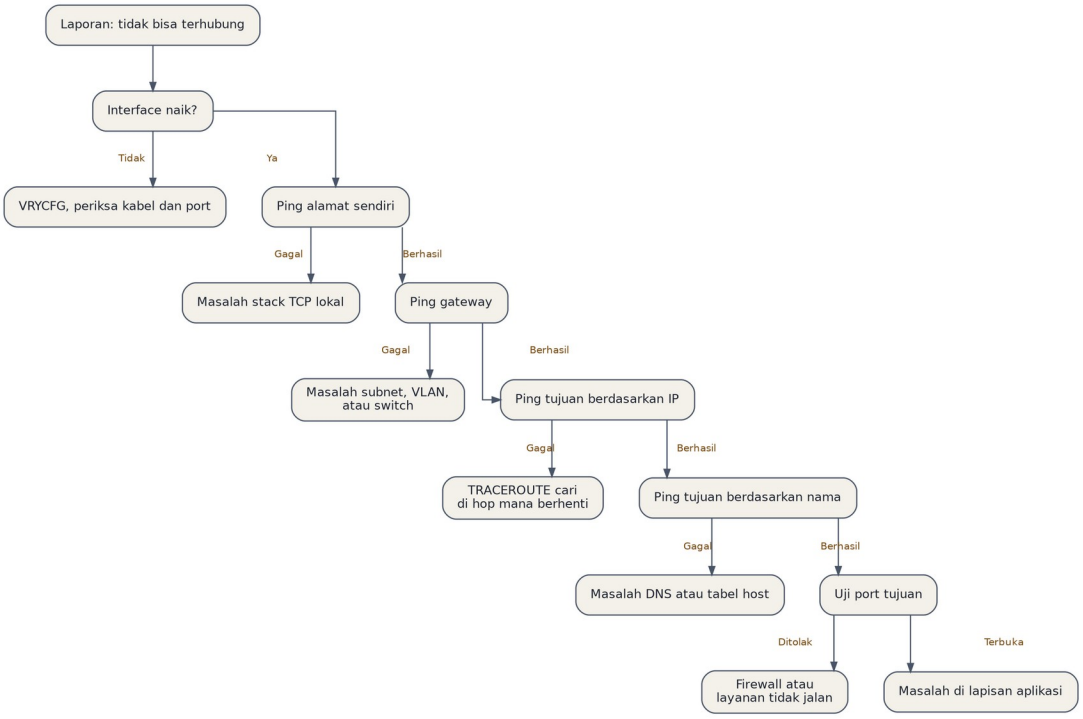
`HOSTSCHPTY(*LOCAL)` berarti tabel host lokal diperiksa lebih dulu sebelum DNS. Untuk server produksi ini pilihan yang lebih aman: bila DNS bermasalah, koneksi ke mitra kritikal tetap berjalan.

```
-- Isi tabel host lewat SQL
SELECT INTERNET_ADDRESS, HOST_NAME_1, HOST_NAME_2, HOST_NAME_3
  FROM QSYS2.TCPIP_HOST_TABLE
 ORDER BY INTERNET_ADDRESS;

-- Konfigurasi DNS yang aktif
SELECT * FROM QSYS2.TCPIP_INFO;
```

Praktik yang disarankan untuk mitra kritikal. Daftarkan alamat mitra interface yang kritikal — switching ATM, sistem pembayaran — di tabel host lokal, bukan hanya mengandalkan DNS. Kegagalan DNS tidak boleh menjatuhkan jalur pembayaran.

🌐 Bab 68. Diagnosis Jaringan

68.1 Alur diagnosis baku



Gambar 78

Ikuti urutan ini tanpa melompat. Melompat langsung ke dugaan favorit adalah sebab paling umum insiden jaringan berlarut-larut.

68.2 PING

```
/* Dasar */
PING RMTSYS('10.20.40.10')

/* Dengan parameter lengkap */
PING RMTSYS('midware01.bank.co.id') +
     PKTLEN(1472) +
     NBRPKT(10) +
     WAITTIME(5) +
     INTNETADR('10.20.30.50') +
     MSGMODE(*VERBOSE)

/* Menguji MTU: paket besar tanpa fragmentasi */
PING RMTSYS('10.20.40.10') PKTLEN(1472)
```

Parameter	Kegunaan
`PKTLEN`	Ukuran paket; untuk menguji masalah MTU
`NBRPKT`	Jumlah paket; banyak untuk menguji kehilangan
`WAITTIME`	Batas tunggu tiap paket
`INTNETADR`	Alamat sumber; penting bila punya banyak interface
`MSGMODE(*VERBOSE)`	Tampilkan tiap paket, bukan hanya ringkasan

Menguji MTU. Bila ping ukuran kecil berhasil tetapi ukuran 1472 gagal, ada perangkat di jalur dengan MTU lebih kecil. Gejalanya di aplikasi: koneksi terbentuk, pertukaran data kecil berhasil, tetapi transfer besar menggantung. Ini masalah yang sering salah didiagnosis sebagai masalah aplikasi.

Ping berhasil tetapi layanan tidak jalan. Ping hanya membuktikan lapisan internet. Ia tidak membuktikan apa pun tentang port atau aplikasi.

68.3 TRACEROUTE

```
TRACEROUTE RMTSYS('10.50.60.70')

TRACEROUTE RMTSYS('mitra.example.co.id') +
           MAXHOP(20) +
           WAITTIME(3) +
           LCLINTNETA('10.20.30.50')
```

Bacaan hasilnya:

Pola	Arti
Berhenti di hop pertama	Masalah gateway lokal
Berhenti di tengah, lalu tanda bintang	Perangkat memblokir ICMP, belum tentu masalah
Waktu melonjak di satu hop	Kemacetan atau jalur lambat di titik itu
Jalur berubah tiap kali dijalankan	Ada beberapa jalur, load balancing
Berputar antara dua hop	Kesalahan routing, routing loop

Tanda bintang di hop tengah bukan selalu masalah. Banyak firewall tidak menjawab ICMP tetapi tetap meneruskan lalu lintas. Yang penting adalah apakah hop terakhir tercapai.

68.4 NETSTAT

```
/* Menu utama */
NETSTAT

/* Langsung ke bagian tertentu */
NETSTAT OPTION(*IFC)   /* interface dan alamat */
NETSTAT OPTION(*RTE)   /* tabel rute */
NETSTAT OPTION(*CNN)   /* koneksi aktif */
```

Di layar koneksi, kolom status adalah yang paling informatif:

Status	Arti	Yang perlu diperhatikan
`Listen`	Port menunggu koneksi	Bandingkan dengan daftar yang disetujui
`Established`	Koneksi aktif	Jumlah tidak wajar berarti kebocoran koneksi
`Time-wait`	Baru ditutup, menunggu	Banyak sekali berarti churn koneksi tinggi
`Close-wait`	Menunggu aplikasi menutup	Menumpuk berarti bug aplikasi
`Syn-sent`	Mencoba menyambung	Banyak berarti tujuan tidak menjawab
`Syn-received`	Setengah terbuka	Lonjakan bisa berarti serangan

```
-- Ringkasan status koneksi, untuk pemantauan terjadwal
SELECT TCP_STATE, COUNT(*) AS jumlah
  FROM QSYS2.NETSTAT_INFO
 GROUP BY TCP_STATE
 ORDER BY jumlah DESC;

-- Koneksi terbanyak dari satu alamat
SELECT REMOTE_ADDRESS, COUNT(*) AS jml_koneksi,
       MIN(CONNECTION_OPENED_TIMESTAMP) AS koneksi_tertua
  FROM QSYS2.NETSTAT_JOB_INFO
 GROUP BY REMOTE_ADDRESS
HAVING COUNT(*) > 20
 ORDER BY jml_koneksi DESC;

-- Koneksi menganggur sangat lama
SELECT REMOTE_ADDRESS, LOCAL_PORT, IDLE_TIME,
       AUTHORIZATION_NAME, JOB_NAME
  FROM QSYS2.NETSTAT_JOB_INFO
 WHERE IDLE_TIME > 7200
 ORDER BY IDLE_TIME DESC;
```

Penumpukan `Close-wait` adalah temuan klasik: aplikasi klien membuka koneksi ODBC dan tidak pernah menutupnya. Bila dibiarkan, ia akan menghabiskan batas koneksi dan menolak pengguna sah.

68.5 Menguji port tujuan

Untuk memeriksa apakah sebuah port di sistem lain terbuka, cara paling sederhana adalah mencoba menyambung.

```
/* Telnet ke port sembarang sebagai uji koneksi */
TELNET RMTSYS('10.20.40.10') PORT(8080)
```

Bila layar kosong muncul, port terbuka. Bila muncul pesan penolakan segera, port tertutup dan ada yang menjawab. Bila menggantung sampai timeout, biasanya firewall membuang paket tanpa menjawab.

Dari PASE, alat yang lebih nyaman tersedia:

```
# Dari QP2TERM atau SSH ke IBM i
curl -v telnet://10.20.40.10:8080
curl -I https://mitra.example.co.id

# Uji resolusi nama
nslookup midware01.bank.co.id
dig midware01.bank.co.id
dig -x 10.20.40.10          # pencarian terbalik

# Uji rantai sertifikat TLS
openssl s_client -connect mitra.example.co.id:443 -showcerts

# Periksa tanggal kedaluwarsa sertifikat pihak lawan
echo | openssl s_client -connect mitra.example.co.id:443 2>/dev/null \
  | openssl x509 -noout -dates
```

Perintah `openssl s_client` sangat berguna saat integrasi TLS gagal. Ia menunjukkan versi protokol yang disepakati, cipher yang dipakai, dan rantai sertifikat — tiga hal yang menjadi penyebab hampir semua kegagalan handshake.

68.6 Pemindaian port untuk audit sendiri

Pemindaian port adalah bagian sah dari pengelolaan sistem: Anda perlu tahu port apa saja yang benar-benar terbuka dari sudut pandang jaringan, bukan hanya apa yang tercatat di konfigurasi.

Sebelum melakukan apa pun: dapatkan persetujuan tertulis. Pemindaian terhadap sistem yang bukan tanggung jawab Anda, termasuk sistem mitra, adalah masalah hukum dan kontraktual. Di lingkungan perbankan, pemindaian tanpa pemberitahuan juga akan memicu alarm di tim keamanan dan menghabiskan waktu banyak orang.

```
# Dijalankan dari host manajemen di segmen yang sama,
# terhadap sistem Anda sendiri, dalam jendela yang disepakati

# Pemindaian port umum
nmap -sT 10.20.30.50

# Rentang port tertentu, dengan deteksi versi layanan
nmap -sV -p 20-25,80,443,446,449,8470-8476,9470-9476 10.20.30.50

# Satu segmen, hanya memeriksa host mana yang hidup
nmap -sn 10.20.30.0/24
```

Port yang khas dipakai IBM i dan perlu Anda kenali:

Port	Layanan	Catatan keamanan
21	FTP	Matikan atau paksa TLS
22	SSH	Pengganti Telnet yang disarankan
23	Telnet	Hanya dengan TLS
25	SMTP	Matikan bila tidak mengirim surel
80	HTTP	Alihkan ke 443
443	HTTPS	Utama untuk layanan web
446, 448	DDM dan DRDA	Batasi ketat
449	Port mapper	Dibutuhkan klien lama
992	Telnet dengan TLS	Pengganti port 23
2001	HTTP Admin	Batasi ke jaringan manajemen
8470–8476	Host server tanpa TLS	Pertimbangkan mematikan
9470–9476	Host server dengan TLS	Yang seharusnya dipakai

Bandingkan hasil pemindaian dengan daftar port yang disetujui. Setiap port terbuka yang tidak ada di daftar adalah temuan yang harus dijelaskan atau ditutup.

```
-- Daftar internal untuk dibandingkan dengan hasil pemindaian
SELECT LOCAL_PORT, LOCAL_PORT_NAME, PROTOCOL, JOB_NAME
  FROM QSYS2.NETSTAT_INFO
 WHERE TCP_STATE = 'LISTEN'
 ORDER BY LOCAL_PORT;
```

68.7 Penelusuran paket

Ketika semua alat di atas tidak cukup, tangkap paketnya.

```
/* Mulai penangkapan pada satu line description */
STRCMNTRC CFGOBJ(ETHLINE01) CFGTYPE(*LIN) +
          MAXSTG(128) +
          TRCFULL(*STOPTRC) +
          USRDTA(*MAX)

/* Hentikan setelah masalah terjadi */
ENDCMNTRC CFGOBJ(ETHLINE01) CFGTYPE(*LIN)

/* Cetak hasil, disaring agar terbaca */
PRTCMNTRC CFGOBJ(ETHLINE01) CFGTYPE(*LIN) +
          CTLD(*NONE) +
          FMTTCP(*YES) +
          TCPIPADR('10.20.40.10') +
          FMTBCD(*NO)

/* Hapus setelah selesai, jejak memakan penyimpanan */
DLTCMNTRC CFGOBJ(ETHLINE01) CFGTYPE(*LIN)
```

Untuk analisis yang lebih dalam, jejak bisa diekspor dan dibuka dengan alat analisis paket standar di komputer Anda.

Kapan penelusuran paket dibenarkan. Ia memakan sumber daya dan menangkap isi lalu lintas — yang di lingkungan perbankan berarti data nasabah. Pakai hanya ketika diagnosis lain buntu, batasi penyaringannya ke alamat yang diselidiki, hapus jejaknya segera setelah analisis, dan catat kegiatannya sebagai bagian dari penanganan insiden.

68.8 Masalah jaringan yang paling sering di lingkungan IBM i

Gejala	Penyebab yang paling sering
Koneksi ODBC lambat hanya dari satu cabang	MTU tidak cocok di jalur itu
Transfer besar menggantung, kecil berhasil	MTU atau fragmentasi diblokir
Layanan hilang setelah failover	Alamat IP berubah, mitra memakai daftar putih IP
Koneksi terputus setelah beberapa menit diam	Timeout perangkat di tengah jalur
Throughput jauh di bawah kapasitas port	Duplex mismatch
Sebagian tujuan tercapai, sebagian tidak	Subnet mask atau rute statis salah
Semua lambat setiap pukul tertentu	Batch atau backup jaringan bertabrakan
Nama tidak terselesaikan tetapi IP berhasil	DNS atau urutan pencarian nama

Baris ketiga layak diberi perhatian khusus. Ini ditemukan saat uji DR bila Anda benar-benar menguji sampai lapisan interface, dan ditemukan saat bencana bila Anda tidak.

🌐 Bab 69. Layanan Jaringan

69.1 Mengelola server TCP

```
/* Memulai dan menghentikan */
STRTCPSVR SERVER(*TELNET)
ENDTCPSVR SERVER(*TELNET)

STRTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
ENDTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)

/* Server yang dimulai otomatis saat STRTCP */
CHGFTPA AUTOSTART(*NO)
CHGTELNA AUTOSTART(*YES)
```

```
-- Status seluruh server, untuk pemantauan terjadwal
SELECT SERVER_NAME, SERVER_STATUS, AUTOSTART, ACTIVE_JOBS
  FROM QSYS2.SERVER_SHARE_INFO
 ORDER BY SERVER_NAME;
```

Prinsip pertama pengerasan. Setiap server yang berjalan adalah permukaan serangan. Matikan yang tidak dipakai, dan pastikan `AUTOSTART` untuk server itu juga dimatikan — kalau tidak, ia hidup lagi pada IPL berikutnya.

69.2 Telnet

Telnet polos mengirim nama pengguna dan password sebagai teks terbuka. Di lingkungan perbankan ini tidak dapat diterima.

```
CHGTELNA ALWSSL(*ONLY)
CHGTELNA TIMMRKTIMO(300)
```

Nilai `ALWSSL`	Arti
`*NO`	Hanya Telnet polos, port 23
`*YES`	Keduanya, port 23 dan 992
`*ONLY`	Hanya terenkripsi, port 992

Setel `*ONLY`, lalu perbarui konfigurasi emulator 5250 di seluruh klien agar memakai TLS. Access Client Solutions mendukungnya secara bawaan.

```
CHGSYSVAL SYSVAL(QINACTITV) VALUE('15')
CHGSYSVAL SYSVAL(QINACTMSGQ) VALUE('*DSCJOB')
```

69.3 SSH

SSH tersedia di PASE dan merupakan pengganti yang lebih baik untuk banyak keperluan.

```
STRTCPSVR SERVER(*SSHD)
ENDTCPSVR SERVER(*SSHD)
```

Konfigurasi ada di `/QOpenSys/QIBM/UserData/SC1/OpenSSH/etc/sshd_config`. Pengerasan yang disarankan:

```
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
X11Forwarding no
MaxAuthTries 3
ClientAliveInterval 300
ClientAliveCountMax 2
AllowUsers builduser deployuser
```

```
# Menyiapkan autentikasi kunci untuk akun otomasi
mkdir -p /home/builduser/.ssh
chmod 700 /home/builduser/.ssh
touch /home/builduser/.ssh/authorized_keys
chmod 600 /home/builduser/.ssh/authorized_keys
```

Autentikasi berbasis kunci penting untuk pipeline CI/CD: proses otomatis tidak bisa mengetikkan password, dan menyimpan password di skrip adalah temuan audit yang pasti.

69.4 FTP

```
CHGFTPA AUTOSTART(*NO) +
        INACTTIMO(300) +
        CCSID(37) +
        ALWSSL(*ONLY)
```

Kalau FTP harus tetap ada, pasang exit program untuk mengendalikan siapa boleh melakukan apa.

```
ADDEXITPGM EXITPNT(QIBM_QTMF_SERVER_REQ) FORMAT(VLRQ0100) +
           PGMNBR(1) PGM(SECLIB/FTPEXIT) +
           TEXT('Kendali perintah FTP')

ADDEXITPGM EXITPNT(QIBM_QTMF_SVR_LOGON) FORMAT(TCPL0300) +
           PGMNBR(1) PGM(SECLIB/FTPLOGON) +
           TEXT('Kendali login FTP')
```

Exit program FTP memungkinkan Anda mengizinkan hanya perintah tertentu — misalnya hanya pengambilan berkas dari satu direktori, bukan akses penuh ke seluruh sistem berkas. Tanpa itu, seorang pengguna FTP yang sah bisa menjelajah seluruh QSYS.LIB.

Alternatif yang lebih baik: ganti FTP dengan SFTP lewat SSH, yang sudah terenkripsi dan lebih mudah dibatasi.

```
# SFTP terbatas pada satu direktori, di sshd_config
Match User mitraftp
    ChrootDirectory /interface/mitra
    ForceCommand internal-sftp
    AllowTcpForwarding no
```

69.5 HTTP Server

IBM i menjalankan Apache sebagai HTTP server bawaan. Konsol pengelolaannya ada di instans ADMIN pada port 2001.

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
ENDTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
STRTCPSVR SERVER(*HTTP) HTTPSVR(ADMIN)
```

Berkas konfigurasi ada di IFS pada `/www/webprod/conf/httpd.conf`, dan lognya di `/www/webprod/logs/`. Contoh konfigurasi yang diperkeras:

```
Listen 10.20.30.50:443
ServerName webprod.bank.co.id

# Sembunyikan informasi versi
ServerTokens Prod
ServerSignature Off
TraceEnable Off

# Paksa TLS
SSLEngine On
SSLAppName QIBM_HTTP_SERVER_WEBPROD
SSLProtocolEnable TLSV1.3
SSLProtocolEnable TLSV1.2
SSLProtocolDisable TLSV1.1
SSLProtocolDisable TLSV1
SSLProtocolDisable SSLV3

# Header keamanan
Header always set Strict-Transport-Security max-age=31536000
Header always set X-Content-Type-Options nosniff
Header always set X-Frame-Options DENY

# Tolak akses ke direktori sistem
<Directory />
    Require all denied
    Options None
    AllowOverride None
</Directory>

<Directory /www/webprod/htdocs>
    Require all granted
    Options -Indexes -Includes
</Directory>

# Batas ukuran dan waktu
LimitRequestBody 10485760
Timeout 60
KeepAliveTimeout 15

# Pencatatan
CustomLog logs/access_log combined
ErrorLog logs/error_log
LogLevel warn
```

Tiga baris yang sering terlupakan: `Options -Indexes` mencegah daftar isi direktori terbuka, `ServerTokens Prod` menyembunyikan versi yang membantu penyerang, dan `TraceEnable Off` menutup metode HTTP yang tidak pernah dibutuhkan tetapi sering menjadi temuan pemindaian keamanan.

```
# Memantau log akses untuk pola mencurigakan
cd /www/webprod/logs
grep -c ' 404 ' access_log
awk '{print $1}' access_log | sort | uniq -c | sort -rn | head -20
```

69.6 Integrated Web Services

IWS membungkus program RPG atau CL menjadi layanan REST atau SOAP. Yang perlu diperhatikan System Engineer:

Aspek	Pertimbangan
Jumlah instans server	Tiap instans memakan memori dan koneksi database
Subsystem penempatan	Pisahkan dari pool transaksional
Autentikasi	Wajib; jangan pernah biarkan terbuka
Batas ukuran permintaan	Cegah permintaan raksasa membekukan job
Timeout	Selaraskan dengan timeout klien
Pencatatan	Simpan di luar sistem untuk analisis

```
-- Job IWS dan konsumsi sumber dayanya
SELECT JOB_NAME, SUBSYSTEM, JOB_STATUS,
       ELAPSED_CPU_PERCENTAGE, TEMPORARY_STORAGE,
       ELAPSED_TOTAL_DISK_IO_COUNT
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         SUBSYSTEM_LIST_FILTER => 'QHTTPSVR'))
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC;
```

69.7 NetServer

NetServer menyediakan berbagi berkas gaya Windows terhadap IFS.

```
SELECT SERVER_SHARE_NAME, PATH_NAME, PERMISSION,
       MAXIMUM_USERS, CURRENT_USERS, TEXT_DESCRIPTION
  FROM QSYS2.SERVER_SHARE_INFO
 WHERE SHARE_TYPE = 'FILE';
```

Risiko yang perlu ditutup. Share yang menunjuk ke `/QSYS.LIB` memberi akses ke seluruh objek sistem lewat protokol berbagi berkas. Tinjau setiap share, dan pastikan tidak ada yang menunjuk ke akar sistem berkas.

69.8 Daftar periksa layanan jaringan

- ☐ Seluruh server yang tidak dipakai dimatikan, termasuk `AUTOSTART`
- ☐ Telnet disetel `ALWSSL(*ONLY)`
- ☐ FTP dimatikan, atau `ALWSSL(*ONLY)` dengan exit program
- ☐ SSH memakai autentikasi kunci untuk akun otomasi
- ☐ HTTP server memakai TLS 1.2 ke atas saja
- ☐ Header keamanan HTTP terpasang
- ☐ Daftar isi direktori dimatikan
- ☐ NetServer tidak berbagi akar sistem berkas
- ☐ Seluruh port terbuka tercatat dan disetujui
- ☐ Log akses dikirim ke luar sistem
- ☐ Sertifikat dipantau masa berlakunya
- ☐ Exit program terpasang pada ODBC, FTP, dan berbagi berkas

🌐 Bab 70. Pemantauan Jaringan Berkelanjutan

70.1 Metrik yang direkam

```
CREATE TABLE APPWRK.NET_MONITOR (
    WAKTU      TIMESTAMP   NOT NULL,
    METRIK     VARCHAR(40) NOT NULL,
    NILAI      DECIMAL(15,2),
    KETERANGAN VARCHAR(200)
);
```

```
PGM
   /* Jumlah koneksi per status */
   RUNSQL SQL('INSERT INTO APPWRK.NET_MONITOR +
      SELECT CURRENT TIMESTAMP, ''CONN_'' || TCP_STATE, COUNT(*), '''' +
        FROM QSYS2.NETSTAT_INFO GROUP BY TCP_STATE') COMMIT(*NONE)

   /* Kesalahan pada interface */
   RUNSQL SQL('INSERT INTO APPWRK.NET_MONITOR +
      SELECT CURRENT TIMESTAMP, ''ERR_'' || LINE_DESCRIPTION, +
             RECEIVE_ERRORS + TRANSMIT_ERRORS, '''' +
        FROM QSYS2.NETSTAT_INTERFACE_INFO') COMMIT(*NONE)

   /* Interface yang tidak aktif padahal seharusnya */
   RUNSQL SQL('INSERT INTO APPWRK.NET_MONITOR +
      SELECT CURRENT TIMESTAMP, ''IFC_DOWN'', COUNT(*), '''' +
        FROM QSYS2.NETSTAT_INTERFACE_INFO +
       WHERE INTERFACE_STATUS <> ''ACTIVE''') COMMIT(*NONE)
ENDPGM
```

70.2 Uji jangkauan mitra secara terjadwal

```
PGM
   PING RMTSYS('10.50.60.70') NBRPKT(3) WAITTIME(2)
   MONMSG MSGID(CPF0000 TCP0000) EXEC(DO)
      SNDMSG MSG('Mitra switching tidak terjangkau') TOUSR(SYSENG)
      RUNSQL SQL('INSERT INTO APPWRK.NET_MONITOR VALUES( +
                  CURRENT TIMESTAMP, ''PING_SWITCHING'', 0, +
                  ''Tidak terjangkau'')') COMMIT(*NONE)
   ENDDO
ENDPGM
```

Jalankan tiap lima menit. Mengetahui bahwa jalur ke mitra pembayaran putus sebelum transaksi nasabah mulai gagal adalah selisih antara insiden kecil dan insiden besar.

70.3 Baseline jaringan

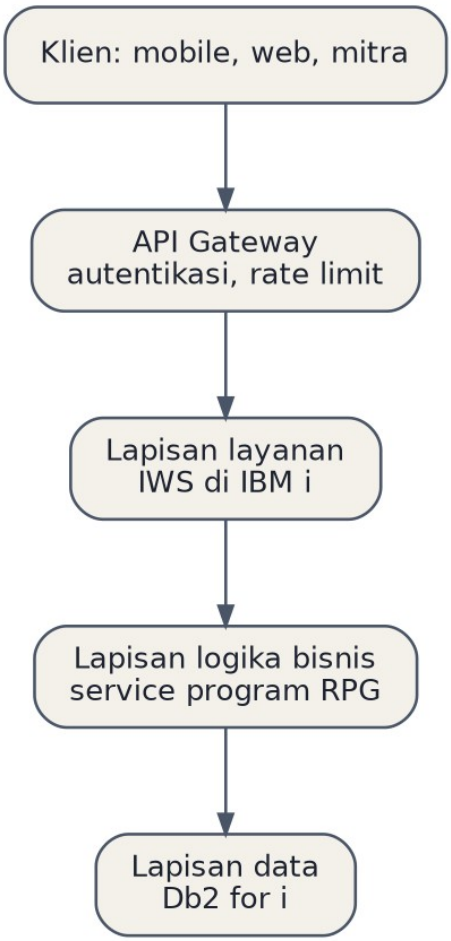
```
PGM
   /* Interface */
   RUNSQL SQL('CREATE TABLE BASELINE/NET_IFC AS +
      (SELECT CURRENT DATE AS TGL, x.* +
         FROM QSYS2.NETSTAT_INTERFACE_INFO x) WITH DATA') COMMIT(*NONE)

   /* Rute */
   RUNSQL SQL('CREATE TABLE BASELINE/NET_RTE AS +
      (SELECT CURRENT DATE AS TGL, x.* +
         FROM QSYS2.NETSTAT_ROUTE_INFO x) WITH DATA') COMMIT(*NONE)

   /* Port yang mendengarkan */
   RUNSQL SQL('CREATE TABLE BASELINE/NET_PORT AS +
      (SELECT CURRENT DATE AS TGL, LOCAL_PORT, LOCAL_PORT_NAME, +
              PROTOCOL, JOB_NAME +
         FROM QSYS2.NETSTAT_INFO +
        WHERE TCP_STATE = ''LISTEN'') WITH DATA') COMMIT(*NONE)
ENDPGM
```

Perbandingan antar bulan menjawab pertanyaan yang selalu muncul saat insiden: apakah ada yang berubah di jaringan sejak terakhir kali ini berjalan normal? Tanpa baseline, jawabannya selalu dugaan.

🚀 Bagian XVIII — Modernisasi Aplikasi

🏦 Bab 71. Merancang REST API di atas Core Banking

71.1 Lapisan yang harus dipisahkan



Gambar 79

Kesalahan yang paling merugikan adalah membiarkan API menyentuh tabel langsung. Ketika struktur tabel berubah — dan ia akan berubah — setiap konsumen API ikut rusak. Lapisan service program menjadi kontrak yang stabil di antaranya.

71.2 Prinsip perancangan

Prinsip	Alasan di konteks perbankan
Versi di jalur URL	Perubahan tidak memutus konsumen lama
Idempotensi pada operasi tulis	Retry klien tidak menghasilkan dobel transaksi
Batas jumlah hasil selalu ada	Satu permintaan tidak bisa menarik seluruh tabel
Timeout eksplisit	Job tidak menggantung menahan sumber daya
Korelasi ID di tiap permintaan	Penelusuran lintas sistem saat insiden

Prinsip	Alasan di konteks perbankan
Tidak pernah mengembalikan data lebih dari yang diminta	Pembatasan paparan data nasabah

71.3 Program layanan sebagai dasar API

```
**free
ctl-opt nomain option(*srcstmt: *nodebugio);

dcl-ds dsSaldoResp qualified template;
  noRek      char(16);
  namaSingkat varchar(30);
  saldo      packed(17:2);
  mataUang   char(3);
  status     char(1);
  kodeHasil  char(4);
  pesan      varchar(100);
end-ds;

dcl-proc apiCekSaldo export;
  dcl-pi *n;
    pNoRek   char(16)              const;
    pKorelasi varchar(40)          const;
    pResp    likeds(dsSaldoResp);
  end-pi;

  clear pResp;
  pResp.noRek = pNoRek;

  // Validasi bentuk masukan sebelum menyentuh basis data
  if %len(%trim(pNoRek)) <> 16 or %check('0123456789': pNoRek) > 0;
     pResp.kodeHasil = 'E001';
     pResp.pesan = 'Format nomor rekening tidak sah';
     catatApi(pKorelasi: 'E001');
     return;
  endif;

  exec sql
    SELECT LEFT(NAMA, 30), SALDO, MATA_UANG, STATUS
      INTO :pResp.namaSingkat, :pResp.saldo,
           :pResp.mataUang, :pResp.status
      FROM DTALIB.REKENING
     WHERE NO_REK = :pNoRek;

  select;
    when sqlcode = 0;
      pResp.kodeHasil = '0000';
      pResp.pesan = 'Berhasil';
    when sqlcode = 100;
      pResp.kodeHasil = 'E404';
      pResp.pesan = 'Rekening tidak ditemukan';
    other;
      pResp.kodeHasil = 'E500';
      pResp.pesan = 'Kesalahan sistem';
      catatKesalahan(pKorelasi: sqlcode);
  endsl;

  catatApi(pKorelasi: pResp.kodeHasil);
end-proc;
```

Perhatikan tiga hal yang membuat prosedur ini layak produksi: validasi bentuk masukan sebelum menyentuh basis data, kode hasil yang eksplisit alih-alih mengandalkan pengecualian, dan pencatatan dengan korelasi ID di setiap jalur keluar.

71.4 Operasi tulis dan idempotensi

```
**free
dcl-proc apiTransfer export;
  dcl-pi *n;
    pIdempotency varchar(64) const;
    pRekAsal     char(16)    const;
    pRekTujuan   char(16)    const;
    pJumlah      packed(17:2) const;
    pResp        likeds(dsTrxResp);
  end-pi;

  dcl-s wAda int(5);

  // Sudah pernah diproses? Kembalikan hasil yang sama.
  exec sql
    SELECT COUNT(*) INTO :wAda
      FROM DTALIB.API_IDEMPOTENCY
     WHERE IDEM_KEY = :pIdempotency;

  if wAda > 0;
     exec sql
       SELECT KODE_HASIL, NO_REF, PESAN
         INTO :pResp.kodeHasil, :pResp.noRef, :pResp.pesan
         FROM DTALIB.API_IDEMPOTENCY
        WHERE IDEM_KEY = :pIdempotency;
     return;   // tidak diproses ulang
  endif;

  // Catat kunci lebih dulu, di dalam transaksi yang sama
  exec sql
    INSERT INTO DTALIB.API_IDEMPOTENCY
      (IDEM_KEY, WAKTU, STATUS)
    VALUES (:pIdempotency, CURRENT TIMESTAMP, 'P');

  if sqlcode <> 0;
     // tabrakan: permintaan kembar sedang berjalan
     pResp.kodeHasil = 'E409';
     pResp.pesan = 'Permintaan sedang diproses';
     return;
  endif;

  monitor;
     prosesTransfer(pRekAsal: pRekTujuan: pJumlah: pResp);

     exec sql
       UPDATE DTALIB.API_IDEMPOTENCY
          SET STATUS = 'S', KODE_HASIL = :pResp.kodeHasil,
              NO_REF = :pResp.noRef, PESAN = :pResp.pesan
        WHERE IDEM_KEY = :pIdempotency;

     exec sql COMMIT;
  on-error *all;
     exec sql ROLLBACK;
     pResp.kodeHasil = 'E500';
```

```
     pResp.pesan = 'Transfer gagal, silakan ulangi';
  endmon;
end-proc;
```

Kunci idempotensi ditulis **di dalam transaksi yang sama** dengan pemindahan dana. Bila dicatat di luar transaksi, rollback akan menghapus jejaknya dan retry klien akan memproses ulang.

```
CREATE TABLE DTALIB.API_IDEMPOTENCY (
    IDEM_KEY    VARCHAR(64)  NOT NULL,
    WAKTU       TIMESTAMP    NOT NULL,
    STATUS      CHAR(1)      NOT NULL,
    KODE_HASIL  CHAR(4),
    NO_REF      VARCHAR(40),
    PESAN       VARCHAR(100),
    PRIMARY KEY (IDEM_KEY)
);

-- Pembersihan terjadwal; kunci tidak perlu disimpan selamanya
DELETE FROM DTALIB.API_IDEMPOTENCY
 WHERE WAKTU < CURRENT TIMESTAMP - 7 DAYS;
```

71.5 Kode hasil yang konsisten

Kode	Arti	HTTP padanan
`0000`	Berhasil	200
`E001`	Format masukan tidak sah	400
`E401`	Autentikasi gagal	401
`E403`	Tidak berwenang atas sumber daya	403
`E404`	Tidak ditemukan	404
`E409`	Konflik, permintaan kembar	409
`E422`	Aturan bisnis menolak	422
`E429`	Melewati batas laju	429
`E500`	Kesalahan sistem	500
`E503`	Layanan sedang tidak tersedia	503

Aturan yang menyelamatkan tim dukungan. `E422` dan `E500` harus dibedakan tegas. Yang pertama berarti permintaan diproses dengan benar dan ditolak oleh aturan bisnis — saldo tidak cukup, rekening diblokir. Yang kedua berarti sistem gagal. Menggabungkan keduanya membuat setiap saldo tidak cukup terlihat seperti gangguan sistem.

71.6 Pencatatan dan penelusuran

```
CREATE TABLE DTALIB.API_LOG (
    WAKTU        TIMESTAMP   NOT NULL,
    KORELASI_ID  VARCHAR(40) NOT NULL,
    NM_LAYANAN   VARCHAR(40) NOT NULL,
    KD_KONSUMEN  VARCHAR(20),
    KODE_HASIL   CHAR(4),
    DURASI_MS    INTEGER,
    JOB_NAME     VARCHAR(50)
);

CREATE INDEX DTALIB.API_LOG_IX1 ON DTALIB.API_LOG (WAKTU, NM_LAYANAN);
CREATE INDEX DTALIB.API_LOG_IX2 ON DTALIB.API_LOG (KORELASI_ID);
```

```
-- Kesehatan API per jam
SELECT NM_LAYANAN, HOUR(WAKTU) AS jam,
       COUNT(*) AS total,
       SUM(CASE WHEN KODE_HASIL = '0000' THEN 1 ELSE 0 END) AS berhasil,
       SUM(CASE WHEN KODE_HASIL LIKE 'E5%' THEN 1 ELSE 0 END) AS gagal_sistem,
       DECIMAL(AVG(DURASI_MS), 9, 1) AS rata_ms,
       MAX(DURASI_MS) AS terlama_ms
  FROM DTALIB.API_LOG
 WHERE WAKTU >= CURRENT DATE
 GROUP BY NM_LAYANAN, HOUR(WAKTU)
 ORDER BY gagal_sistem DESC, rata_ms DESC;

-- Konsumen yang paling banyak memanggil
SELECT KD_KONSUMEN, NM_LAYANAN, COUNT(*) AS jml
  FROM DTALIB.API_LOG
 WHERE WAKTU >= CURRENT TIMESTAMP - 1 HOUR
 GROUP BY KD_KONSUMEN, NM_LAYANAN
 ORDER BY jml DESC
 FETCH FIRST 20 ROWS ONLY;
```

Korelasi ID memungkinkan satu permintaan ditelusuri dari aplikasi mobile, lewat gateway, sampai ke job IBM i. Tanpa itu, investigasi keluhan nasabah berubah menjadi pencocokan stempel waktu secara manual.

Bab 72. Open Source di IBM i

72.1 Menyiapkan lingkungan

```
# Dari SSH atau QP2TERM
yum update
yum install nodejs20 python3 git bash coreutils-gnu
yum install python3-itoolkit python3-ibm_db

# Memastikan jalur
export PATH=/QOpenSys/pkgs/bin:$PATH
echo 'export PATH=/QOpenSys/pkgs/bin:$PATH' >> ~/.profile

node --version
python3 --version
git --version
```

72.2 Perbedaan yang harus dipahami System Engineer

Aspek	Job IBM i biasa	Job PASE
Terlihat di `WRKACTJOB`	Ya, dengan nama jelas	Sebagai `QP0ZSPWP`
Subsystem bawaan	Sesuai konfigurasi	`QUSRWRK`
Manajemen memori	Single-level storage	Model AIX
Penanganan sinyal	Message queue	Sinyal Unix
Pencatatan	Job log	stdout, stderr, berkas log
Penghentian	`ENDJOB`	`ENDJOB` atau `kill`

```
-- Menemukan job PASE dan apa yang dijalankannya
SELECT JOB_NAME, AUTHORIZATION_NAME, SUBSYSTEM,
       ELAPSED_CPU_PERCENTAGE, TEMPORARY_STORAGE,
       FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 WHERE JOB_NAME LIKE '%QP0ZSPWP%'
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC;
```

Rekomendasi penempatan. Buat subsystem terpisah untuk beban open source, dengan memory pool sendiri. Aplikasi Node.js yang bocor memori tidak boleh bisa menggerus pool transaksional.

```
CRTSBSD SBSD(QGPL/OSSSBS) POOLS((1 *BASE) (2 *SHRPOOL4)) +
        TEXT('Beban open source')
CHGSHRPOOL POOL(*SHRPOOL4) SIZE(4000000) ACTLVL(10)
STRSBS SBSD(QGPL/OSSSBS)
```

72.3 Python untuk otomasi operasional

```
#!/QOpenSys/pkgs/bin/python3
import ibm_db_dbi as dbi
import sys
from datetime import datetime

def cek_kesehatan():
    conn = dbi.connect()
    cur = conn.cursor()
    masalah = []

    # Pemakaian ASP
    cur.execute('''
        SELECT SYSTEM_ASP_USED, AVERAGE_CPU_UTILIZATION
          FROM TABLE(QSYS2.SYSTEM_STATUS())
    ''')
    asp, cpu = cur.fetchone()
    if asp > 85:
        masalah.append(f'ASP terpakai {asp}%')
    if cpu > 85:
        masalah.append(f'CPU {cpu}%')

    # Job menunggu jawaban
    cur.execute('''
        SELECT COUNT(*) FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
            JOB_STATUS_FILTER => '*MSGW'))
    ''')
    msgw = cur.fetchone()[0]
    if msgw > 0:
        masalah.append(f'{msgw} job menunggu jawaban')

    # Progres EOD
    cur.execute('''
        SELECT COUNT(*) FROM APPWRK.EOD_KONTROL
         WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'E'
    ''')
    gagal = cur.fetchone()[0]
    if gagal > 0:
        masalah.append(f'{gagal} fase EOD gagal')

    conn.close()
    return masalah

if __name__ == '__main__':
```

```
    hasil = cek_kesehatan()
    if hasil:
        print(f'{datetime.now()} PERHATIAN:')
        for m in hasil:
            print(f'  - {m}')
        sys.exit(1)
    print(f'{datetime.now()} Semua normal')
```

Skrip semacam ini dijadwalkan lewat `ADDJOBSCDE` yang memanggil `QSH` atau lewat cron di PASE. Keluarannya bisa dikirim ke sistem pemantauan luar dengan mudah, yang sulit dilakukan dari CL murni.

72.4 Node.js untuk layanan ringan

```
const express = require('express');
const odbc = require('odbc');

const app = express();
app.use(express.json());

let pool;
(async () => {
  pool = await odbc.pool({
    connectionString: 'DSN=*LOCAL',
    initialSize: 3,
    maxSize: 10
  });
})();

app.get('/api/v1/rekening/:norek/saldo', async (req, res) => {
  const korelasi = req.header('X-Correlation-Id') || 'none';
  const norek = req.params.norek;

  if (!/^\d{16}$/.test(norek)) {
    return res.status(400).json({ kode: 'E001', pesan: 'Format tidak sah' });
  }

  try {
    const hasil = await pool.query(
      'SELECT SALDO, STATUS FROM DTALIB.REKENING WHERE NO_REK = ?',
      [norek]
    );
    if (hasil.length === 0) {
      return res.status(404).json({ kode: 'E404', pesan: 'Tidak ditemukan' });
    }
    res.json({ kode: '0000', saldo: hasil[0].SALDO, status: hasil[0].STATUS });
  } catch (err) {
    console.error(`[${korelasi}] ${err.message}`);
    res.status(500).json({ kode: 'E500', pesan: 'Kesalahan sistem' });
  }
});

app.listen(8081, () => console.log('Layanan berjalan di port 8081'));
```

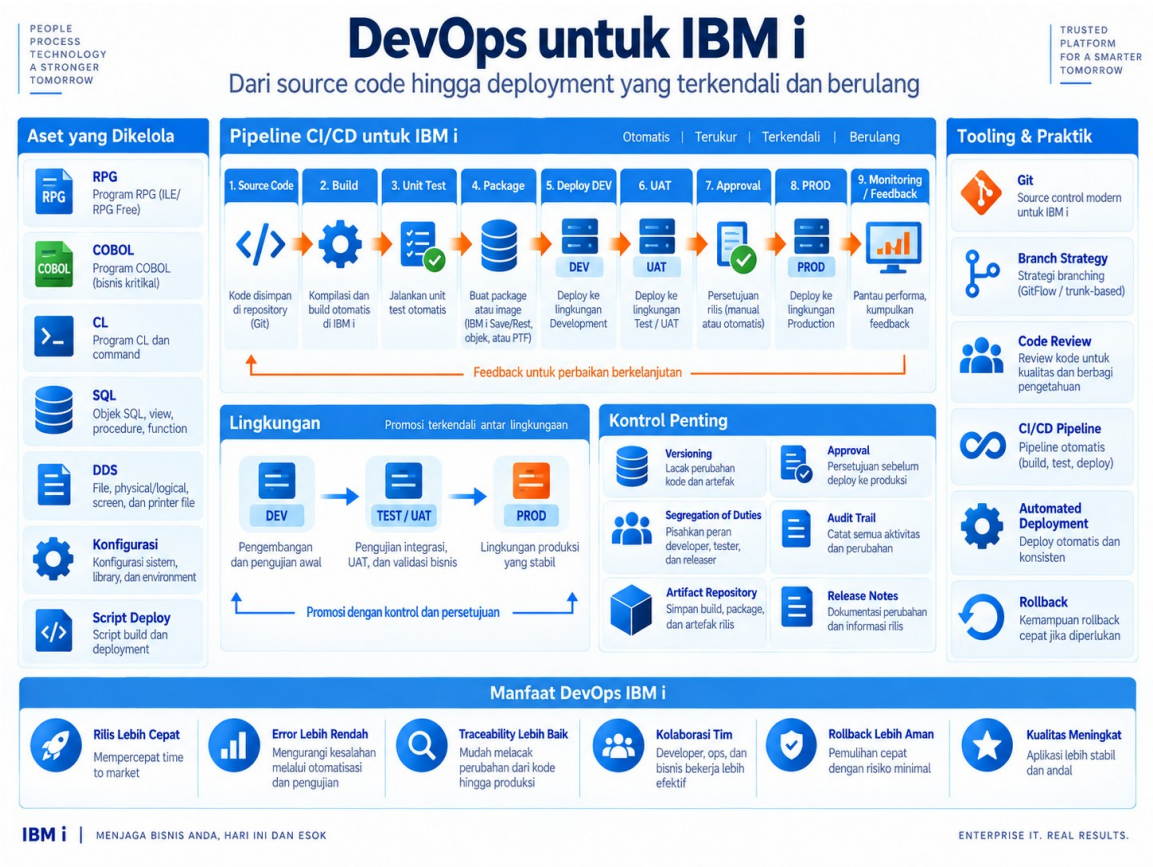
Peringatan yang perlu disampaikan ke tim pengembang. Connection pool yang tidak dibatasi akan membuka ratusan job `QZDASOINIT` di IBM i. Tetapkan `maxSize` yang wajar, dan pantau jumlah job server database.

```
-- Berapa banyak job server database yang aktif
SELECT COUNT(*) AS jml_qzdasoinit
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO())
 WHERE JOB_NAME LIKE '%QZDASOINIT%';
```

72.5 Batas yang perlu ditetapkan

Aturan	Alasan
Open source tidak menyentuh tabel produksi langsung	Lewat service program atau API
Pustaka pihak ketiga harus melewati tinjauan	Rantai pasok perangkat lunak adalah risiko nyata
Versi runtime dikunci dan dicatat	Pembaruan tidak sengaja memutus layanan
Berjalan di subsystem dan pool terpisah	Isolasi sumber daya
Log dikirim ke luar sistem	PASE tidak menulis ke job log
Akun layanan terpisah dengan wewenang minimal	Bukan memakai profil administrator

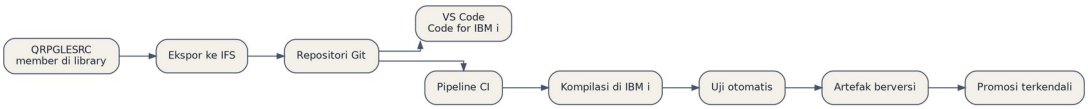
🚀 Bab 73. DevOps untuk IBM i



Gambar 80 — Pipeline CI/CD dan kontrol untuk IBM i

73.1 Perpindahan dari source physical file ke Git

Ini perubahan tata kelola, bukan sekadar perubahan alat. Selama kode sumber tinggal di `QRPGLESRC`, pertanyaan "siapa mengubah baris ini dan kenapa" tidak punya jawaban yang bisa diaudit.



Gambar 81

Langkah migrasi yang aman.

```
/* Ekspor seluruh member ke IFS sebagai berkas teks */
CPYTOSTMF FROMMBR('/QSYS.LIB/APPSRC.LIB/QRPGLESRC.FILE/HITBUNGA.MBR') +
          TOSTMF('/src/rpgle/hitbunga.rpgle') +
          STMFOPT(*REPLACE) +
          STMFCCSID(1208) +
          ENDLINFMT(*LF)

/* Kebalikannya, bila perlu kembali ke member */
CPYFRMSTMF FROMSTMF('/src/rpgle/hitbunga.rpgle') +
           TOMBR('/QSYS.LIB/APPSRC.LIB/QRPGLESRC.FILE/HITBUNGA.MBR') +
           MBROPT(*REPLACE)
```

CCSID 1208 adalah UTF-8. Ini penting: berkas yang diekspor dengan CCSID salah akan tampak rusak di Git dan di editor modern.

```
# Skrip ekspor massal, dijalankan sekali saat migrasi
#!/QOpenSys/pkgs/bin/bash
LIB=APPSRC
for SRCF in QRPGLESRC QCLLESRC QSQLSRC QDDSSRC; do
  mkdir -p /src/$(echo $SRCF | tr 'A-Z' 'a-z')
  system "DSPFD FILE($LIB/$SRCF) TYPE(*MBRLIST) OUTPUT(*OUTFILE) \
          OUTFILE(QTEMP/MBRS)"
  # baca daftar member lalu salin satu per satu
done
```

73.2 Struktur repositori

```
/src
  /rpgle          # program RPG bebas format
  /sqlrpgle       # RPG dengan embedded SQL
  /clle           # program CL
  /dds            # definisi file lama
  /sql            # DDL, view, fungsi, prosedur
  /binder         # binder source untuk service program
/tests            # skrip uji
/build
  build.sh        # skrip kompilasi
  targets.json    # daftar objek dan urutan bangun
/docs
  runbook/        # runbook terkait aplikasi ini
  CHANGELOG.md
.gitignore
README.md
```

73.3 Skrip build

```
#!/QOpenSys/pkgs/bin/bash
set -euo pipefail

LIB=${1:?Library target wajib diisi}
SRC=${2:-/home/builder/src}
GAGAL=0
```

```
echo "Build ke library $LIB dimulai $(date)"

system "CRTLIB LIB($LIB) TEXT('Build otomatis')" || true

# 1. Objek basis data lebih dulu
for f in $SRC/sql/*.sql; do
  echo "DDL: $(basename $f)"
  system "RUNSQLSTM SRCSTMF('$f') COMMIT(*NONE) NAMING(*SQL)" || GAGAL=1
done

# 2. Modul dan service program
for f in $SRC/sqlrpgle/svc*.sqlrpgle; do
  NAMA=$(basename "$f" .sqlrpgle | tr 'a-z' 'A-Z')
  echo "Modul: $NAMA"
  system "CRTSQLRPGI OBJ($LIB/$NAMA) SRCSTMF('$f') OBJTYPE(*MODULE) \
          DBGVIEW(*SOURCE) COMMIT(*CHG) RPGPPOPT(*LVL2)" || GAGAL=1
done

# 3. Program
for f in $SRC/rpgle/*.rpgle; do
  NAMA=$(basename "$f" .rpgle | tr 'a-z' 'A-Z')
  echo "Program: $NAMA"
  system "CRTBNDRPG PGM($LIB/$NAMA) SRCSTMF('$f') DBGVIEW(*SOURCE) \
          OPTION(*EVENTF) REPLACE(*YES) DFTACTGRP(*NO) ACTGRP('APPGRP')" \
    || GAGAL=1
done

# 4. Program CL
for f in $SRC/clle/*.clle; do
  NAMA=$(basename "$f" .clle | tr 'a-z' 'A-Z')
  echo "CL: $NAMA"
  system "CRTBNDCL PGM($LIB/$NAMA) SRCSTMF('$f') DBGVIEW(*SOURCE) \
          REPLACE(*YES)" || GAGAL=1
done

if [ $GAGAL -ne 0 ]; then
  echo "BUILD GAGAL"
  exit 1
fi

echo "Build selesai $(date)"
```

Mengapa urutannya begitu. Objek basis data harus ada sebelum program yang memakainya dikompilasi, dan service program harus ada sebelum program yang terikat padanya. Urutan yang salah menghasilkan kegagalan kompilasi yang terlihat misterius.

73.4 Membaca hasil kompilasi secara terprogram

Opsi `*EVENTF` menghasilkan berkas event yang berisi kesalahan kompilasi dalam bentuk terstruktur.

```
-- Kesalahan kompilasi dari EVFEVENT
SELECT * FROM TABLE(QSYS2.IFS_READ(
    PATH_NAME => '/QSYS.LIB/DEVPGM.LIB/EVFEVENT.FILE/HITBUNGA.MBR'));
```

Alat seperti Code for IBM i membaca berkas ini dan menandai kesalahan langsung di editor. Dalam pipeline, ia dipakai untuk menghasilkan laporan kegagalan yang bisa dibaca mesin.

73.5 Uji otomatis

Ini bagian yang paling sering dilewati, dan paling menentukan apakah otomasi Anda sungguh berguna.

```
**free
// Uji unit sederhana untuk prosedur hitungBunga
ctl-opt dftactgrp(*no) actgrp('TESTGRP') bnddir('APPBND');

/copy APPSRC/QRPGLEREF,SVCBUNGA_H

dcl-s wGagal int(5) inz(0);

// Kasus 1: perhitungan normal
ujiSama(hitungBunga(1000000: 'DP01': 365):
        13.70: 'Bunga harian saldo 1 juta rate 5%');

// Kasus 2: saldo nol
ujiSama(hitungBunga(0: 'DP01': 365): 0: 'Saldo nol');

// Kasus 3: produk tidak dikenal harus melempar kesalahan
monitor;
   hitungBunga(1000000: 'XXXX': 365);
   catatGagal('Produk tidak dikenal seharusnya gagal');
on-error *all;
   catatLulus('Produk tidak dikenal ditolak dengan benar');
endmon;

if wGagal > 0;
   sndEscape('Uji gagal: ' + %char(wGagal) + ' kasus');
endif;

*inlr = *on;
```

```
# Dijalankan pipeline setelah build
system "CALL PGM(DEVPGM/TSTBUNGA)"
if [ $? -ne 0 ]; then
  echo "UJI GAGAL, deploy dibatalkan"
  exit 1
fi
```

Mulai dari mana bila belum ada uji sama sekali. Jangan mencoba menutupi seluruh kode lawas. Mulai dari tiga tempat: prosedur perhitungan finansial, logika yang pernah menyebabkan insiden, dan kode yang akan Anda ubah berikutnya. Ini memberi nilai tertinggi per jam yang diinvestasikan.

73.6 Pipeline lengkap

```
# Contoh definisi pipeline; sintaks menyesuaikan alat CI Anda
stages:
  - validasi
  - build
  - uji
  - paket
  - deploy-uat
  - deploy-prod

validasi:
  stage: validasi
  script:
```

```
    - ./scripts/cek-format.sh
    - ./scripts/cek-rahasia.sh   # cari password yang tertinggal di kode

build:
  stage: build
  script:
    - ssh builder@ibmi "cd /home/builder && ./build/build.sh BUILD$CI_ID"
  artifacts:
    paths: [build/laporan-build.txt]

uji:
  stage: uji
  script:
    - ssh builder@ibmi "system 'CALL PGM(BUILD$CI_ID/TSTSUITE)'"

paket:
  stage: paket
  script:
    - ssh builder@ibmi "system 'SAVLIB LIB(BUILD$CI_ID) DEV(*SAVF) \
        SAVF(ARTIFACT/REL$CI_ID)'"

deploy-uat:
  stage: deploy-uat
  when: manual
  script:
    - ./scripts/promosi.sh UAT $CI_ID

deploy-prod:
  stage: deploy-prod
  when: manual
  only: [main]
  script:
    - ./scripts/cek-approval.sh $CI_ID   # verifikasi tiket CAB disetujui
    - ./scripts/promosi.sh PROD $CI_ID
```

Tahap `cek-rahasia` layak ada sejak awal. Password yang tertinggal di kode sumber adalah temuan audit yang berulang, dan pemeriksaan otomatis jauh lebih andal daripada tinjauan manual.

73.7 Batas yang harus dijaga

Praktik DevOps umum	Penyesuaian untuk core banking
Deploy otomatis ke produksi	Tidak; selalu manual dengan persetujuan CAB
Deploy beberapa kali sehari	Mingguan sudah agresif
Rollback dengan mengganti image	Salinan objek lama, diuji prosedurnya
Lingkungan sekali pakai	Partisi mahal; lingkungan bersifat tetap
Uji di produksi	Tidak pernah
Pengembang punya akses produksi	Tidak; pemisahan tugas

Mengambil kendali versi, pipeline build, dan uji otomatis dari dunia DevOps adalah kemajuan nyata. Mengambil deploy kontinu ke produksi perbankan bukan kemajuan, melainkan pemindahan risiko dari proses ke keberuntungan.

🚀 Bab 74. Mengukur Keberhasilan Modernisasi

74.1 Ukuran yang bermakna

Ukuran	Cara mengukur	Arah yang diinginkan
Waktu dari permintaan sampai produksi	Tiket sampai deploy	Turun
Tingkat kegagalan perubahan	Deploy yang butuh rollback	Turun
Waktu pemulihan setelah kegagalan	Insiden sampai pulih	Turun
Cakupan uji otomatis	Prosedur kritikal yang punya uji	Naik
Program yang masih fixed format	Hitungan objek	Turun
Ketergantungan pada satu orang	Area tanpa cadangan	Turun

```
-- Berapa banyak program yang masih fixed format
SELECT COUNT(*) AS total_program,
       SUM(CASE WHEN SOURCE_FILE_MEMBER IS NOT NULL THEN 1 ELSE 0 END)
         AS dari_member,
       SUM(CASE WHEN SOURCE_STREAM_FILE_PATH IS NOT NULL THEN 1 ELSE 0 END)
         AS dari_ifs
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM';
```

Kolom `SOURCE_STREAM_FILE_PATH` yang terisi menandakan program dikompilasi dari IFS, yang berarti sumbernya kemungkinan besar sudah ada di Git. Rasio antara keduanya adalah ukuran kemajuan migrasi yang paling jujur.

74.2 Yang tidak perlu dimodernisasi

Tidak semua kode lawas layak disentuh. Kriteria untuk membiarkannya:

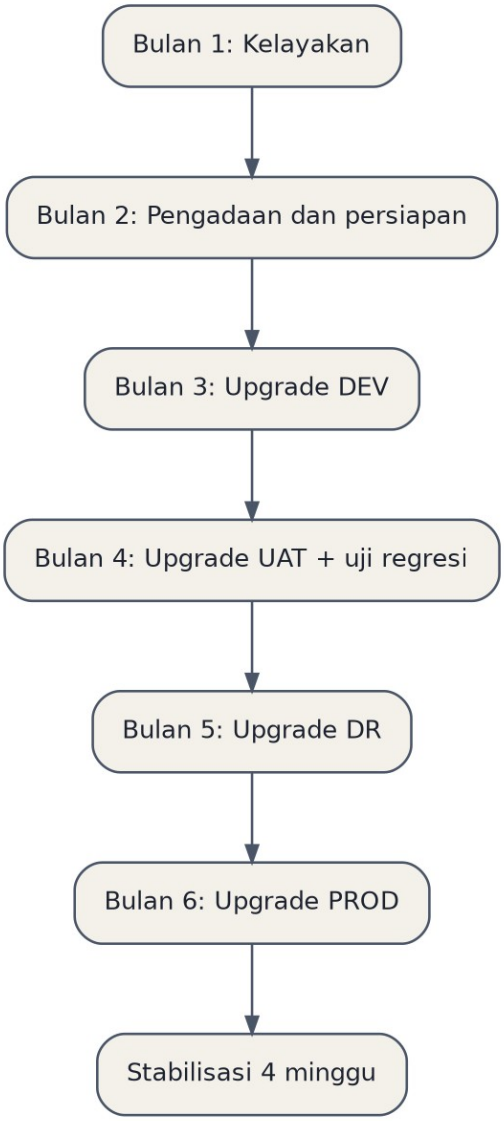
1. Berjalan stabil bertahun-tahun tanpa insiden
2. Tidak akan berubah karena kebutuhan bisnis
3. Kinerjanya memadai
4. Tidak menghalangi perubahan lain

Kode yang memenuhi keempatnya lebih baik dibiarkan dan didokumentasikan. Menyentuh kode yang bekerja demi kerapian adalah cara paling umum memperkenalkan cacat baru ke sistem yang stabil.

📦 Bagian XIX — Manajemen Proyek Teknis

📦 Bab 75. Proyek Upgrade Rilis OS

75.1 Garis waktu enam bulan



Gambar 82

75.2 Bulan 1 — kelayakan

Pertanyaan yang harus terjawab tertulis sebelum melangkah:

Pertanyaan	Sumber jawaban
Apakah perangkat keras mendukung rilis target?	Matriks dukungan IBM
Apakah vendor aplikasi mendukung rilis target?	Surat resmi vendor, bukan lisan

Pertanyaan	Sumber jawaban
Apakah seluruh perangkat lunak pihak ketiga kompatibel?	Vendor masing-masing
Apakah ada program yang perlu konversi?	Analisis objek program
Berapa kapasitas disk yang dibutuhkan?	Dokumentasi rilis
Berapa lama jendela yang dibutuhkan?	Uji di DEV

```
/* Analisis kesiapan objek program */
ANZOBJCVN LIB(*ALLUSR) OPTION(*OUTFILE) +
          OUTFILE(QTEMP/CVNANZ) +
          OUTMBR(*FIRST *REPLACE)
```

```
-- Objek yang butuh perhatian sebelum upgrade
SELECT OBJLIB, OBJNAME, OBJTYPE, CVNSTS, CVNSTSTXT
  FROM QTEMP.CVNANZ
 WHERE CVNSTS <> '0'
 ORDER BY OBJLIB, OBJNAME;
```

Objek yang tidak punya data observasi tidak bisa dikonversi otomatis dan harus dikompilasi ulang dari sumber. Bila sumbernya sudah hilang — situasi yang lebih umum daripada yang diakui orang — inilah saatnya diketahui, bukan pada malam upgrade.

75.3 Uji regresi yang memadai

Cakupan	Cara
Seluruh menu aplikasi dibuka	Daftar periksa manual oleh pengguna kunci
Satu transaksi tiap jenis	Skenario uji tertulis
Satu siklus EOD lengkap	Dengan salinan data produksi termutakhir
Satu siklus EOM	Bila jadwalnya memungkinkan
Seluruh interface	Uji dengan mitra, terjadwal
Seluruh laporan	Bandingkan keluaran sebelum dan sesudah
Kinerja	Bandingkan durasi EOD sebelum dan sesudah

Perbandingan keluaran laporan adalah uji yang paling sering diabaikan dan paling sering menangkap masalah. Simpan keluaran dari lingkungan lama, jalankan proses yang sama di lingkungan baru, bandingkan baris demi baris.

```
# Membandingkan dua keluaran laporan
diff <(sort lama.txt) <(sort baru.txt) > selisih.txt
wc -l selisih.txt
```

75.4 Rencana rollback

Setiap upgrade produksi harus punya rencana rollback tertulis dengan **batas waktu keputusan**.

Waktu	Keputusan
Jam ke-0	Mulai, backup penuh sudah terverifikasi
Jam ke-4	Bila upgrade OS belum selesai, tinjau
Jam ke-6	Titik tanpa jalan kembali; setelah ini rollback berarti restore penuh
Jam ke-8	Bila verifikasi aplikasi gagal, putuskan rollback

Waktu	Keputusan
Jam ke-10	Batas akhir; layanan harus buka atau diumumkan tertunda

Titik tanpa jalan kembali harus ditentukan dan disepakati sebelum hari H. Keputusan rollback yang diambil setelah melewati titik itu jauh lebih mahal daripada melanjutkan dan memperbaiki.

📦 Bab 76. Proyek Migrasi Perangkat Keras

76.1 Pilihan metode

Metode	Cara	Durasi henti	Risiko
Save dan restore	`SAVSYS` + `SAVLIB` lalu restore di mesin baru	Panjang, 8–24 jam	Menengah
Migrasi berbantuan replikasi	Replikasi berjalan, lalu switchover	Sangat pendek	Rendah, tetapi kompleks
Pemindahan disk	Pindahkan fisik unit disk	Menengah	Tinggi bila gagal
Migrasi via HMC dan storage	Lepas LUN, pasang di mesin baru	Pendek	Menengah

Untuk core banking, metode berbantuan replikasi hampir selalu yang terbaik meskipun persiapannya paling panjang. Ia memberi kemampuan uji berulang tanpa mengganggu produksi, dan rollback berarti kembali menunjuk ke mesin lama.

76.2 Daftar periksa pra-migrasi

- ☐ Serial dan model mesin baru tercatat, lisensi dipindahkan
- ☐ Kapasitas disk mesin baru sama atau lebih besar
- ☐ Seluruh perangkat yang terhubung diinventarisasi: tape, printer, konsol
- ☐ Konfigurasi jaringan mesin baru disiapkan, termasuk VLAN di switch
- ☐ Alamat IP: tetap sama, atau perubahan sudah dikomunikasikan ke mitra
- ☐ Kunci lisensi perangkat lunak pihak ketiga terikat serial baru
- ☐ Sertifikat DCM diekspor
- ☐ Baseline lengkap mesin lama diambil dan disimpan
- ☐ Backup penuh diverifikasi dapat dibaca
- ☐ Jendela disepakati dengan bisnis dan regulator bila perlu
- ☐ Rencana rollback tertulis dan disetujui
- ☐ Uji coba penuh sudah dilakukan minimal sekali

Poin tentang kunci lisensi yang terikat serial layak diperhatikan. Perangkat lunak pihak ketiga yang lisensinya terikat nomor seri mesin akan berhenti bekerja di mesin baru, dan penerbitan lisensi baru dari vendor bisa memakan hari kerja — bukan jam.

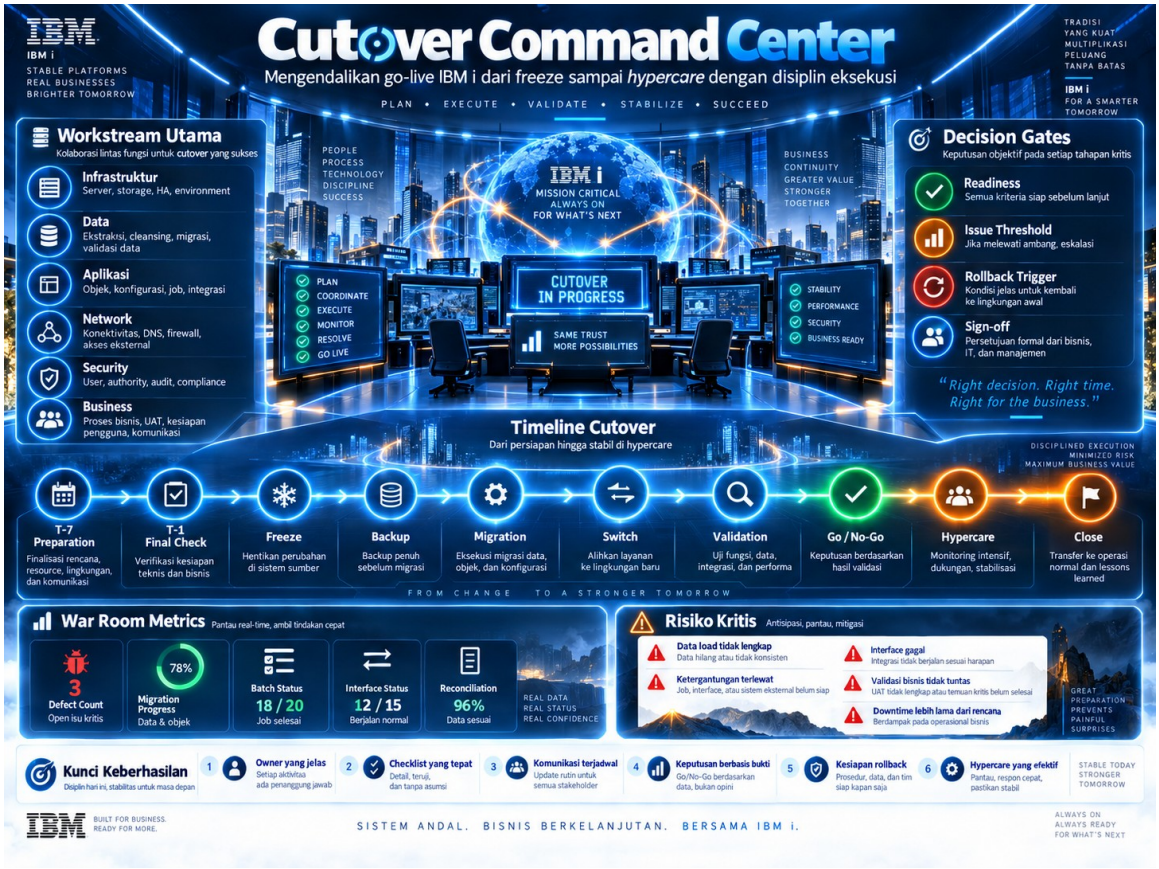
76.3 Verifikasi pasca-migrasi

```
PGM
   /* Bandingkan dengan baseline mesin lama */
   RUNSQL SQL('CREATE TABLE QTEMP/BANDING AS ( +
      SELECT ''OBJEK'' AS KATEGORI, COUNT(*) AS JUMLAH +
        FROM TABLE(QSYS2.OBJECT_STATISTICS(''*ALLUSR'',''*ALL'')) +
      UNION ALL +
      SELECT ''PROFIL'', COUNT(*) FROM QSYS2.USER_INFO +
      UNION ALL +
      SELECT ''PUSTAKA'', COUNT(DISTINCT OBJLONGSCHEMA) +
        FROM TABLE(QSYS2.OBJECT_STATISTICS(''*ALLUSR'',''*LIB'')) +
      ) WITH DATA') COMMIT(*NONE)
ENDPGM
```

Yang diverifikasi	Cara	Toleransi
Jumlah objek per library	Bandingkan dengan baseline	Nol selisih
Jumlah record tabel kunci	Hitung dan bandingkan	Nol selisih
Total saldo kontrol	Hitung dan bandingkan	Nol selisih
Jumlah user profile	Bandingkan	Nol selisih
System value	Bandingkan seluruhnya	Hanya yang disengaja
Konfigurasi journaling	Bandingkan daftar file dijurnal	Nol selisih
Konektivitas seluruh interface	Uji satu transaksi tiap kanal	Semua berhasil
Durasi EOD	Jalankan satu siklus	Tidak lebih lambat

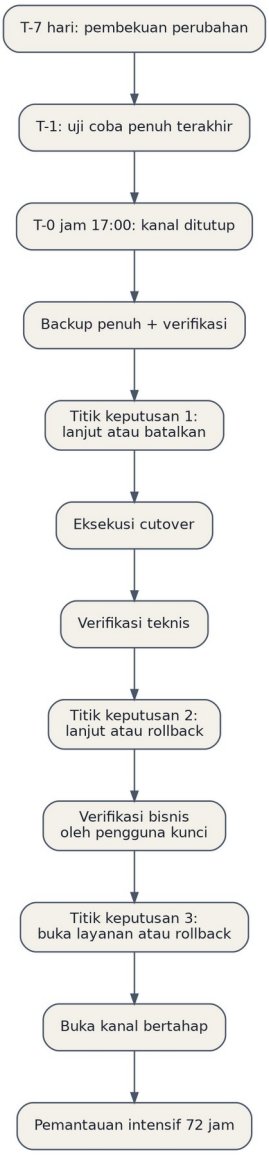
Toleransi nol selisih pada jumlah objek terdengar berlebihan sampai Anda menemukan satu library yang tidak ikut ter-restore karena tercatat di daftar pengecualian bertahun-tahun lalu.

Bab 77. Cutover



Gambar 83 — Pusat kendali cutover: dari pembekuan sampai hypercare

77.1 Anatomi malam cutover



Gambar 84

Tiga titik keputusan yang eksplisit adalah inti dari cutover yang terkendali. Masing-masing punya kriteria tertulis, penanggung jawab bernama, dan batas waktu.

77.2 Peran pada malam cutover

Peran	Tanggung jawab	Kewenangan
Pemimpin cutover	Mengambil keputusan lanjut atau batal	Penuh
System Engineer	Eksekusi teknis platform	Teknis
Application Support	Verifikasi aplikasi	Teknis aplikasi
DBA	Verifikasi integritas data	Teknis data
Perwakilan bisnis	Verifikasi fungsional	Menerima atau menolak
Komunikasi	Memberi kabar ke pemangku kepentingan	—

Peran	Tanggung jawab	Kewenangan
Pencatat	Mencatat setiap langkah dan waktunya	—

Peran pencatat sering dianggap sepele dan dihapus untuk menghemat orang. Itu kesalahan. Catatan waktu yang akurat adalah satu-satunya dasar untuk analisis pasca-kegiatan, dan bukti untuk auditor.

77.3 Kriteria buka layanan

Tulis ini sebelum hari H, dan jangan mengubahnya pada pukul empat pagi ketika semua orang lelah dan ingin pulang.

- ☐ Seluruh verifikasi teknis lulus tanpa pengecualian
- ☐ Jumlah record dan saldo kontrol cocok
- ☐ Satu transaksi berhasil dari setiap kanal
- ☐ Seluruh interface kritikal terhubung
- ☐ Tidak ada job dalam status MSGW atau gagal
- ☐ Perwakilan bisnis menyatakan menerima, tertulis
- ☐ Tim dukungan siap di tempat untuk 72 jam berikutnya
- ☐ Rencana rollback masih dapat dieksekusi bila dibutuhkan

77.4 Pemantauan pasca-cutover

```
-- Perbandingan volume transaksi terhadap hari normal
SELECT HOUR(WAKTU) AS jam,
       COUNT(*) AS hari_ini,
       (SELECT COUNT(*) FROM DTALIB.TRXHIST t2
         WHERE t2.TGL_TRX = CURRENT DATE - 7 DAYS
           AND HOUR(t2.WAKTU) = HOUR(t1.WAKTU)) AS minggu_lalu
  FROM DTALIB.TRXHIST t1
 WHERE TGL_TRX = CURRENT DATE
 GROUP BY HOUR(WAKTU)
 ORDER BY jam;
```

Volume transaksi yang jauh di bawah hari normal setelah cutover bukan kabar baik. Itu berarti sebagian nasabah tidak bisa bertransaksi dan belum ada yang melapor.

77.5 Analisis pasca-kegiatan

Jadwalkan dalam tujuh hari setelah cutover, saat ingatan masih segar.

Pertanyaan	Kegunaan
Langkah mana yang lebih lama dari perkiraan?	Perbaiki estimasi berikutnya
Langkah mana yang harus diimprovisasi?	Perbaiki dokumentasi
Masalah apa yang tidak terantisipasi?	Tambah ke daftar risiko
Keputusan mana yang paling sulit?	Perjelas kriteria
Apa yang berjalan lebih baik dari harapan?	Pertahankan

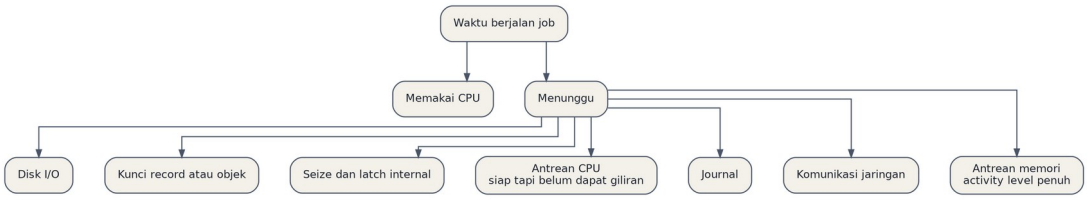
Tulis hasilnya sebagai dokumen, bukan sebagai rapat yang berlalu. Proyek berikutnya akan dipimpin orang yang mungkin berbeda, dan satu-satunya cara pengetahuan ini berpindah adalah kalau ia tertulis.

⚡ Bagian XX — Performance Engineering Mendalam

Bab 78. Wait Accounting

Ini kerangka berpikir paling berguna untuk kinerja di IBM i, dan yang paling jarang dipakai. Alih-alih bertanya "berapa CPU terpakai", ia bertanya "job ini menghabiskan waktunya menunggu apa".

78.1 Ke mana waktu sebuah job pergi



Gambar 85

Sebuah job yang berjalan tiga jam mungkin hanya memakai CPU delapan menit. Sisanya menunggu. Mengetahui menunggu **apa** langsung menunjuk ke perbaikan yang benar.

Jenis tunggu dominan	Arti	Perbaikan yang tepat
Disk I/O	Membaca terlalu banyak data	Index, kurangi cakupan kueri
Kunci record	Berebut dengan job lain	Ubah urutan proses, perkecil transaksi
Seize atau latch	Kontensi objek internal	Pisahkan objek, kurangi paralelisme
Antrean CPU	Kapasitas CPU kurang	Tambah CPU atau turunkan beban serentak
Journal	Journaling jadi penyempitan	Pindahkan receiver ke ASP lain
Antrean memori	Activity level terlalu rendah	Naikkan `MAXACT` pool
Jaringan	Menunggu sistem lain	Bukan masalah IBM i

78.2 Mengaktifkan dan membaca

```
/* Pastikan Collection Services mengumpulkan data tunggu */
STRPFRCOL COLSRVPRF(*CUSTOM) INTERVAL(5) +
          LIB(QMPGDATA) RETPERIOD(30 *DAYS)

/* Ubah kategori yang dikumpulkan lewat */
CFGPFRCOL
```

```
-- Rincian tunggu untuk job tertentu, dari data koleksi
SELECT JOB_NAME,
       ELAPSED_TIME,
       ELAPSED_CPU_TIME,
       ELAPSED_ASYNC_DISK_IO_COUNT,
       ELAPSED_SYNC_DISK_IO_COUNT,
       ELAPSED_PAGE_FAULT_COUNT,
       ELAPSED_INTERACTION_COUNT
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         DETAILED_INFO => 'ALL',
         JOB_NAME_FILTER => 'EODPOST'));
```

Untuk analisis tunggu yang sesungguhnya rinci, Performance Data Investigator di Navigator for i menyajikan grafik per kategori tunggu. Itu antarmuka yang layak Anda pelajari — grafiknya langsung menunjukkan kategori mana yang mendominasi tanpa perlu menafsirkan angka mentah.

78.3 Job Watcher

Job Watcher mengambil cuplikan sangat sering, sampai tiap detik, sehingga bisa menangkap masalah yang luput dari interval lima menit.

```
/* Mulai pengamatan pada job tertentu */
STRJWMON MONITOR(EODWATCH) +
         JOB((123456/EODUSER/EODPOST)) +
         INTERVAL(5) +
         LIB(PFRDATA)

/* Hentikan */
ENDJWMON MONITOR(EODWATCH)

/* Pengamatan seluruh subsystem selama jendela EOD */
STRJWMON MONITOR(EODALL) +
         SBS(EODSBS) +
         INTERVAL(10) +
         MAXRCDS(*NOMAX)
```

Kapan memakainya. Ketika masalah berlangsung singkat dan hilang sebelum Anda sempat mengetik `WRKACTJOB`, atau ketika Anda perlu tahu apa yang dilakukan job pada detik tertentu. Untuk fase EOD yang tiba-tiba melambat, ini alat yang tepat.

Biayanya. Job Watcher menambah beban dan menghasilkan data dalam jumlah besar. Jalankan pada jendela terbatas, bukan terus-menerus.

78.4 Disk Watcher

```
STRDW SSNID(DISKWATCH) +
      DFN(*DEFAULT) +
      COLPRF(*STATISTICS) +
      LIB(PFRDATA) +
      INTERVAL(5)

ENDDW SSNID(DISKWATCH)
```

Dipakai ketika kecurigaan mengarah ke penyimpanan: respons disk tinggi, distribusi I/O tidak merata, atau satu unit disk jauh lebih sibuk dari yang lain.

```
-- Ketidakmerataan beban antar unit disk
SELECT UNIT_NUMBER, PERCENT_USED, ELAPSED_PERCENT_BUSY,
       ELAPSED_IO_REQUESTS, ELAPSED_READ_REQUESTS,
       ELAPSED_WRITE_REQUESTS
  FROM QSYS2.SYSDISKSTAT
 ORDER BY ELAPSED_PERCENT_BUSY DESC;
```

Selisih besar antar unit menandakan distribusi data tidak merata, yang sering terjadi setelah penambahan disk tanpa penyeimbangan ulang.

```
/* Menyeimbangkan data antar unit disk, butuh perencanaan jendela */
STRASPBAL TYPE(*CAPACITY) ASP(1) TIMLMT(180)
ENDASPBAL ASP(1)

/* Melihat status penyeimbangan */
DSPASPBAL
```

78.5 Menyetel memory pool

```
-- Dasar keputusan: faulting per pool
SELECT POOL_NAME, POOL_ID,
       CURRENT_SIZE / 1024 AS mb,
       DEFINED_SIZE / 1024 AS mb_didefinisikan,
       MAXIMUM_ACTIVE_THREADS AS actlvl,
       DATABASE_FAULTS, NON_DATABASE_FAULTS,
       DATABASE_PAGES, NON_DATABASE_PAGES,
       ACTIVE_TO_WAIT, WAIT_TO_INELIGIBLE, ACTIVE_TO_INELIGIBLE
  FROM TABLE(QSYS2.MEMORY_POOL(RESET_STATISTICS => 'NO'));
```

Gejala	Arti	Tindakan
Faulting tinggi, `ACTIVE_TO_INELIGIBLE` nol	Memori kurang	Tambah ukuran pool
Faulting rendah, `ACTIVE_TO_INELIGIBLE` tinggi	Activity level terlalu rendah	Naikkan `MAXACT`
Keduanya tinggi	Beban melebihi kapasitas pool	Tambah memori dan tinjau beban
Keduanya rendah, job tetap lambat	Bukan masalah memori	Cari di tempat lain

`ACTIVE_TO_INELIGIBLE` yang tidak nol berarti ada thread yang siap bekerja tetapi ditolak masuk pool karena activity level penuh. Ini penyempitan yang murah diperbaiki dan sering terlewat.

```
/* Menyetel manual */
CHGSHRPOOL POOL(*SHRPOOL2) SIZE(24000000) ACTLVL(16)

/* Atau serahkan ke penyetel otomatis */
CHGSYSVAL SYSVAL(QPFRADJ) VALUE('2')
```

Nilai `QPFRADJ`	Perilaku
`0`	Tanpa penyesuaian otomatis
`1`	Hanya saat IPL
`2`	Saat IPL dan berkelanjutan
`3`	Hanya berkelanjutan, tidak saat IPL

Untuk lingkungan dengan pola beban siang-malam yang sangat berbeda seperti core banking, nilai `2` bisa membantu, tetapi penyetelan manual terjadwal sering lebih tepat karena Anda tahu kapan EOD dimulai dan penyetel otomatis tidak.

78.6 Paralelisme SMP

```
/* Tingkat paralelisme untuk kueri */
CHGQRYA DEGREE(*OPTIMIZE)
CHGQRYA DEGREE(*MAX)
CHGQRYA DEGREE(*NONE)

/* Nilai bawaan sistem */
CHGSYSVAL SYSVAL(QQRYDEGREE) VALUE('*OPTIMIZE')
```

Nilai	Kapan dipakai
`*NONE`	Jam layanan; kueri tidak boleh menyerap semua CPU
`*OPTIMIZE`	Bawaan yang wajar
`*MAX`	Jendela batch, ketika throughput total yang penting
`*IO`	Paralelisme I/O saja, tanpa paralelisme CPU

Pola yang lazim: setel `*NONE` atau `*OPTIMIZE` pada jam layanan, dan naikkan ke `*MAX` di dalam job EOD. Kueri paralel yang berjalan pada pukul sepuluh pagi bisa membuat seluruh sesi interaktif melambat.

78.7 Job description dan prioritas

```
CRTJOBD JOBD(APPPGM/EODJOBD) +
        JOBQ(QGPL/EODJOBQ) +
        JOBPTY(3) OUTPTY(5) +
        RUNPTY(30) +
        LOG(4 00 *SECLVL) +
        LOGCLPGM(*YES) +
        INLLIBL(APPPGM DTALIB APPPRM QGPL QTEMP) +
        TEXT('Job description untuk EOD')
```

Parameter	Arti	Nilai khas
`JOBPTY`	Prioritas di job queue, 1 tertinggi	3 untuk EOD
`RUNPTY`	Prioritas saat berjalan, 1 tertinggi	20 interaktif, 30 EOD, 50 batch biasa
`LOG`	Tingkat pencatatan	`(4 00 *SECLVL)` saat menyelidiki
`INLLIBL`	Library list awal	Eksplisit, jangan mengandalkan bawaan

Tentang `RUNPTY`. Angka lebih kecil berarti prioritas lebih tinggi. Job dengan `RUNPTY(20)` akan mendahului `RUNPTY(50)`. Memberi job batch prioritas lebih tinggi daripada interaktif adalah keputusan yang sah pada malam hari dan bencana pada siang hari.

78.8 Prestart job

Untuk beban dengan banyak koneksi pendek — ODBC, layanan web — prestart job menghilangkan biaya pembuatan job tiap koneksi.

```
/* Melihat konfigurasi prestart untuk server database */
DSPSBSD SBSD(QSYS/QUSRWRK)

/* Menyetel */
CHGPJE SBSD(QSYS/QUSRWRK) PGM(QSYS/QZDASOINIT) +
       INLJOBS(20) +
       THRESHOLD(10) +
       ADLJOBS(10) +
       MAXUSE(200) +
       WAIT(*YES)
```

Parameter	Arti
INLJOBS	Jumlah job disiapkan saat subsystem dimulai
THRESHOLD	Batas bawah sebelum job baru dibuat
ADLJOBS	Berapa job ditambahkan sekaligus
MAXUSE	Berapa kali satu job dipakai sebelum diakhiri

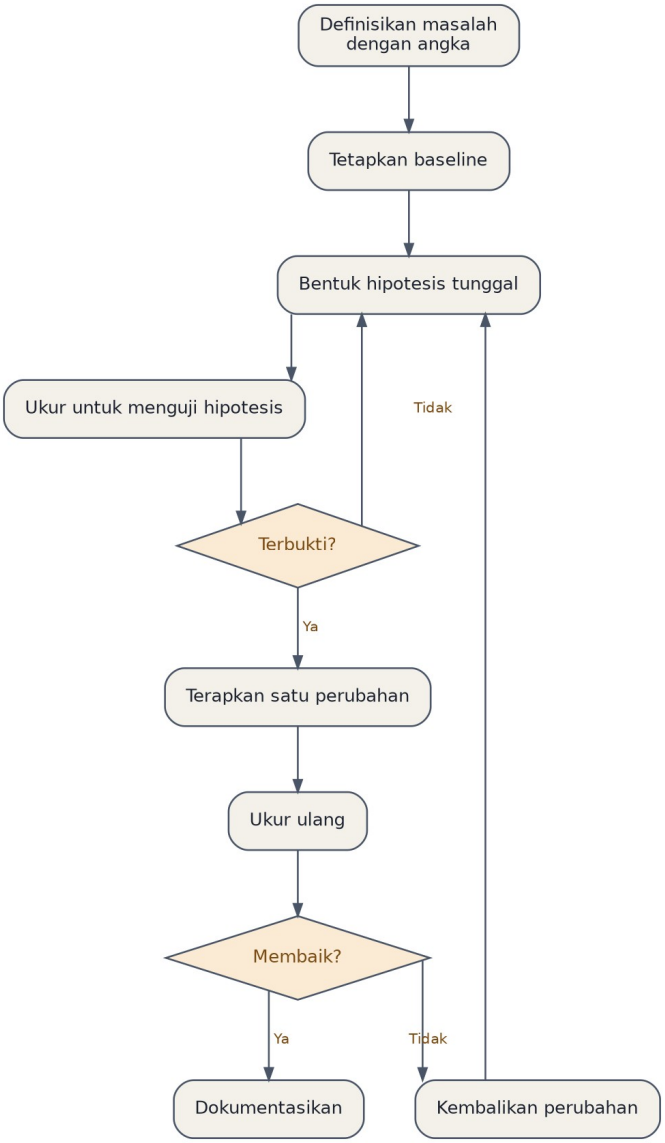
MAXUSE yang terlalu tinggi menyebabkan job menumpuk memori dari koneksi sebelumnya. Yang terlalu rendah meniadakan manfaat prestart. Nilai antara 200 dan 500 umumnya wajar; ukur di lingkungan Anda.

```
-- Apakah jumlah prestart job memadai
SELECT COUNT(*) AS jml_prestart,
       SUM(CASE WHEN JOB_STATUS = 'PSRW' THEN 1 ELSE 0 END) AS menganggur,
       SUM(CASE WHEN JOB_STATUS <> 'PSRW' THEN 1 ELSE 0 END) AS terpakai
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO())
 WHERE JOB_NAME LIKE '%QZDASOINIT%';
```

Bila jumlah menganggur sering nol, naikkan INLJOBS dan THRESHOLD. Koneksi yang harus menunggu job baru dibuat terasa lambat bagi pengguna.

⚡ Bab 79. Menyusun Studi Kinerja

79.1 Kerangka kerja



Gambar 86

Satu perubahan pada satu waktu. Menerapkan lima perubahan sekaligus lalu melihat perbaikan tidak memberi tahu Anda perubahan mana yang bekerja — dan mana yang justru merugikan tetapi tertutupi oleh yang lain.

79.2 Mendefinisikan masalah dengan angka

Pernyataan yang tidak berguna: "EOD lambat".

Pernyataan yang berguna: "Fase akrual berjalan 94 menit, naik dari baseline 38 menit selama enam minggu terakhir, sementara jumlah rekening naik 4%. Jendela batch tersisa 40 menit."

Yang kedua bisa diselidiki. Yang pertama hanya bisa dikeluhkan.

79.3 Template laporan studi kinerja

```
STUDI KINERJA: [judul]
Tanggal: [   ]   Penyusun: [   ]

1. MASALAH
   Gejala terukur:
   Dampak bisnis:
   Sejak kapan:

2. BASELINE
   Metrik            Nilai saat ini    Nilai normal
   ------------------------------------------------

3. PENGUKURAN
   Alat yang dipakai:
   Jendela pengukuran:
   Temuan utama:

4. ANALISIS
   Hipotesis yang diuji:
   Bukti pendukung:
   Hipotesis yang ditolak dan alasannya:

5. REKOMENDASI
   Perubahan          Perkiraan dampak   Risiko   Usaha
   -------------------------------------------------------

6. HASIL SETELAH PENERAPAN
   Metrik            Sebelum   Sesudah   Selisih
   ---------------------------------------------

7. TINDAK LANJUT
```

Bagian empat, khususnya baris hipotesis yang ditolak, adalah yang paling berharga bagi orang berikutnya. Ia mencegah penyelidikan yang sama diulang dari nol enam bulan kemudian.

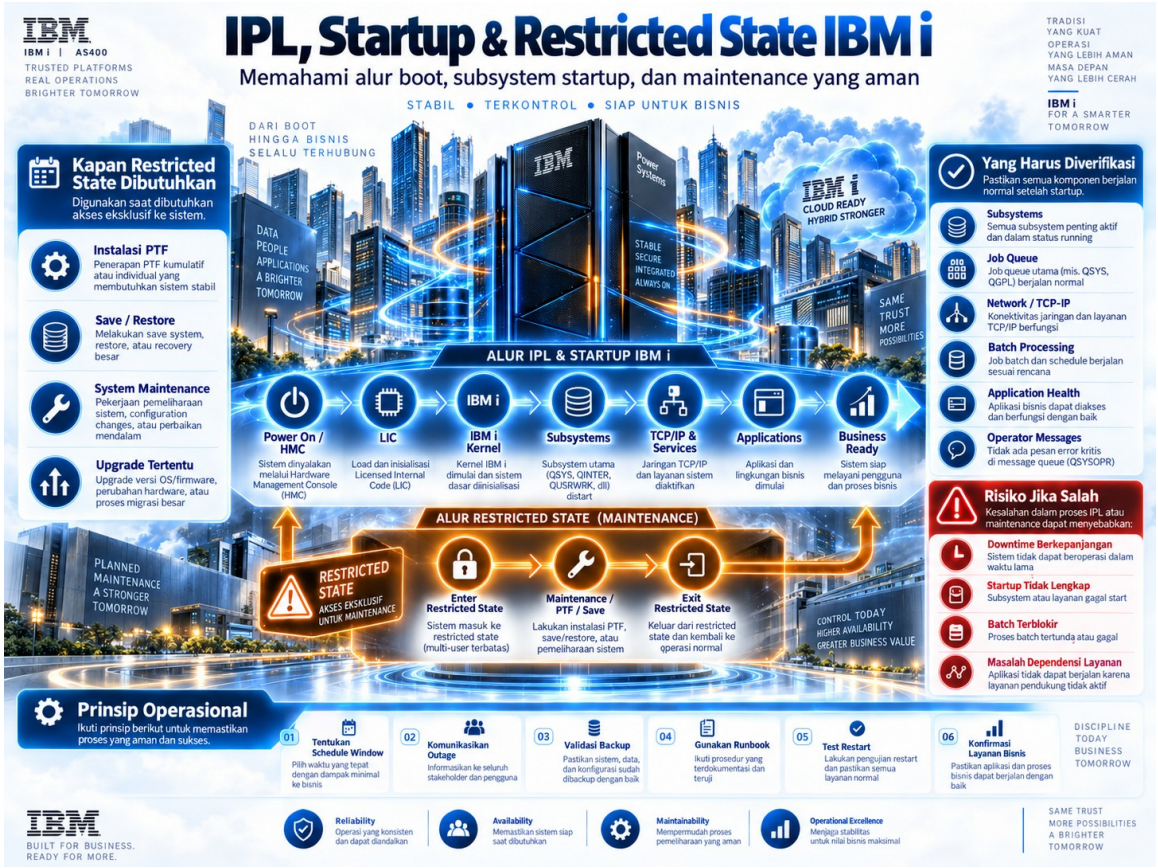
79.4 Kesalahan analisis yang sering terjadi

Kesalahan	Akibat
Mengukur tanpa mereset statistik	Angka mencerminkan berhari-hari, bukan saat ini
Mengukur di luar jendela masalah	Tidak menangkap kondisi yang diselidiki
Menyimpulkan dari satu pengamatan	Anomali dikira pola
Mengubah dan mengukur bersamaan	Tidak tahu perubahan mana yang bekerja
Mengabaikan perubahan volume data	Sistem yang sama dengan beban dua kali lipat
Menyalahkan yang paling terlihat	Job dengan CPU tertinggi belum tentu penyebabnya

Baris terakhir layak diperhatikan. Job yang memakai CPU paling banyak sering adalah korban, bukan penyebab: ia berjalan lambat karena menunggu, dan menunggu sambil menghabiskan siklus dalam perulangan menunggu.

⚙️ Bagian XXI — Operasi Tingkat Sistem

⚙️ Bab 80. IPL



Gambar 87 — IPL, startup, dan restricted state

IPL adalah operasi paling berisiko yang dilakukan secara rutin. Setiap IPL adalah kesempatan bagi masalah tersembunyi untuk muncul.

80.1 Jenis IPL

Jenis	Cara	Kapan
Normal terkendali	`PWRDWNSYS RESTART(*YES)`	Pemeliharaan terjadwal
Tanpa restart	`PWRDWNSYS RESTART(*NO)`	Pemadaman terencana
Setelah kegagalan	Otomatis atau manual	Setelah kehilangan daya
Mode A	Dari kode berlisensi permanen	Operasi normal
Mode B	Dari kode berlisensi sementara	Setelah menerapkan PTF LIC
Mode C	Servis	Hanya atas arahan IBM
Mode D	Dari media alternatif	Instalasi dan pemulihan

80.2 Prosedur IPL terkendali

```
PGM
   /* 1. Umumkan */
   SNDBRKMSG MSG('Sistem akan dimatikan dalam 30 menit') TOMSGQ(*ALLWS)
   DLYJOB DLY(1200)
```

```
   SNDBRKMSG MSG('Sistem dimatikan dalam 10 menit') TOMSGQ(*ALLWS)
   DLYJOB DLY(600)

   /* 2. Hentikan subsystem aplikasi secara berurutan */
   ENDSBS SBS(EODSBS)  OPTION(*CNTRLD) DELAY(300)
   ENDSBS SBS(QINTER)  OPTION(*CNTRLD) DELAY(300)
   ENDSBS SBS(QBATCH)  OPTION(*CNTRLD) DELAY(600)

   /* 3. Hentikan layanan jaringan */
   ENDTCPSVR SERVER(*ALL)
   DLYJOB DLY(60)
   ENDTCP OPTION(*CNTRLD) ENDTCPSVR(*YES) DELAY(300)

   /* 4. Backup terakhir bila diperlukan */

   /* 5. IPL */
   PWRDWNSYS OPTION(*CNTRLD) DELAY(600) RESTART(*YES) +
             IPLSRC(B)
ENDPGM
```

Tentang `DELAY`. Nilai ini adalah berapa lama sistem menunggu job berhenti dengan rapi sebelum memaksa. Terlalu pendek berarti job terputus di tengah transaksi. Untuk lingkungan dengan batch panjang, 600 detik adalah titik awal yang wajar, tetapi sesuaikan dengan durasi transaksi terpanjang Anda.

80.3 Yang diperiksa sebelum IPL

- ☐ Tidak ada job batch penting yang sedang berjalan
- ☐ Tidak ada transaksi commitment control yang terbuka
- ☐ Backup terbaru sudah terverifikasi
- ☐ PTF yang menunggu IPL sudah diketahui daftarnya
- ☐ Mode IPL sudah benar, A atau B
- ☐ Konsol dapat diakses
- ☐ Personel siap bila IPL tidak selesai

```
-- Transaksi terbuka yang akan terganggu
SELECT JOB_NAME, COMMITMENT_DEFINITION, PENDING_CHANGES, STATE
  FROM QSYS2.DB_TRANSACTION_INFO
 WHERE PENDING_CHANGES > 0;

-- PTF yang akan aktif setelah IPL
SELECT PTF_IDENTIFIER, PTF_PRODUCT_ID, PTF_IPL_ACTION,
       PTF_ACTION_PENDING, PTF_ACTION_REQUIRED
  FROM QSYS2.PTF_INFO
 WHERE PTF_IPL_ACTION <> 'NONE';
```

80.4 Setelah IPL

```
-- Berapa lama IPL berlangsung dan jenisnya
SELECT * FROM QSYS2.SYSTEM_STATUS_INFO;

-- Riwayat IPL terakhir dari history log
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, MESSAGE_TEXT
  FROM TABLE(QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 24 HOURS))
 WHERE MESSAGE_ID IN ('CPF0999','CPI0954','CPI091D','CPC1129')
 ORDER BY MESSAGE_TIMESTAMP;
```

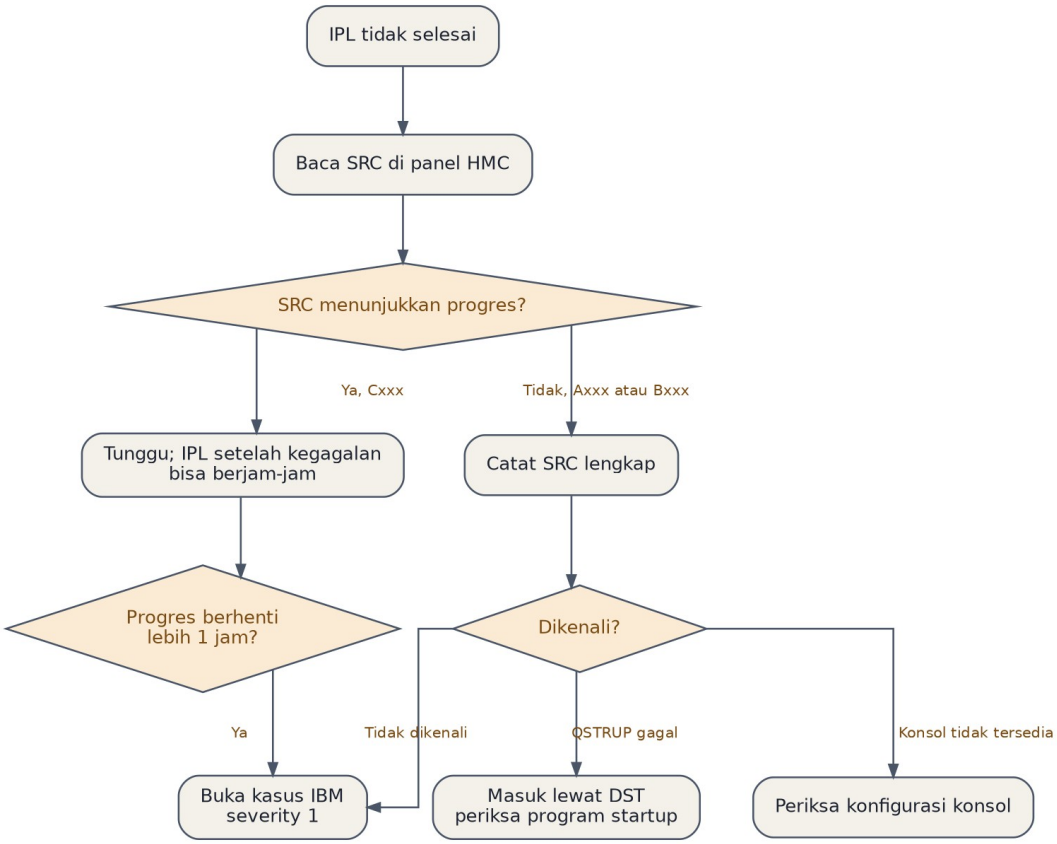
Daftar periksa pasca-IPL:

- ☐ Seluruh subsystem yang seharusnya aktif sudah aktif
- ☐ Seluruh interface TCP/IP naik
- ☐ Seluruh server TCP yang dibutuhkan berjalan
- ☐ Tidak ada pesan gagal di QSYSOPR
- ☐ Journaling aktif untuk seluruh file yang seharusnya
- ☐ Job scheduler entry tidak ada yang terlewat jadwalnya
- ☐ Replikasi ke DR kembali berjalan
- ☐ Satu transaksi uji berhasil dari tiap kanal

```
/* Program startup yang dijalankan otomatis; periksa isinya */
DSPSYSVAL SYSVAL(QSTRUPPGM)
RTVCLSRC PGM(QSYS/QSTRUP) SRCFILE(APPSRC/QCLSRC)
```

Program startup adalah tempat di mana urutan penyalaan layanan ditentukan. Salinannya harus ada di kendali versi, dan setiap perubahan padanya menuntut change request — sebab kesalahan di sini hanya terlihat pada IPL berikutnya, yang mungkin berbulan-bulan kemudian.

80.5 IPL yang tidak selesai



Gambar 88

Jangan melakukan IPL paksa berulang. Bila sistem sedang melakukan klaim ruang penyimpanan setelah penghentian tidak normal, memaksa IPL lagi hanya mengulang proses dari awal dan memperpanjangnya.

Bab 81. Restricted State

Beberapa operasi menuntut sistem dalam keadaan terbatas: hanya satu job berjalan di konsol.

81.1 Operasi yang membutuhkannya

Operasi	Alasan
`SAVSYS`	Menyimpan sistem operasi
`SAVLIB *NONSYS`	Menyimpan library sistem
`RCLSTG`	Klaim ruang penyimpanan
Sebagian upgrade	Objek sistem harus bebas
`RSTLIB` ke library sistem	Objek tidak boleh terpakai

```
/* Masuk restricted state */
ENDSBS SBS(*ALL) OPTION(*CNTRLD) DELAY(600)

/* Tunggu sampai benar-benar selesai; periksa dengan */
WRKACTJOB

/* Keluar */
STRSBS SBSD(QSYS/QCTL)
```

Peringatan yang tidak boleh dilupakan. Perintah `ENDSBS SBS(*ALL)` harus dijalankan dari konsol, bukan dari sesi Telnet. Menjalankannya dari sesi jarak jauh akan memutus sesi Anda sendiri di tengah jalan, dan Anda kehilangan kendali atas sistem yang sedang dalam keadaan setengah berhenti.

```
/* Pola aman: kirim sebagai job batch dengan penundaan */
SBMJOB CMD(CALL PGM(APPPGM/GORESTRICT)) +
       JOB(RESTRICT) JOBQ(QSYS/QCTL)
```

81.2 Reclaim Storage

```
/* Dalam restricted state */
RCLSTG SELECT(*ALL)

/* Versi yang lebih ringan, bisa berjalan tanpa restricted state */
RCLSTG SELECT(*DBXREF)
```

`RCLSTG` mencari objek yang kehilangan penunjuknya dan ruang yang tidak terhitung. Ia bisa memakan berjam-jam pada sistem besar, dan durasinya sulit diperkirakan. Jadwalkan pada jendela akhir pekan, bukan malam hari kerja.

Kapan ia dibutuhkan. Setelah penghentian tidak normal, ketika pemakaian ASP tampak lebih besar daripada jumlah objek yang terlihat, atau ketika `DSPOBJD` melaporkan objek yang tidak bisa diakses.

81.3 Membangun jendela restricted state

```
PGM
   DCL VAR(&MULAI) TYPE(*CHAR) LEN(6)

   RTVSYSVAL SYSVAL(QTIME) RTNVAR(&MULAI)
   SNDMSG MSG('Restricted state dimulai ' *CAT &MULAI) TOUSR(SYSENG)

   /* Hentikan seluruh subsystem */
   ENDSBS SBS(*ALL) OPTION(*CNTRLD) DELAY(900)
   DLYJOB DLY(60)
```

```
   /* Pastikan benar-benar terbatas sebelum melanjutkan */
   /* Periksa jumlah job aktif */

   /* Operasi yang membutuhkan restricted state */
   SAVSYS DEV(TAPMLB01) ENDOPT(*LEAVE)
   MONMSG MSGID(CPF0000) EXEC(DO)
      SNDMSG MSG('SAVSYS GAGAL') TOUSR(SYSENG) MSGTYPE(*INQ)
   ENDDO

   SAVLIB LIB(*NONSYS) DEV(TAPMLB01) ENDOPT(*UNLOAD)
   MONMSG MSGID(CPF0000)

   /* Kembalikan sistem */
   STRSBS SBSD(QSYS/QCTL)
   DLYJOB DLY(120)
   STRSBS SBSD(QGPL/EODSBS)
   STRTCP

   SNDMSG MSG('Sistem kembali normal') TOUSR(SYSENG)
ENDPGM
```

Bab 82. HMC dan PowerVM

82.1 Peran HMC

Hardware Management Console mengelola partisi, bukan sistem operasi di dalamnya. Pemisahan ini penting untuk pemisahan tugas: seseorang yang bisa mengelola HMC bisa mematikan partisi produksi tanpa memiliki akses apa pun ke IBM i di dalamnya.

Tugas	Di HMC	Di IBM i
Membuat atau menghapus partisi	Ya	Tidak
Mengubah alokasi CPU dan memori	Ya	Tidak
Menyalakan dan mematikan partisi	Ya	Sebagian
Mengelola profil partisi	Ya	Tidak
Memasang PTF firmware	Ya	Tidak
Melihat SRC saat IPL gagal	Ya	Tidak
Mengelola objek dan pengguna	Tidak	Ya

82.2 Dynamic LPAR

Sumber daya bisa dipindahkan antar partisi tanpa mematikan apa pun.

Sumber daya	Bisa dipindah hidup?	Catatan
Processing units	Ya	Dalam batas profil
Virtual processors	Ya	Dalam batas profil
Memori	Ya	Lebih lambat, bertahap
Adapter fisik	Ya, dengan syarat	Perlu perencanaan
Virtual adapter	Ya	Lewat VIOS

Pola yang lazim di lingkungan perbankan: pindahkan sumber daya dari partisi UAT ke partisi produksi pada malam EOD, kembalikan pagi hari. Ini memberi kapasitas tambahan tanpa membeli perangkat keras.

Batas yang harus dipahami. Nilai maksimum di profil partisi tidak bisa dilampaui tanpa mematikan partisi. Setel nilai maksimum lebih tinggi dari kebutuhan sejak awal, karena menaikkannya kemudian menuntut penghentian.

82.3 VIOS

```
# Dari baris perintah VIOS
lsmap -all                    # pemetaan virtual ke fisik
lsdev -type disk              # perangkat disk
lsdev -type adapter           # adapter
errlog                        # log kesalahan
fcstat fcs0                   # statistik adapter fibre channel
entstat -all ent0             # statistik adapter ethernet
```

Dua VIOS dipasang berpasangan agar tidak ada titik kegagalan tunggal. Yang perlu diverifikasi berkala: apakah kedua jalur benar-benar aktif, atau sudah berbulan-bulan berjalan pada satu jalur karena kegagalan yang tidak terdeteksi.

```
# Memeriksa redundansi jalur dari sisi VIOS
lspath                        # status jalur multipath
```

82.4 Firmware

Pembaruan firmware sering diabaikan sampai IBM mensyaratkannya untuk dukungan. Perlakukan sebagai bagian dari siklus pemeliharaan tahunan:

1. Periksa level firmware saat ini terhadap yang disarankan
2. Baca catatan rilis untuk masalah yang diketahui
3. Terapkan di lingkungan non-produksi lebih dulu
4. Jadwalkan jendela; sebagian pembaruan menuntut penghentian seluruh mesin
5. Verifikasi seluruh partisi naik kembali dengan benar

Pembaruan firmware yang menuntut penghentian seluruh mesin berarti seluruh partisi mati bersamaan — termasuk partisi DR bila berada di mesin fisik yang sama. Inilah alasan DR sebaiknya di mesin terpisah, idealnya di lokasi terpisah.

⚙️ Bab 83. Pembersihan Sistem Terjadwal

83.1 Yang menumpuk tanpa disadari

Objek	Tumbuh karena	Cara membersihkan
Spooled file	Laporan tidak pernah dihapus	`DLTSPLF`, kebijakan retensi
Job log	Setiap job menghasilkan satu	`QSYS2.JOBLOG_INFO`, pembersihan
Journal receiver	Setiap perubahan data	Simpan lalu lepas
Save file	Backup dan promosi	Hapus setelah dipindah ke media
Objek di QTEMP job mati	Penghentian tidak normal	`RCLSTG`
Berkas di IFS	Interface dan log aplikasi	Skrip pembersihan
Tabel riwayat aplikasi	Snapshot harian	Arsip dan hapus
Objek di QRPLOBJ	Penggantian objek	Dibersihkan saat IPL

83.2 Pembersihan otomatis bawaan

```
/* Konfigurasi pembersihan otomatis sistem */
CHGCLNUP ALWCLNUP(*YES) +
         STRTIME(010000) +
         USRMSG(30) +
         SYSMSG(30) +
         JOBLOG(30) +
         OUTQ(30) +
         PRBLOG(30)

/* Mulai */
STRCLNUP
```

Ini menangani sebagian, tetapi bukan semuanya. Journal receiver, save file, dan berkas IFS tetap menjadi tanggung jawab Anda.

83.3 Skrip pembersihan sendiri

```
PGM
   /* Spooled file melewati retensi */
   RUNSQL SQL('CREATE TABLE QTEMP/SPLHAPUS AS ( +
      SELECT SPOOLED_FILE_NAME, JOB_NAME, FILE_NUMBER +
        FROM QSYS2.OUTPUT_QUEUE_ENTRIES +
       WHERE CREATE_TIMESTAMP < CURRENT DATE - 90 DAYS +
         AND OUTPUT_QUEUE_NAME NOT IN (''ARSIP'',''REGULATOR'') +
      ) WITH DATA') COMMIT(*NONE)

   /* Save file lama di library kerja */
   RUNSQL SQL('CREATE TABLE QTEMP/SAVFHAPUS AS ( +
      SELECT OBJNAME, OBJLONGSCHEMA +
        FROM TABLE(QSYS2.OBJECT_STATISTICS(''ROLLBACK'',''*FILE'')) +
       WHERE OBJATTRIBUTE = ''SAVF'' +
         AND OBJCREATED < CURRENT TIMESTAMP - 60 DAYS +
      ) WITH DATA') COMMIT(*NONE)

   SNDMSG MSG('Daftar kandidat pembersihan siap ditinjau') TOUSR(SYSENG)
ENDPGM
```

Pola yang saya sarankan: daftar dulu, hapus kemudian. Program menghasilkan daftar kandidat, seorang manusia meninjaunya, lalu penghapusan dijalankan. Penghapusan otomatis tanpa tinjauan pernah menghapus laporan regulator yang seharusnya disimpan tujuh tahun.

83.4 Pembersihan IFS

```
#!/QOpenSys/pkgs/bin/bash
# Berkas interface yang sudah diproses, lebih tua dari 30 hari
find /interface/arsip -type f -mtime +30 -name '*.txt' -print > /tmp/kandidat.txt
wc -l /tmp/kandidat.txt

# Tinjau dulu, baru hapus
# find /interface/arsip -type f -mtime +30 -name '*.txt' -delete

# Log aplikasi
find /var/log/app -type f -mtime +14 -name '*.log' -exec gzip {} \;
find /var/log/app -type f -mtime +90 -name '*.log.gz' -print
```

```
-- Direktori IFS terbesar
SELECT PATH_NAME, OBJECT_TYPE,
       DATA_SIZE / 1048576 AS mb,
       LAST_USED_TIMESTAMP
  FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
         START_PATH_NAME => '/',
         SUBTREE_DIRECTORIES => 'YES',
         OBJECT_TYPE_LIST => '*ALLDIR'))
 ORDER BY DATA_SIZE DESC
 FETCH FIRST 30 ROWS ONLY;
```

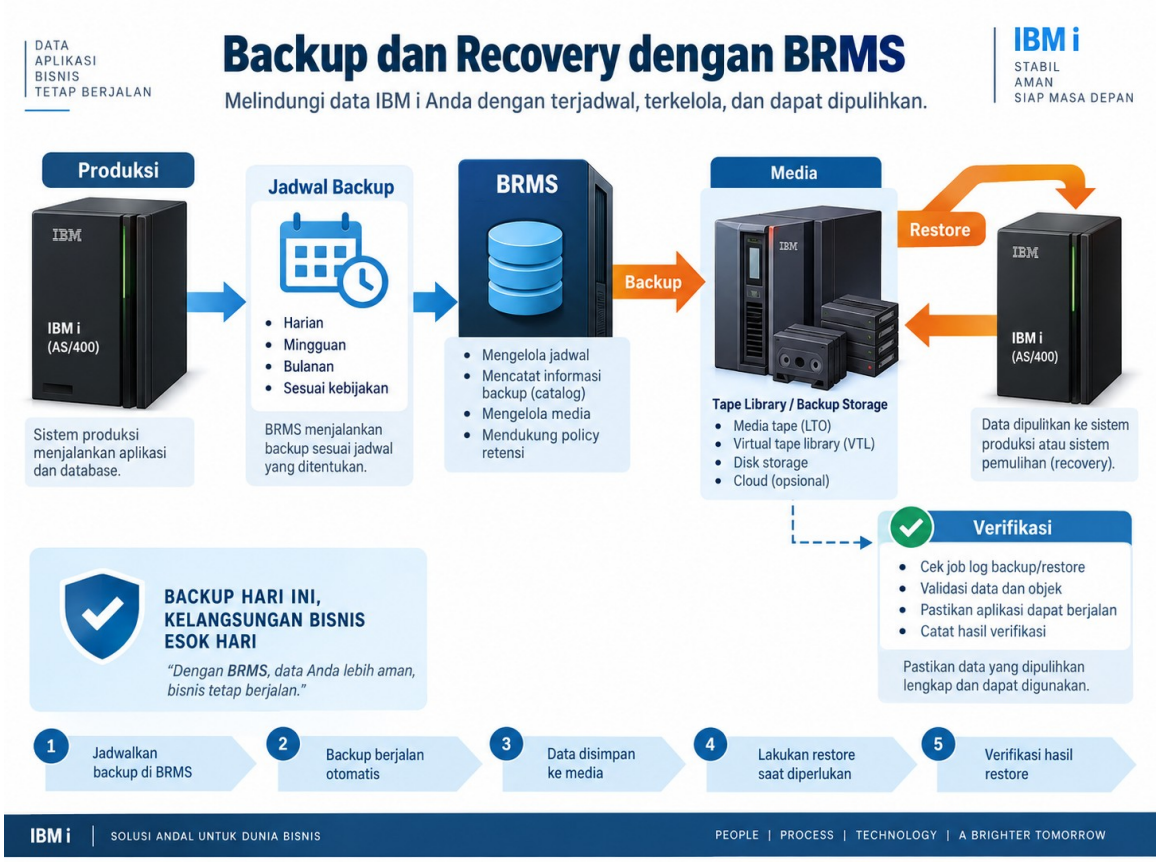
83.5 Kalender pemeliharaan tahunan

Periode	Kegiatan
Harian	Pemeriksaan pagi, backup, pemantauan
Mingguan	Tinjau tren, bersihkan spooled file, audit wewenang
Bulanan	Baseline, uji restore satu objek, tinjau PTF group
Triwulanan	Terapkan group PTF, tinjau runbook, latihan DR tingkat 1
Semesteran	Uji restore library penuh, tinjau kapasitas, verifikasi media offsite
Tahunan	Upgrade TR atau rilis, uji DR penuh, tinjau arsitektur, `RCLSTG`

Kalender ini layak ditempel dan dijadikan komitmen tertulis. Pekerjaan pemeliharaan yang tidak terjadwal adalah pekerjaan yang tidak dilakukan, sampai ia berubah menjadi insiden.

💾 Bagian XXII — BRMS dan Manajemen Media

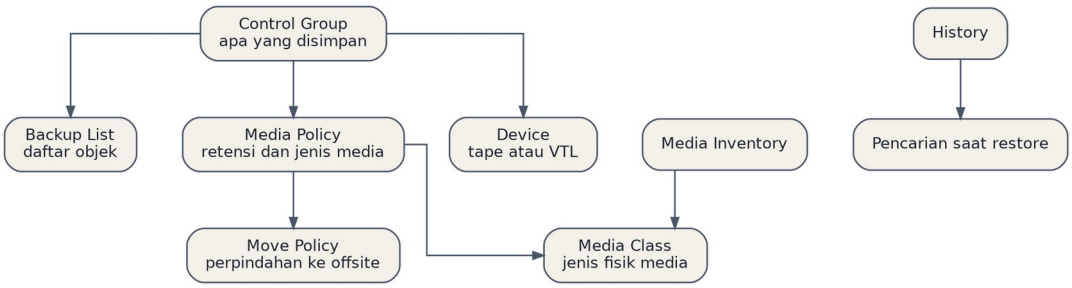
💾 Bab 84. Konsep BRMS



Gambar 89 — Backup dan recovery dengan BRMS

Backup Recovery and Media Services mengubah backup dari kumpulan skrip menjadi sistem yang bisa diaudit. Perbedaan terpenting: BRMS tahu di media mana sebuah objek tersimpan, sehingga pemulihan tidak lagi menuntut seseorang mengingat.

84.1 Susunan konsep



Gambar 90

Objek BRMS	Menjawab pertanyaan
Control group	Apa yang disimpan dan dengan urutan apa

Objek BRMS	Menjawab pertanyaan
Backup list	Objek mana saja yang termasuk
Media policy	Berapa lama disimpan, media jenis apa
Media class	Kapasitas, jenis fisik, kompatibilitas perangkat
Move policy	Kapan keluar ke lokasi cadangan, kapan kembali
Media inventory	Media apa yang ada, di mana, isinya apa
History	Objek ini ada di media mana

84.2 Perintah harian

```
/* Menu utama */
WRKPCYBRM TYPE(*SYS)
WRKCTLGBRM
WRKMEDBRM
WRKMEDIBRM

/* Menjalankan control group */
STRBKUBRM CTLGRP(HARIAN) SBMJOB(*YES)
STRBKUBRM CTLGRP(MINGGUAN) SBMJOB(*YES)

/* Melihat riwayat backup */
WRKMEDIBRM
DSPLOGBRM TYPE(*BKU) PERIOD((*AVAIL *BEGIN) (*AVAIL *END))

/* Pemulihan */
WRKMEDIBRM
STRRCYBRM OPTION(*RESTORE) ACTION(*RESTORE)
```

84.3 Merancang control group

```
/* Melihat dan mengubah control group */
WRKCTLGBRM TYPE(*BKU)
```

Rancangan yang lazim untuk lingkungan core banking:

Control group	Isi	Jadwal	Retensi
`PREEOD`	Library data dan parameter	Harian sebelum EOD	14 hari
`POSTEOD`	Library data, journal receiver	Harian sesudah EOD	35 hari
`HARIAN`	Objek berubah, IFS	Harian	14 hari
`MINGGUAN`	Seluruh library pengguna	Mingguan	5 minggu
`BULANAN`	`SAVSYS` dan seluruh sistem	Bulanan	13 bulan
`TAHUNAN`	Seluruh sistem	Tahunan	7 tahun atau sesuai regulasi

Retensi tahunan tujuh tahun bukan angka teknis melainkan angka regulatoris. Pastikan ia berasal dari kebijakan tertulis institusi Anda, bukan dari kebiasaan.

84.4 Media policy dan retensi

```
WRKPCYBRM TYPE(*MED)
```

Parameter	Arti	Pertimbangan
Retention type	Berdasarkan hari, tanggal, atau versi	Hari paling sederhana

Parameter	Arti	Pertimbangan
Retain media	Berapa lama sebelum bisa dipakai ulang	Harus melebihi kebutuhan pemulihan
Media class	Jenis media yang dipakai	Harus cocok dengan perangkat
Move policy	Jadwal keluar masuk lokasi cadangan	Selaras dengan kurir
Secure media	Perlukah enkripsi	Wajib untuk media yang keluar

Kesalahan retensi yang paling berbahaya. Media yang bisa dipakai ulang terlalu cepat berarti backup lama tertimpa sebelum Anda sempat menyadari data rusak. Kerusakan data yang tidak terdeteksi selama dua minggu, dengan retensi tujuh hari, berarti tidak ada satu pun salinan yang bersih.

Pertimbangkan pola retensi bertingkat: harian pendek, mingguan menengah, bulanan panjang. Ini memberi titik pulih pada berbagai kedalaman waktu tanpa menyimpan segalanya selamanya.

84.5 Memantau BRMS lewat SQL

```
-- Media yang sudah kedaluwarsa dan bisa dipakai ulang
SELECT VOLUME_SERIAL, MEDIA_CLASS, EXPIRATION_DATE,
       LOCATION, VOLUME_STATUS, BYTES_WRITTEN / 1073741824 AS gb
  FROM QUSRBRM.MEDIA_INFO
 WHERE EXPIRATION_DATE <= CURRENT DATE
 ORDER BY EXPIRATION_DATE;

-- Media yang berada di lokasi cadangan
SELECT VOLUME_SERIAL, LOCATION, MOVE_DATE,
       EXPIRATION_DATE, MEDIA_CLASS
  FROM QUSRBRM.MEDIA_INFO
 WHERE LOCATION <> 'HOME'
 ORDER BY MOVE_DATE DESC;

-- Riwayat penyimpanan objek tertentu, untuk mencari saat restore
SELECT SAVE_TIMESTAMP, LIBRARY_NAME, OBJECT_NAME, OBJECT_TYPE,
       VOLUME_SERIAL, SEQUENCE_NUMBER, SAVE_TYPE
  FROM QUSRBRM.SAVE_HISTORY
 WHERE LIBRARY_NAME = 'DTALIB'
   AND OBJECT_NAME = 'REKENING'
 ORDER BY SAVE_TIMESTAMP DESC
 FETCH FIRST 20 ROWS ONLY;
```

Nama tabel BRMS dapat berbeda antar rilis. Verifikasi di sistem Anda sebelum menjadikannya bagian dari skrip pemantauan permanen.

84.6 Memverifikasi bahwa backup benar-benar berhasil

Status "selesai" bukan bukti. Yang menjadi bukti:

```
/* Berapa objek yang benar-benar tersimpan */
DSPLOGBRM TYPE(*BKU) PERIOD((*AVAIL *CURRENT))

/* Objek yang GAGAL disimpan */
/* Cari pesan CPF3730 dan sejenisnya di log */
```

```
-- Objek yang tidak tersimpan pada backup terakhir
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, MESSAGE_TEXT
  FROM TABLE(QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 24 HOURS))
 WHERE MESSAGE_ID IN ('CPF3730','CPF3771','CPF3768','CPF3701')
 ORDER BY MESSAGE_TIMESTAMP;
```

`CPF3730` berarti sebagian objek tidak tersimpan. Ini pesan yang sering diabaikan karena job secara keseluruhan tampak selesai. Periksa setiap kemunculannya.

Daftar periksa verifikasi harian:

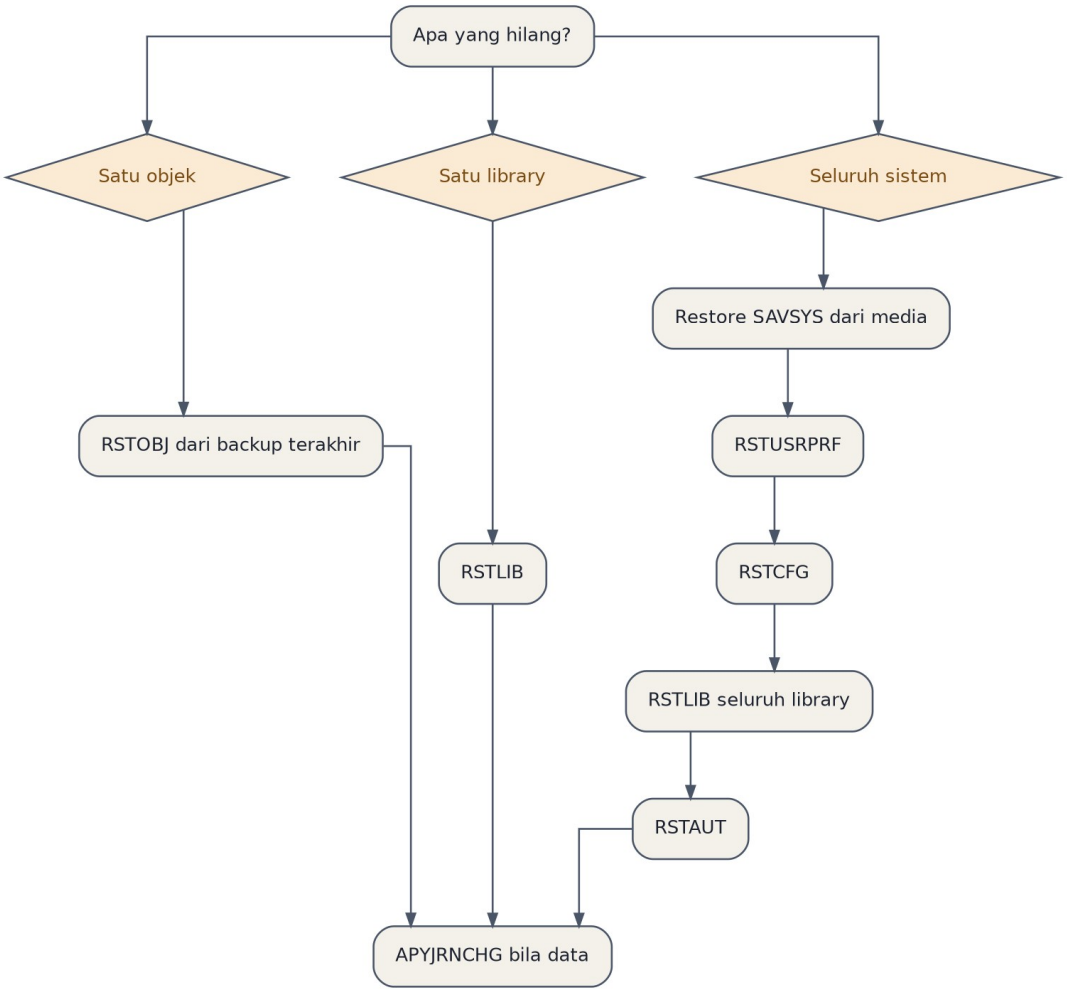
- ☐ Control group selesai tanpa pesan gagal
- ☐ Jumlah objek tersimpan sesuai perkiraan
- ☐ Tidak ada `CPF3730` atau sejenisnya
- ☐ Media terpakai tercatat di inventaris
- ☐ Journal receiver ikut tersimpan
- ☐ Ukuran backup masuk akal dibanding hari sebelumnya

```
-- Ukuran backup harian, untuk mendeteksi anomali
SELECT DATE(SAVE_TIMESTAMP) AS tgl,
       SUM(BYTES_SAVED) / 1073741824 AS gb,
       COUNT(*) AS jml_objek
  FROM QUSRBRM.SAVE_HISTORY
 WHERE SAVE_TIMESTAMP > CURRENT DATE - 30 DAYS
 GROUP BY DATE(SAVE_TIMESTAMP)
 ORDER BY tgl DESC;
```

Ukuran backup yang tiba-tiba separuh dari biasanya berarti ada yang tidak ikut tersimpan. Ini deteksi yang murah dan sangat efektif.

💾 Bab 85. Prosedur Pemulihan

85.1 Hierarki pemulihan



Gambar 91

Urutan pemulihan sistem penuh tidak boleh diubah. `RSTAUT` harus dijalankan terakhir, setelah seluruh objek dan profil ada. Menjalankannya lebih awal berarti wewenang diberikan ke objek yang belum ada, dan hilang.

85.2 Pemulihan objek tunggal

```
/* Cari lebih dulu di media mana objeknya */
WRKMEDIBRM

/* Restore satu objek */
RSTOBJ OBJ(HITBUNGA) SAVLIB(APPPGM) DEV(TAPMLB01) +
       OBJTYPE(*PGM) RSTLIB(APPPGM) +
       MBROPT(*ALL) ALWOBJDIF(*NONE)

/* Restore ke library lain untuk dibandingkan dulu */
RSTOBJ OBJ(REKENING) SAVLIB(DTALIB) DEV(TAPMLB01) +
       OBJTYPE(*FILE) RSTLIB(QTEMP)
```

Me-restore ke library sementara lebih dulu adalah kebiasaan yang layak dibangun. Ia memberi kesempatan memverifikasi isi sebelum menimpa yang ada di produksi.

85.3 Pemulihan dengan journal

```
/* 1. Restore dari save terakhir */
RSTLIB SAVLIB(DTALIB) DEV(TAPMLB01) RSTLIB(DTALIB)

/* 2. Terapkan perubahan sampai titik waktu tertentu */
APYJRNCHG JRN(JRNLIB/JRNDTA) +
          FILE((DTALIB/*ALL)) +
          RCVRNG(*LASTSAVE) +
          FROMENT(*LASTSAVE) +
          TOTIME('18/09/2026' '16:00:00') +
          OPTION(*IGNINQMSG)

/* Atau: batalkan perubahan sampai titik tertentu */
RMVJRNCHG JRN(JRNLIB/JRNDTA) +
          FILE((DTALIB/REKENING)) +
          RCVRNG(*CURCHAIN) +
          FROMENT(*LAST) +
          TOENT(123456)
```

`RMVJRNCHG` membalik perubahan mundur dari kondisi saat ini. Ini berguna ketika kerusakan data diketahui tepat setelah terjadi, dan lebih cepat daripada restore penuh. Tetapi ia hanya bisa membalik sampai titik save terakhir.

85.4 Menentukan titik waktu pemulihan

Pertanyaan yang harus dijawab sebelum memilih `TOTIME`: kapan tepatnya kerusakan terjadi?

```
/* Telusuri journal untuk menemukan kapan perubahan mencurigakan terjadi */
DSPJRN JRN(JRNLIB/JRNDTA) +
       FILE((DTALIB/REKENING)) +
       RCVRNG(*CURCHAIN) +
       FROMTIME('18/09/2026' '00:00:00') +
       ENTTYP(*RCD) +
       OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4) +
       OUTFILE(QTEMP/JRNCARI)
```

```
-- Kapan perubahan massal terjadi
SELECT SUBSTR(CHAR(JOTSTP), 1, 16) AS menit,
       JOJOB, JOUSER, JOPGM,
       COUNT(*) AS jml_perubahan
  FROM QTEMP.JRNCARI
 WHERE JOENTT IN ('UP','DL')
 GROUP BY SUBSTR(CHAR(JOTSTP), 1, 16), JOJOB, JOUSER, JOPGM
HAVING COUNT(*) > 1000
 ORDER BY menit;
```

Lonjakan perubahan pada satu menit oleh satu program menunjuk waktu kejadian dengan tepat. Pilih `TOTIME` tepat sebelum itu.

85.5 Latihan pemulihan

Skenario latihan	Frekuensi	Berhasil bila
Restore satu program	Bulanan	Objek identik dengan aslinya
Restore satu tabel + apply journal	Triwulanan	Jumlah record dan saldo cocok
Restore satu library penuh ke UAT	Triwulanan	Aplikasi berjalan normal
Restore sistem penuh ke mesin uji	Tahunan	RTO tercapai, data lengkap
Baca media offsite tertua	Semesteran	Media terbaca tanpa kesalahan

Baris terakhir sering diabaikan. Media tape punya masa pakai, dan media yang disimpan bertahun-tahun di lokasi dengan pengendalian suhu yang buruk bisa tidak terbaca ketika akhirnya dibutuhkan. Ujilah sebelum Anda benar-benar bergantung padanya.

85.6 Dokumen pemulihan bencana

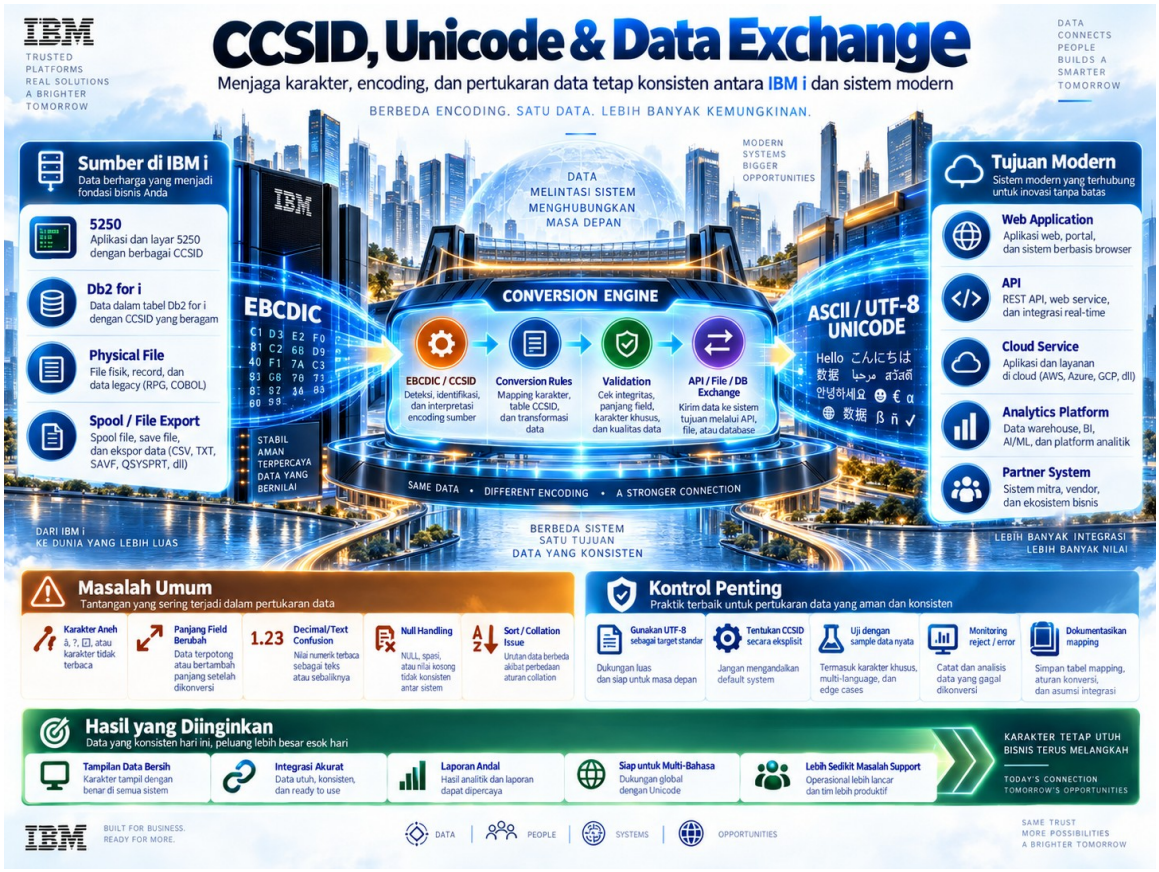
Satu berkas yang harus ada di luar sistem — dicetak, dan salinannya di lokasi cadangan:

```
DOKUMEN PEMULIHAN SISTEM

1. IDENTITAS SISTEM
   Nama sistem, serial, model, rilis OS
   Nomor kontrak dukungan IBM dan nomor telepon

2. KONFIGURASI DASAR
   Alamat IP, subnet, gateway, DNS
   Konfigurasi partisi: CPU, memori, disk
   Daftar perangkat: tape, printer, konsol

3. URUTAN PEMULIHAN
   Langkah demi langkah, dengan perintah persis
   Media apa yang dibutuhkan di tiap langkah

4. KATA SANDI DAN AKSES
   Prosedur mendapatkan kredensial darurat
   TIDAK berisi kata sandi itu sendiri

5. KONTAK
   Tim internal dengan cadangan
   Vendor aplikasi, nomor kontrak
   IBM support, nomor kontrak

6. VERIFIKASI
   Daftar periksa setelah pemulihan
   Angka kontrol yang harus cocok

7. RIWAYAT REVISI
```

Dokumen ini tidak berguna bila tersimpan hanya di sistem yang hilang. Cetak, perbarui setiap kali konfigurasi berubah, dan simpan salinannya di tempat yang bisa dicapai ketika data center utama tidak bisa.

Bagian XXIII — CCSID, Unicode, dan Pertukaran Data

Bab 86. Memahami CCSID

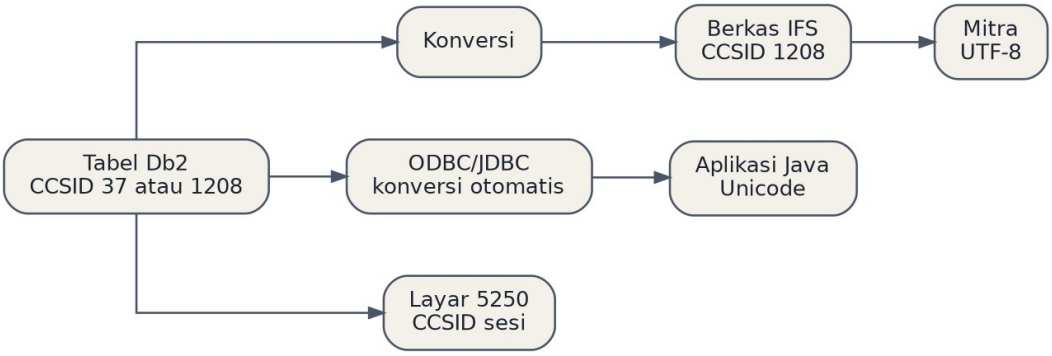


Gambar 92 — CCSID, Unicode, dan pertukaran data

Ini sumber masalah yang paling membingungkan bagi orang yang datang dari platform lain, dan penyebab keluhan nyata: nama nasabah yang berubah menjadi karakter aneh, berkas interface yang ditolak mitra, laporan dengan tanda kutip yang salah.

86.1 Akar masalahnya

IBM i secara historis memakai EBCDIC, sementara hampir seluruh dunia memakai ASCII atau UTF-8. Setiap kali data berpindah antara IBM i dan sistem lain, terjadi konversi. Konversi yang salah menghasilkan data rusak yang sering baru ketahuan berminggu-minggu kemudian.



Gambar 93

86.2 CCSID yang perlu dikenali

CCSID	Nama	Dipakai untuk
37	EBCDIC AS/400 Amerika	Bawaan banyak sistem lama
273, 277, 278	EBCDIC Eropa	Sistem di Eropa
500	EBCDIC internasional	Bawaan sebagian sistem
819	ISO 8859-1 Latin-1	ASCII Eropa Barat
1208	UTF-8	Standar pertukaran modern
1200	UTF-16	Unicode dua byte
65535	Tanpa konversi	Data biner

CCSID 65535 adalah jebakan. Ia berarti "jangan konversi apa pun". Kolom dengan CCSID ini tidak akan dikonversi saat diakses lewat ODBC atau ditulis ke IFS, sehingga data EBCDIC mentah sampai ke sistem yang mengharapkan teks. Untuk kolom teks, ini hampir selalu salah.

86.3 Memeriksa CCSID

```
/* CCSID sistem */
DSPSYSVAL SYSVAL(QCCSID)

/* CCSID job saat ini */
DSPJOB OPTION(*DFNA)

/* CCSID profil pengguna */
DSPUSRPRF USRPRF(BUDI001) TYPE(*BASIC)

/* CCSID berkas IFS */
DSPF STMF('/interface/out/data.txt')
```

```
-- CCSID tiap kolom teks di satu tabel
SELECT COLUMN_NAME, DATA_TYPE, LENGTH, CCSID,
       CASE CCSID
         WHEN 65535 THEN 'PERHATIAN: tanpa konversi'
         WHEN 1208  THEN 'UTF-8'
         WHEN 37    THEN 'EBCDIC AS'
         ELSE CHAR(CCSID)
       END AS keterangan
  FROM QSYS2.SYSCOLUMNS
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND TABLE_NAME = 'CIF_UTAMA'
   AND DATA_TYPE IN ('CHAR','VARCHAR','CLOB')
 ORDER BY ORDINAL_POSITION;

-- Cari seluruh kolom teks berCCSID 65535 di library produksi
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, LENGTH
  FROM QSYS2.SYSCOLUMNS
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND CCSID = 65535
   AND DATA_TYPE IN ('CHAR','VARCHAR')
 ORDER BY TABLE_NAME, COLUMN_NAME;
```

Kueri kedua layak dijalankan sekali di lingkungan Anda. Hasilnya sering mengejutkan.

86.4 Konversi berkas dengan benar

```
/* Dari member ke berkas IFS, ditulis sebagai UTF-8 */
CPYTOSTMF FROMMBR('/QSYS.LIB/APPWRK.LIB/EKSPOR.FILE/DATA.MBR') +
          TOSTMF('/interface/out/data.txt') +
          STMFOPT(*REPLACE) +
          STMFCCSID(1208) +
          ENDLINFMT(*LF)

/* Sebaliknya, membaca berkas UTF-8 ke member */
CPYFRMSTMF FROMSTMF('/interface/in/data.txt') +
           TOMBR('/QSYS.LIB/APPWRK.LIB/IMPOR.FILE/DATA.MBR') +
           MBROPT(*REPLACE) +
           STMFCCSID(1208)

/* Antar berkas IFS dengan konversi */
CPY OBJ('/interface/in/data.txt') +
    TOOBJ('/interface/kerja/data.txt') +
    TOCCSID(1208) +
    DTAFMT(*TEXT) +
    REPLACE(*YES)
```

Parameter	Pentingnya
`STMFCCSID`	Menentukan encoding berkas hasil
`ENDLINFMT`	`*LF` untuk Unix dan mitra modern, `*CRLF` untuk Windows
`DTAFMT(*TEXT)`	Melakukan konversi; `*BINARY` tidak
`TOCCSID`	Target konversi

Kesalahan `ENDLINFMT` yang sering terjadi. Berkas dengan akhir baris `CRLF` yang dikirim ke sistem Unix menghasilkan karakter tak terlihat di akhir setiap baris. Mitra melaporkan "field terakhir selalu salah", dan penyebabnya tidak terlihat di editor biasa.

```
# Memeriksa akhir baris dan encoding dari PASE
file /interface/out/data.txt
od -c /interface/out/data.txt | head -5
hexdump -C /interface/out/data.txt | head -5
```

86.5 Mendeteksi data yang sudah rusak

```
-- Karakter yang tidak masuk akal di kolom nama
SELECT ID_NASABAH, NAMA, HEX(NAMA) AS heksa
  FROM DTALIB.CIF_UTAMA
 WHERE TRANSLATE(NAMA,
         '                                        ',
         'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 .,-''()/')
       <> REPEAT(' ', LENGTH(NAMA))
 FETCH FIRST 100 ROWS ONLY;

-- Panjang karakter versus panjang byte, penanda masalah UTF-8
SELECT ID_NASABAH, NAMA,
       LENGTH(NAMA) AS panjang_byte,
       CHARACTER_LENGTH(NAMA) AS panjang_karakter
  FROM DTALIB.CIF_UTAMA
 WHERE LENGTH(NAMA) <> CHARACTER_LENGTH(NAMA)
 FETCH FIRST 50 ROWS ONLY;
```

Kolom `HEX()` adalah alat diagnosis terbaik. Ketika sebuah nama terlihat aneh, tampilkan heksanya dan bandingkan dengan nilai yang diharapkan — itu langsung menunjukkan apakah masalahnya konversi ganda, konversi salah arah, atau memang data masukan yang buruk.

86.6 Pilihan rancangan untuk sistem baru

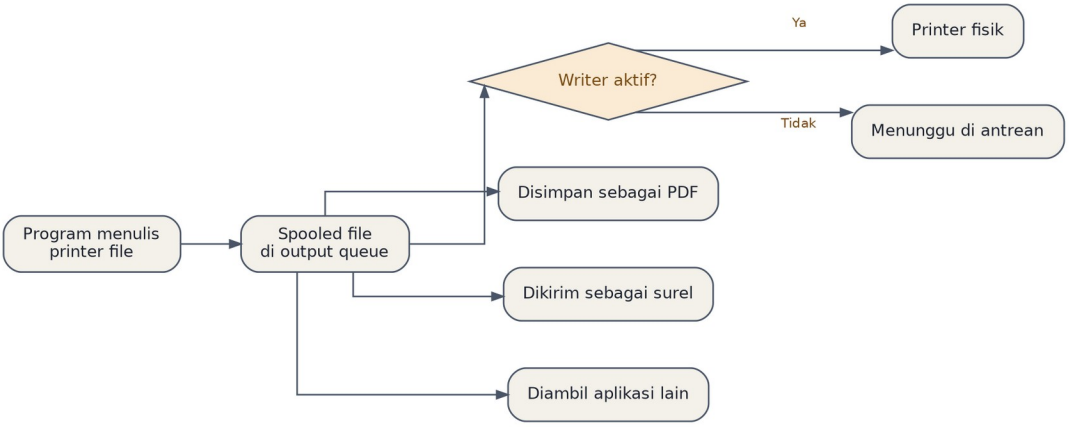
Pilihan	Kelebihan	Kekurangan
Tabel CCSID 37, konversi di tepi	Kompatibel dengan program lama	Konversi di banyak tempat
Kolom Unicode (`GRAPHIC`, `NVARCHAR`)	Mendukung semua bahasa	Program RPG lama perlu penyesuaian
Tabel CCSID 1208	Sederhana untuk aplikasi baru	Panjang byte berbeda dari panjang karakter

Untuk sistem perbankan Indonesia yang hanya memakai huruf Latin, CCSID 37 dengan konversi di tepi umumnya memadai dan paling sedikit mengganggu. Bila ada kebutuhan menyimpan nama dengan aksara lain, barulah Unicode menjadi keharusan.

Perangkap panjang field. Pada UTF-8, satu karakter bisa memakan lebih dari satu byte. Kolom `VARCHAR(60)` menampung 60 byte, bukan 60 karakter. Nama dengan karakter beraksen bisa terpotong di tengah, menghasilkan byte tidak lengkap yang merusak seluruh berkas ekspor.

Bab 87. Manajemen Keluaran dan Cetak

87.1 Alur spooled file



Gambar 94

87.2 Perintah dasar

```
/* Melihat spooled file */
WRKSPLF
WRKSPLF SELECT(EODUSER)
WRKOUTQ OUTQ(APPPGM/LAPORAN)

/* Mengelola writer */
STRPRTWTR DEV(PRT01) OUTQ(APPPGM/CETAK)
ENDWTR WTR(PRT01) OPTION(*IMMED)
WRKWTR WTR(*ALL)

/* Memindahkan dan mengubah */
CHGSPLFA FILE(LAPORAN) JOB(123456/EODUSER/EODRPT) +
         SPLNBR(1) OUTQ(APPPGM/ARSIP)

HLDSPLF FILE(LAPORAN) JOB(123456/EODUSER/EODRPT) SPLNBR(1)
RLSSPLF FILE(LAPORAN) JOB(123456/EODUSER/EODRPT) SPLNBR(1)
DLTSPLF FILE(LAPORAN) JOB(123456/EODUSER/EODRPT) SPLNBR(1)
```

87.3 Mengubah spooled file menjadi PDF

```
/* Transformasi bawaan ke PDF di IFS */
CPYSPLF FILE(LAPORAN) +
        JOB(123456/EODUSER/EODRPT) +
        SPLNBR(1) +
        TOSTMF('/arsip/laporan/2026-09-18-neraca.txt') +
        STMFOPT(*REPLACE) +
        WSCST(*NONE)

/* Untuk PDF sesungguhnya, dipakai output queue dengan
   transformasi, atau produk transform */
```

Untuk arsip laporan regulator, pola yang lazim adalah mengarahkan spooled file ke antrean khusus, lalu program terjadwal menyalinnya ke IFS sebagai PDF dengan penamaan yang konsisten dan checksum tersimpan.

```
-- Spooled file yang menunggu diarsipkan
SELECT SPOOLED_FILE_NAME, JOB_NAME, FILE_NUMBER,
       USER_NAME, CREATE_TIMESTAMP, TOTAL_PAGES,
       SIZE / 1024 AS kb
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
 WHERE OUTPUT_QUEUE_NAME = 'REGULATOR'
   AND STATUS = 'READY'
 ORDER BY CREATE_TIMESTAMP;
```

87.4 Kebijakan retensi keluaran

Jenis keluaran	Retensi	Alasan
Laporan operasional harian	30 hari	Kebutuhan operasional
Laporan kontrol EOD	1 tahun	Penelusuran insiden
Laporan regulator	Sesuai ketentuan, sering 5–10 tahun	Kewajiban hukum
Rekening koran nasabah	Sesuai ketentuan	Kewajiban hukum
Log teknis	90 hari	Analisis kinerja
Keluaran uji	7 hari	Tidak bernilai jangka panjang

Peringatan. Kebijakan retensi TI tidak boleh lebih pendek dari kewajiban retensi regulatoris. Pembersihan otomatis yang menghapus laporan regulator adalah temuan audit serius, dan pemulihannya sering tidak mungkin.

```
-- Antrean keluaran yang tidak pernah dibersihkan
SELECT OUTPUT_QUEUE_LIBRARY, OUTPUT_QUEUE_NAME,
       COUNT(*) AS jml_berkas,
       SUM(SIZE) / 1048576 AS total_mb,
       MIN(CREATE_TIMESTAMP) AS tertua
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
 GROUP BY OUTPUT_QUEUE_LIBRARY, OUTPUT_QUEUE_NAME
HAVING COUNT(*) > 5000
 ORDER BY total_mb DESC;
```

87.5 Distribusi laporan

```
/* Mengirim spooled file sebagai lampiran surel */
SNDDST TYPE(*DOC) TOINTNET(('kepala.cabang@bank.co.id')) +
       DSTD('Laporan harian cabang') +
       DOC(LAPORAN) FOLDER(CETAK)
```

Cara yang lebih modern dan lebih mudah dikendalikan adalah menyalin ke IFS lalu mengirim lewat skrip PASE:

```
#!/QOpenSys/pkgs/bin/bash
BERKAS=/arsip/laporan/$(date +%Y-%m-%d)-cabang.pdf

if [ ! -f "$BERKAS" ]; then
  echo "Berkas tidak ditemukan: $BERKAS" >&2
  exit 1
fi

# Kirim lewat relay surel internal
# Kredensial diambil dari penyimpanan terkendali, bukan ditulis di skrip
```

Prinsip yang perlu dijaga. Laporan yang berisi data nasabah tidak dikirim sebagai lampiran surel tanpa enkripsi. Pola yang lebih aman: simpan di lokasi terkendali, kirim pemberitahuan berisi tautan, dan catat siapa mengunduhnya.

⚙️ Bab 88. Pertukaran Data dengan Sistem Lain

88.1 Metode dan kapan dipakai

Metode	Cocok untuk	Perhatian
Berkas CSV di IFS	Batch besar, mitra sederhana	CCSID dan akhir baris
Berkas posisi tetap	Mitra lama, format regulator	Padding dan perataan
JSON lewat REST	Integrasi modern	Ukuran dan timeout
XML	Mitra korporat, standar lama	Verbose, lambat diurai
ODBC langsung	Pelaporan internal	Wajib exit program
MQ	Pertukaran andal	Perlu infrastruktur

88.2 Menghasilkan CSV dengan benar

```
-- Ekspor langsung ke IFS dengan CCSID yang benar
CALL QSYS2.QCMDEXC('
  CPYTOIMPF FROMFILE(APPWRK/EKSPOR)
            TOSTMF(''/interface/out/rekening.csv'')
            MBROPT(*REPLACE)
            STMFCCSID(1208)
            RCDDLM(*LF)
            STRDLM(*DBLQUOTE)
            FLDDLM('','')
            ADDCOLNAM(*SQL)');
```

Parameter	Kegunaan
`STMFCCSID(1208)`	UTF-8
`RCDDLM(*LF)`	Akhir baris Unix
`STRDLM(*DBLQUOTE)`	Pembungkus teks
`FLDDLM`	Pemisah kolom
`ADDCOLNAM`	Sertakan baris judul

Masalah yang harus diantisipasi. Data yang mengandung pemisah kolom, tanda kutip, atau baris baru akan merusak berkas CSV bila tidak dibungkus dengan benar. Bersihkan data sebelum ekspor:

```
-- Membersihkan karakter yang merusak CSV
SELECT NO_REK,
       TRANSLATE(NAMA, '   ', ',' || CHR(13) || CHR(10)) AS nama_bersih,
       SALDO
  FROM DTALIB.REKENING;

-- Deteksi data bermasalah sebelum ekspor
SELECT COUNT(*) AS baris_bermasalah
  FROM DTALIB.REKENING
 WHERE POSITION(',' IN NAMA) > 0
    OR POSITION(CHR(13) IN NAMA) > 0
    OR POSITION(CHR(10) IN NAMA) > 0
    OR POSITION('"' IN NAMA) > 0;
```

88.3 Memuat berkas masuk

```
CPYFRMIMPF FROMSTMF('/interface/in/mutasi.csv') +
           TOFILE(APPWRK/STG_MUTASI) +
           MBROPT(*REPLACE) +
           RCDDLM(*LF) +
           STRDLM(*DBLQUOTE) +
           FLDDLM(',') +
           FROMRCD(2) +
           ERRLVL(0)
```

`ERRLVL(0)` berarti tidak ada baris rusak yang ditoleransi. Untuk berkas keuangan ini pilihan yang benar: lebih baik seluruh berkas ditolak daripada sebagian dimuat dengan diam-diam.

```
-- Validasi setelah pemuatan, sebelum diproses
SELECT 'Jumlah baris' AS pemeriksaan, COUNT(*) AS nilai
  FROM APPWRK.STG_MUTASI
UNION ALL
SELECT 'Nomor rekening tidak valid',
       COUNT(*) FROM APPWRK.STG_MUTASI
 WHERE LENGTH(TRIM(NO_REK)) <> 16
    OR TRANSLATE(NO_REK, '                ', '0123456789')
       <> REPEAT(' ', 16)
UNION ALL
SELECT 'Nominal nol atau negatif',
       COUNT(*) FROM APPWRK.STG_MUTASI WHERE NOMINAL <= 0
UNION ALL
SELECT 'Rekening tidak dikenal',
       COUNT(*) FROM APPWRK.STG_MUTASI s
 WHERE NOT EXISTS (SELECT 1 FROM DTALIB.REKENING r
                   WHERE r.NO_REK = s.NO_REK);
```

88.4 Format posisi tetap

Masih lazim untuk pelaporan regulator dan mitra lama.

```
**free
// Menulis baris dengan lebar tetap
dcl-ds dsBaris qualified;
  kdBank    char(3);
  noRek     char(16);
  nama      char(50);
  saldo     char(18);   // angka sebagai teks, rata kanan, nol di depan
  tglLapor  char(8);    // YYYYMMDD
  filler    char(5);
end-ds;

dsBaris.kdBank   = '014';
dsBaris.noRek    = wNoRek;
dsBaris.nama     = %subst(wNama + *blanks: 1: 50);
dsBaris.saldo    = %editc(wSaldo: 'X');   // nol di depan
dsBaris.tglLapor = %char(%date(): *iso0);
dsBaris.filler   = *blanks;

write(e) BERKASOUT dsBaris;
```

Tiga kesalahan yang paling sering. Pertama, perataan salah — teks harus rata kiri dengan spasi, angka rata kanan dengan nol. Kedua, panjang tidak persis sesuai spesifikasi, sehingga seluruh baris bergeser. Ketiga, tanda negatif yang diperlakukan berbeda dari yang diharapkan regulator.

```
# Verifikasi panjang baris konsisten
awk '{print length}' /interface/out/lapor.txt | sort -u
# Harus menghasilkan satu angka saja
```

Satu baris perintah itu menangkap sebagian besar kesalahan format sebelum berkas dikirim.

88.5 Menangani berkas besar

Ukuran	Pertimbangan
Di bawah 100 MB	Tidak ada masalah khusus
100 MB sampai 1 GB	Perhatikan ruang IFS dan waktu konversi
Di atas 1 GB	Pecah menjadi beberapa berkas, atau alirkan

Memecah berkas besar menjadi beberapa bagian dengan penomoran dan berkas kontrol yang mencatat jumlah bagian adalah pola yang lebih andal daripada satu berkas raksasa. Bila satu bagian gagal, hanya bagian itu yang dikirim ulang.

🔍 Bagian XXIV — Studi Kasus Troubleshooting

Sepuluh kasus berikut disusun dari pola masalah yang berulang di lingkungan core banking. Setiap kasus ditulis dengan struktur yang sama: gejala, jalan buntu yang ditempuh lebih dulu, diagnosis yang benar, akar masalah, perbaikan, dan pelajaran.

⚡ Bab 89. Kasus Kinerja

89.1 Kasus 1 — Cabang tertentu lambat, cabang lain normal

Gejala. Teller di satu cabang melaporkan setiap transaksi memakan 8–12 detik. Cabang lain di kota yang sama normal.

Jalan buntu. Tim memeriksa beban CPU dan disk di IBM i selama tiga hari. Semuanya normal. Dugaan berpindah ke aplikasi, lalu ke perangkat teller.

Diagnosis yang benar. Ping dari IBM i ke cabang tersebut dengan ukuran paket bertingkat.

```
PING RMTSYS('10.40.12.5') PKTLEN(64)    /* berhasil */
PING RMTSYS('10.40.12.5') PKTLEN(1472)  /* gagal */
```

Akar masalah. Jalur ke cabang itu melewati tautan dengan MTU lebih kecil setelah penggantian perangkat oleh penyedia jaringan. Paket besar difragmentasi atau dibuang, sehingga setiap transaksi harus menunggu pengiriman ulang.

Perbaikan. Penyedia menyesuaikan MTU jalur. Sebagai penahan sementara, MTU di sisi interface diturunkan.

Pelajaran. Ping dasar membuktikan jangkauan, bukan kelayakan jalur. Gejala "berfungsi tetapi lambat" hampir tidak pernah terdeteksi oleh ping ukuran bawaan. Masukkan uji MTU ke dalam runbook diagnosis jaringan.

89.2 Kasus 2 — EOD melambat setiap akhir bulan

Gejala. EOD biasa 3 jam. Pada hari terakhir bulan, 7 jam, kadang melewati batch window.

Jalan buntu. Diasumsikan wajar karena volume EOM memang lebih besar. Selama setahun tidak diselidiki.

Diagnosis yang benar. Perbandingan durasi per fase antara hari biasa dan hari EOM.

```
SELECT NM_FASE,
       AVG(CASE WHEN DAY(TGL_BUKU) = DAY(LAST_DAY(TGL_BUKU))
                THEN TIMESTAMPDIFF(4, CHAR(JAM_SELESAI - JAM_MULAI)) END) AS eom,
       AVG(CASE WHEN DAY(TGL_BUKU) <> DAY(LAST_DAY(TGL_BUKU))
                THEN TIMESTAMPDIFF(4, CHAR(JAM_SELESAI - JAM_MULAI)) END) AS biasa
  FROM APPWRK.EOD_KONTROL
 WHERE STATUS = 'S' AND TGL_BUKU > CURRENT DATE - 6 MONTHS
 GROUP BY NM_FASE, URUTAN
 ORDER BY URUTAN;
```

Hasilnya: sebagian besar fase naik 20–30% pada EOM, wajar. Tetapi satu fase pelaporan naik dari 12 menit menjadi 3 jam 40 menit.

Akar masalah. Laporan bulanan memakai kueri yang mengagregasi seluruh tabel riwayat sejak awal, bukan hanya periode berjalan. Tabel itu tumbuh setiap bulan, sehingga kueri melambat secara kuadratik.

Perbaikan. Tabel ringkasan saldo harian, sehingga laporan bulanan menjumlahkan 30 baris ringkasan alih-alih memindai ratusan juta baris transaksi.

Pelajaran. Angka rata-rata menyembunyikan masalah. Selalu bandingkan per fase, bukan total.

89.3 Kasus 3 — Sistem melambat setiap Selasa pukul 10

Gejala. Keluhan lambat yang berulang dengan pola waktu yang sangat konsisten.

Diagnosis. Pola waktu yang konsisten hampir selalu berarti pekerjaan terjadwal.

```
SELECT SCHEDULED_JOB_NAME, FREQUENCY, SCHEDULE_DATE,
       SCHEDULE_TIME, SCHEDULE_DAYS, COMMAND_STRING, STATUS
  FROM QSYS2.SCHEDULED_JOB_INFO
 WHERE STATUS = '*SCDE'
 ORDER BY SCHEDULE_TIME;
```

Akar masalah. Ekstraksi data ke gudang data dijadwalkan mingguan pukul sepuluh pagi, dijalankan dengan `DEGREE(*MAX)`. Kueri paralel menyerap seluruh CPU yang tersedia.

Perbaikan. Jadwal dipindah ke akhir pekan, dan `CHGQRYA DEGREE(*NONE)` ditambahkan di awal job sebagai pengaman.

Pelajaran. Pola waktu adalah petunjuk terkuat dalam diagnosis kinerja. Sebelum menganalisis apa pun, tanyakan: apakah ini terjadi pada waktu yang sama?

Bab 90. Kasus Data

90.1 Kasus 4 — Nama nasabah rusak setelah integrasi baru

Gejala. Sebagian nama nasabah di aplikasi mobile tampil dengan karakter aneh. Di layar 5250 normal.

Diagnosis.

```
SELECT ID_NASABAH, NAMA, HEX(NAMA) AS heksa
  FROM DTALIB.CIF_UTAMA
 WHERE ID_NASABAH = 12345;
```

Heksa menunjukkan data di tabel benar. Masalahnya ada pada jalur keluar.

```
SELECT COLUMN_NAME, CCSID
  FROM QSYS2.SYSCOLUMNS
 WHERE TABLE_SCHEMA = 'DTALIB' AND TABLE_NAME = 'CIF_UTAMA'
   AND COLUMN_NAME = 'NAMA';
```

Hasilnya: CCSID 65535.

Akar masalah. Kolom dibuat bertahun-tahun lalu tanpa CCSID. Layar 5250 menampilkan benar karena tidak ada konversi yang dibutuhkan. Aplikasi mobile mengambil lewat ODBC, yang tidak mengonversi kolom berCCSID 65535, sehingga byte EBCDIC mentah sampai ke aplikasi yang mengharapkan teks.

Perbaikan. CCSID kolom diubah ke 37, setelah verifikasi di UAT bahwa seluruh program yang memakainya tetap bekerja.

Pelajaran. Jalankan pemindaian CCSID 65535 di seluruh tabel produksi sebelum integrasi baru, bukan setelah keluhan masuk.

90.2 Kasus 5 — Berkas interface ditolak mitra setiap hari Senin

Gejala. Mitra melaporkan berkas gagal diurai, tetapi hanya pada hari Senin.

Diagnosis.

```
awk '{print length}' /interface/out/lapor-senin.txt | sort -u
# menghasilkan dua angka: 250 dan 251
```

Akar masalah. Baris dengan panjang 251 adalah transaksi akhir pekan yang jumlahnya lebih besar, sehingga field nominal meluap satu posisi. Program tidak memvalidasi panjang hasil, dan format posisi tetap langsung rusak.

Perbaikan. Panjang field nominal diperbesar sesuai spesifikasi mitra, dan validasi panjang baris ditambahkan sebelum berkas dikirim.

Pelajaran. Satu baris `awk` yang memeriksa keseragaman panjang baris menangkap seluruh kelas kesalahan format. Pasang sebagai gerbang wajib sebelum pengiriman.

90.3 Kasus 6 — Saldo satu rekening berbeda antara layar dan laporan

Gejala. Rekening menunjukkan saldo berbeda antara inquiry di cabang dan laporan harian.

Diagnosis.

```
-- Saldo di tabel rekening
SELECT SALDO FROM DTALIB.REKENING WHERE NO_REK = '...';

-- Saldo dihitung dari mutasi
SELECT SUM(CASE WHEN JENIS = 'K' THEN JUMLAH ELSE -JUMLAH END)
  FROM DTALIB.TRXHIST WHERE NO_REK = '...';
```

Keduanya berbeda. Penelusuran journal menemukan waktunya.

```
DSPJRN JRN(JRNLIB/JRNDTA) FILE((DTALIB/REKENING)) +
       RCVRNG(*CURCHAIN) ENTTYP(*RCD) +
       OUTPUT(*OUTFILE) OUTFILE(QTEMP/JRNREK)
```

Akar masalah. Sebuah program koreksi yang dijalankan manual memperbarui saldo tanpa menulis mutasi pendamping, dan berjalan di luar commitment control.

Perbaikan. Saldo dikoreksi dengan mutasi penyeimbang yang tercatat, bukan dengan pembaruan langsung. Program koreksi dicabut dari produksi dan diganti prosedur bermodal dual control.

Pelajaran. Program yang bisa mengubah saldo tanpa menulis mutasi tidak boleh ada di produksi. Lakukan inventarisasi program semacam ini.

```
-- Program yang menulis ke tabel rekening
SELECT WHPNAM AS program, WHLNAM AS library, WHUSAG
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'REKENING'
   AND WHUSAG IN ('2','3');   -- update dan output
```

⚙️ Bab 91. Kasus Operasional

91.1 Kasus 7 — EOD menggantung tanpa pesan

Gejala. Fase posting berstatus berjalan selama empat jam tanpa progres, tanpa pesan apa pun.

Diagnosis.

```
SELECT JOB_NAME, JOB_STATUS, FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(SUBSYSTEM_LIST_FILTER => 'EODSBS'));
```

Status `DEQW` — menunggu data queue.

```
WRKJOB JOB(123456/EODUSER/EODPOST) OPTION(*PGMSTK)
```

Call stack menunjukkan program berhenti di `QRCVDTAQ`.

Akar masalah. Program posting menunggu sinyal dari job pendamping lewat data queue. Job pendamping gagal beberapa jam sebelumnya, dan pemanggilan `QRCVDTAQ` memakai waktu tunggu tak terbatas.

Perbaikan. Seluruh pemanggilan `QRCVDTAQ` diberi batas waktu, dan kedaluwarsanya memicu escape message alih-alih menunggu selamanya.

Pelajaran. Setiap operasi menunggu di program batch harus punya batas waktu. Tidak ada pengecualian. Periksa seluruh `QRCVDTAQ`, `DLYJOB` berulang, dan `ALCOBJ WAIT(*NOMAX)` di kode produksi Anda.

91.2 Kasus 8 — Backup berhasil tetapi restore gagal

Gejala. Uji restore triwulanan gagal. Backup harian dilaporkan selesai selama enam bulan.

Diagnosis.

```
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, MESSAGE_TEXT
  FROM TABLE(QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 180 DAYS))
 WHERE MESSAGE_ID = 'CPF3730';
```

Pesan `CPF3730` muncul setiap malam selama enam bulan.

Akar masalah. Satu library ditambahkan ke sistem setelah control group dibuat, dan tidak pernah dimasukkan ke daftar backup. Job backup melaporkan selesai karena library lain berhasil; `CPF3730` yang menandai objek tidak tersimpan tidak pernah dibaca siapa pun.

Perbaikan. Control group diperbarui. Pemeriksaan `CPF3730` ditambahkan ke daftar verifikasi harian. Dibuat kueri pembanding antara daftar library yang ada dan daftar yang di-backup.

```
-- Library yang ada tetapi tidak pernah muncul di riwayat backup
SELECT DISTINCT OBJNAME AS library
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*LIB'))
 WHERE OBJNAME NOT IN (
       SELECT DISTINCT LIBRARY_NAME FROM QUSRBRM.SAVE_HISTORY
        WHERE SAVE_TIMESTAMP > CURRENT DATE - 7 DAYS);
```

Pelajaran. Status "selesai" bukan bukti keberhasilan. Kueri di atas layak dijadwalkan mingguan di lingkungan mana pun.

91.3 Kasus 9 — Kanal mobile putus setelah uji DR

Gejala. Setelah switchover terencana ke DR, seluruh fungsi mobile banking gagal. Kanal lain normal.

Diagnosis. Koneksi dari middleware ke IBM i ditolak. Uji dari sisi middleware menunjukkan koneksi tidak terbentuk.

Akar masalah. Middleware memakai daftar putih alamat IP. Partisi DR punya alamat berbeda dari partisi produksi, dan alamat itu tidak pernah didaftarkan.

Perbaikan. Alamat DR ditambahkan ke daftar putih seluruh mitra. Virtual IP yang berpindah saat failover diterapkan agar alamat tetap sama.

Pelajaran. Uji DR yang berhenti pada "sistem naik" tidak menguji apa pun yang penting. Uji harus sampai satu transaksi berhasil dari setiap kanal, dan daftar kanal harus lengkap.

91.4 Kasus 10 — Seluruh sesi terputus di tengah hari

Gejala. Seluruh sesi 5250 terputus serentak. Sistem tetap berjalan, batch tidak terganggu.

Diagnosis.

```
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, MESSAGE_TEXT, FROM_USER
  FROM TABLE(QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 2 HOURS))
 WHERE MESSAGE_ID IN ('CPF1164','CPF2451','CPI0999')
 ORDER BY MESSAGE_TIMESTAMP;
```

Subsystem QINTER berakhir. Penelusuran audit journal menunjukkan siapa yang menjalankannya.

Akar masalah. Seorang staf menjalankan prosedur pemeliharaan yang dimaksudkan untuk lingkungan UAT, tetapi sesi yang terbuka menunjuk ke produksi. Program tidak memeriksa nama sistem sebelum bertindak.

Perbaikan. Seluruh program pemeliharaan diberi pemeriksaan nama sistem di awal, seperti pola di Bab 30.2. Tampilan sesi produksi diberi warna dan judul berbeda agar keliru lebih sulit terjadi.

Pelajaran. Pencegahan kesalahan manusia lebih efektif daripada pelatihan kehati-hatian. Program yang

berbahaya harus menolak berjalan di lingkungan yang salah.

Bab 92. Pola yang Berulang

Dari sepuluh kasus di atas dan pengalaman lapangan pada umumnya, muncul beberapa pola yang layak dijadikan naluri.

92.1 Pertanyaan yang membuka jalan

Pertanyaan	Kenapa berguna
Kapan terakhir kali ini berjalan normal?	Mempersempit jendela perubahan
Apa yang berubah sejak saat itu?	Penyebab hampir selalu perubahan
Apakah terjadi pada waktu yang sama?	Menunjuk pekerjaan terjadwal
Apakah terjadi pada semua, atau sebagian?	Membedakan masalah sistem dan masalah jalur
Apakah ada pesan, atau diam saja?	Kegagalan diam butuh pendekatan berbeda
Apakah angka kontrolnya cocok?	Membedakan masalah tampilan dan masalah data

Pertanyaan pertama adalah yang paling kuat. "Sudah lama begini" dan "mulai Selasa lalu" mengarah ke penyelidikan yang sepenuhnya berbeda.

92.2 Tanda bahaya yang sering diabaikan

1. Pesan yang muncul setiap hari dan dianggap normal
2. Proses yang durasinya perlahan naik
3. Backup yang ukurannya berubah tanpa penjelasan
4. Job yang selalu perlu "di-restart saja"
5. Prosedur yang hanya satu orang bisa jalankan
6. Angka yang dicocokkan manual setiap bulan
7. Perubahan yang dilakukan di luar jam kerja tanpa tiket

Setiap butir di daftar ini adalah insiden yang belum terjadi. Menanganinya saat masih berupa gangguan kecil jauh lebih murah daripada menanganinya pada pukul tiga pagi.

92.3 Setelah insiden selesai

Langkah	Dalam waktu
Catat kronologi dengan stempel waktu	Segera, saat masih ingat
Tulis analisis akar masalah	3 hari
Buat atau perbarui runbook	7 hari
Tambahkan pemantauan yang akan mendeteksi lebih awal	14 hari
Tambahkan uji yang akan menangkapnya	30 hari
Tinjau apakah pola serupa ada di tempat lain	30 hari

Baris terakhir adalah yang paling sering dilewati dan paling bernilai. Bila satu program menunggu tanpa batas waktu, kemungkinan besar ada sepuluh program lain yang sama. Perbaiki satu, lalu cari sembilan sisanya.

👥 Bagian XXV — Operasi Tim

Platform yang dikelola dengan baik oleh tim yang kacau tetap menghasilkan insiden. Bagian ini tentang sisi manusia dari pekerjaan.

Bab 93. Jaga dan Eskalasi

93.1 Merancang rotasi jaga

Aspek	Pertimbangan
Panjang giliran	Satu minggu; lebih pendek memutus konteks, lebih panjang melelahkan
Jumlah orang dalam rotasi	Minimal empat; di bawah itu tidak berkelanjutan
Cadangan	Selalu ada orang kedua yang bisa dihubungi
Kompensasi	Harus ada, formal, tertulis
Pemulihan	Yang jaga semalam tidak bekerja penuh esok harinya
Batas eskalasi	Kapan membangunkan orang yang lebih senior

Rotasi dengan dua orang bukan rotasi, melainkan dua orang yang tidak pernah benar-benar libur. Ini penyebab kelelahan dan perputaran staf yang tinggi, dan di lingkungan dengan sedikit ahli IBM i, kehilangan satu orang sangat mahal.

93.2 Apa yang harus ada sebelum giliran dimulai

- ☐ Akses ke seluruh sistem yang mungkin dibutuhkan, sudah diuji
- ☐ Nomor kontak eskalasi, termasuk vendor dan IBM
- ☐ Nomor kontrak dukungan, bisa diakses tanpa membuka sistem
- ☐ Runbook tersedia di luar sistem yang mungkin mati
- ☐ Daftar perubahan yang baru diterapkan minggu ini
- ☐ Daftar masalah yang sedang berjalan
- ☐ Jadwal batch dan jam kritikal

Butir ketiga sering terlupakan. Nomor kontrak IBM yang hanya tersimpan di wiki yang berjalan di sistem yang sedang mati tidak membantu siapa pun.

93.3 Tingkat keparahan

Tingkat	Definisi	Respons	Eskalasi
1 Kritis	Layanan nasabah berhenti	Segera, 24 jam	Manajer TI langsung
2 Tinggi	Sebagian layanan terganggu, atau EOD gagal	30 menit	Manajer TI dalam 1 jam
3 Menengah	Fungsi terganggu, ada jalan pintas	Jam kerja berikutnya	Bila melewati 1 hari
4 Rendah	Gangguan kecil, tidak mendesak	Dijadwalkan	Tidak

Definisi harus objektif. "Layanan nasabah berhenti" lebih baik daripada "dampak besar", karena yang pertama bisa diverifikasi pada pukul tiga pagi dan yang kedua bisa diperdebatkan.

93.4 Kapan membangunkan orang

Ini pertanyaan yang membuat orang ragu, dan keraguan itu memperpanjang insiden. Tetapkan aturan yang jelas:

1. Bila ada keraguan tentang keputusan yang tidak bisa dibatalkan — bangunkan
2. Bila insiden melewati batas waktu yang ditetapkan — bangunkan, tanpa menunggu diminta
3. Bila keputusan menuntut dua orang — bangunkan, itu memang aturannya
4. Bila Anda sudah mencoba lebih dari 30 menit tanpa kemajuan — bangunkan

Tidak ada seorang pun yang pernah dipecat karena membangunkan manajernya untuk insiden produksi perbankan. Banyak yang menyesal karena tidak melakukannya.

93.5 Komando insiden

Untuk insiden tingkat 1, tetapkan peran secara eksplisit sejak menit pertama.

Peran	Tugas
Komandan insiden	Mengoordinasi, mengambil keputusan, tidak mengerjakan teknis
Teknis utama	Mengerjakan perbaikan
Komunikasi	Memberi kabar ke bisnis dan manajemen
Pencatat	Mencatat kronologi dengan waktu

Komandan insiden tidak boleh sekaligus menjadi teknis utama. Orang yang sedang menelusuri job log tidak bisa sekaligus menjawab telepon manajemen dan memutuskan kapan eskalasi. Pada tim kecil, komandan bisa dipegang siapa saja yang tidak sedang mengetik.

93.6 Komunikasi selama insiden

Templat pembaruan yang dikirim tiap 30 menit:

```
INSIDEN [nomor] - PEMBARUAN [n]
Waktu: [   ]

STATUS: [Investigasi / Perbaikan berjalan / Verifikasi / Selesai]

DAMPAK:
  Layanan terpengaruh:
  Perkiraan jumlah nasabah terdampak:

YANG DIKETAHUI:
  [fakta saja, bukan dugaan]

YANG SEDANG DIKERJAKAN:

PERKIRAAN PULIH: [waktu, atau: belum dapat diperkirakan]

PEMBARUAN BERIKUTNYA: [waktu]
```

Kalimat "belum dapat diperkirakan" lebih baik daripada tebakan optimis yang meleset. Tebakan yang meleset menghancurkan kepercayaan dan memaksa Anda menjelaskan dua hal sekaligus.

Bab 94. Dokumentasi yang Bertahan

94.1 Kenapa dokumentasi gagal

Sebab	Penanggulangan
Ditulis sekali lalu usang	Perbarui saat memakainya
Terlalu panjang untuk dibaca saat darurat	Pisahkan runbook dari penjelasan
Tersimpan di tempat yang tidak dicari orang	Satu tempat, semua tahu
Menjelaskan apa, bukan kenapa	Catat alasan keputusan
Ditulis untuk penulisnya sendiri	Uji dengan orang yang belum tahu

Uji yang paling jujur: berikan runbook kepada orang yang belum pernah menangani masalah itu, dan minta ia menjalankannya tanpa bertanya. Setiap kali ia harus bertanya, di situlah dokumentasi bocor.

94.2 Tiga jenis dokumen yang berbeda

Jenis	Untuk	Bentuk
Runbook	Dijalankan saat darurat	Langkah, perintah persis, tanpa penjelasan panjang
Dokumen arsitektur	Memahami sistem	Diagram, keputusan dan alasannya
Catatan keputusan	Mengingat kenapa	Konteks, pilihan yang dipertimbangkan, alasan

Mencampur ketiganya menghasilkan dokumen yang terlalu panjang untuk darurat dan terlalu dangkal untuk memahami.

94.3 Catatan keputusan arsitektur

```
KEPUTUSAN [nomor]: [judul singkat]
Tanggal: [   ]   Status: [Diusulkan / Diterima / Digantikan]

KONTEKS
  Situasi yang menuntut keputusan ini.

PILIHAN YANG DIPERTIMBANGKAN
  1. [pilihan] — kelebihan, kekurangan
  2. [pilihan] — kelebihan, kekurangan

KEPUTUSAN
  Yang dipilih dan alasannya.

KONSEKUENSI
  Yang menjadi lebih mudah:
  Yang menjadi lebih sulit:
  Yang harus dipantau:
```

Dokumen sependek satu halaman ini menjawab pertanyaan yang paling sering muncul bertahun-tahun kemudian: kenapa dulu dibuat begini? Tanpa itu, penerus Anda akan menganggap keputusan yang cermat sebagai kecerobohan, dan mengubahnya tanpa tahu apa yang dulu dipertimbangkan.

94.4 Serah terima

Ketika seseorang meninggalkan peran, yang harus berpindah bukan hanya dokumen.

Yang dipindahkan	Cara
Akses dan kredensial	Dicabut dari yang lama, diberikan ke yang baru, tercatat
Pengetahuan tertulis	Runbook, dokumen arsitektur, catatan keputusan
Pengetahuan tak tertulis	Pendampingan langsung minimal empat minggu
Hubungan	Perkenalan langsung ke vendor, mitra, tim lain
Konteks historis	Sesi bercerita: kenapa sistem ini seperti ini
Pekerjaan yang sedang berjalan	Daftar dengan status dan langkah berikutnya

Baris kelima terdengar tidak formal tetapi sering paling bernilai. Sistem berusia dua puluh tahun menyimpan banyak keputusan yang masuk akal pada zamannya dan tampak aneh sekarang. Orang yang tahu ceritanya adalah aset yang akan hilang bila tidak direkam.

94.5 Mengurangi ketergantungan pada satu orang

```
-- Siapa yang menjalankan apa, dari audit journal
SELECT USER_NAME,
       COUNT(*) AS jml_tindakan,
       COUNT(DISTINCT OBJECT_NAME) AS jml_objek
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'CD',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 90 DAYS))
 GROUP BY USER_NAME
 ORDER BY jml_tindakan DESC;
```

Bila satu nama mendominasi secara mencolok, itu peta risiko Anda. Area yang hanya disentuh satu orang selama sembilan puluh hari adalah area yang akan berhenti bila orang itu tidak ada.

Cara mengurangi tanpa memperlambat pekerjaan:

1. Pasangkan satu junior pada setiap area kritikal, dengan tugas nyata bukan sekadar mengamati
2. Rotasi tugas rutin setiap triwulan
3. Wajibkan runbook sebelum suatu prosedur dianggap selesai dibangun
4. Jadikan latihan DR sebagai kesempatan orang lain memegang kemudi
5. Rekam sesi penyelesaian masalah nyata sebagai bahan belajar

👥 Bab 95. Membangun Kemampuan Tim

95.1 Jalur belajar untuk anggota baru

Bulan	Fokus	Hasil yang diharapkan
1	Navigasi dasar, layar hijau, membaca job log	Bisa memeriksa status sistem sendiri
2	Work management, subsystem, job queue	Memahami kenapa job berperilaku begitu
3	CL dasar, IBM i Services	Bisa menulis utilitas operasional sederhana
4	Backup, journaling, restore	Bisa melakukan restore satu objek
5	Kinerja dan pemantauan	Bisa membaca `WRKSYSSTS` dan menyimpulkan
6	Keamanan dan audit	Bisa menjawab pertanyaan audit

Bulan	Fokus	Hasil yang diharapkan
		dasar
7–12	EOD, core banking, pendampingan jaga	Siap masuk rotasi jaga dengan cadangan

Jangan mempercepat bulan empat. Seseorang yang belum pernah melakukan restore tidak boleh masuk rotasi jaga, karena pemulihan adalah keterampilan yang tidak bisa dipelajari saat dibutuhkan.

95.2 Latihan yang efektif

Latihan	Kegunaan
Merusak sesuatu di DEV lalu memperbaikinya	Membangun naluri diagnosis
Menjalankan runbook orang lain tanpa bertanya	Menguji dokumentasi dan pemahaman
Menjelaskan insiden lama ke rekan	Memantapkan pemahaman
Membaca kode produksi tanpa mengubahnya	Membangun peta mental sistem
Simulasi EOD gagal	Melatih pengambilan keputusan di bawah tekanan

Latihan pertama adalah yang paling cepat membangun kemampuan, dan yang paling jarang diberikan karena terasa seperti pemborosan. Padahal seseorang yang pernah memperbaiki sepuluh kerusakan buatan jauh lebih siap daripada yang membaca sepuluh dokumen.

95.3 Yang menahan orang bertahan

Di bidang dengan sedikit praktisi, mempertahankan orang lebih penting daripada merekrut.

Faktor	Yang bisa Anda lakukan
Merasa terjebak di teknologi lama	Beri ruang belajar SQL modern, API, open source, Git
Kelelahan jaga	Rotasi yang layak, pemulihan setelah malam panjang
Tidak ada jalur karier	Jalur teknis yang jelas, bukan hanya jalur manajerial
Pekerjaan tidak terlihat	Laporkan pencegahan, bukan hanya pemadaman
Selalu memadamkan api	Alokasikan waktu tetap untuk perbaikan struktural

Baris terakhir adalah akar dari sebagian besar yang lain. Tim yang seluruh waktunya habis menangani insiden tidak akan pernah punya waktu membangun hal yang mengurangi insiden. Memecah lingkaran itu menuntut keputusan sadar untuk melindungi sebagian waktu — dan keputusan itu ada di tangan manajemen, bukan di tangan orang yang sedang kelelahan.

💻 Bagian XXVI — DDS dan Antarmuka 5250

Layar hijau masih menjadi antarmuka utama operasi harian di sebagian besar bank. Anda tidak perlu merancangnya, tetapi harus bisa membacanya — karena setiap keluhan "layar teller error" berakhir di berkas DDS.

💻 Bab 96. Struktur DDS

96.1 Empat jenis berkas DDS

Jenis	Isi	Perintah buat
Physical file	Definisi tabel gaya lama	`CRTPF`
Logical file	View dan jalur akses gaya lama	`CRTLF`
Display file	Layar 5250	`CRTDSPF`
Printer file	Tata letak cetak	`CRTPRTF`

Untuk sistem baru, physical dan logical file sebaiknya diganti dengan `CREATE TABLE` dan `CREATE INDEX`. Display dan printer file belum ada penggantinya di lingkungan 5250.

96.2 Kolom yang menentukan

DDS adalah format kolom tetap. Posisi menentukan arti.

Posisi	Isi
1–5	Nomor urut, biasanya kosong
6	Jenis, `A` untuk semua DDS
7–16	Nama kondisi atau indikator
17	Jenis nama: `R` record, kosong untuk field
19–28	Nama record atau field
29–34	Panjang dan tipe
35–37	Desimal dan penggunaan
38–44	Posisi baris dan kolom di layar
45–80	Kata kunci

Menggeser satu kolom saja membuat kompilasi gagal atau, lebih buruk, menghasilkan perilaku yang berbeda dari yang dimaksud.

96.3 Physical file

```
     A          R REKENINGR                 TEXT('Master rekening')
     A            NO_REK        16A         COLHDG('Nomor' 'Rekening')
     A            NAMA          60A         COLHDG('Nama Nasabah')
     A            KD_PRODUK      4A         COLHDG('Kode' 'Produk')
     A            SALDO         17P 2       COLHDG('Saldo')
     A            STATUS         1A         COLHDG('Status')
     A            TGL_BUKA        L         COLHDG('Tanggal' 'Buka')
     A          K NO_REK
     A                                       UNIQUE
```

Kode tipe	Arti
`A`	Alfanumerik
`P`	Packed decimal
`S`	Zoned decimal
`B`	Binary
`L`	Date
`T`	Time
`Z`	Timestamp

Kode tipe	Arti
F	Floating point

Padanan SQL-nya:

```
CREATE TABLE DTALIB.REKENING (
    NO_REK    CHAR(16)      NOT NULL,
    NAMA      CHAR(60)      NOT NULL,
    KD_PRODUK CHAR(4)       NOT NULL,
    SALDO     DECIMAL(17,2) NOT NULL,
    STATUS    CHAR(1)       NOT NULL,
    TGL_BUKA  DATE          NOT NULL,
    PRIMARY KEY (NO_REK)
);
```

Perbedaan yang penting. Tabel SQL punya kemampuan yang tidak dimiliki physical file DDS: constraint, trigger, kolom identitas, temporal, dan RCAC. Bila ada kesempatan mengonversi, konversinya membuka fitur-fitur itu.

96.4 Logical file

```
     A          R REKENINGR                 PFILE(REKENING)
     A          K KD_PRODUK
     A          K NO_REK
     A            STATUS        COMP(EQ 'A')
```

Ini memberi jalur akses berdasarkan produk lalu nomor rekening, dengan hanya rekening aktif. Padanannya:

```
CREATE INDEX DTALIB.REKENING_IX_PRODUK
    ON DTALIB.REKENING (KD_PRODUK, NO_REK)
  WHERE STATUS = 'A';
```

Catatan kinerja. Logical file DDS yang dipakai sebagai tabel dalam kueri SQL bisa memaksa kueri jatuh ke mesin kueri lama. Bila Anda melihat kinerja kueri yang aneh, periksa apakah ia menunjuk logical file DDS.

Bab 97. Display File

97.1 Struktur dasar layar

```
     A                                   DSPSIZ(24 80 *DS3)
     A                                   CA03(03 'Keluar')
     A                                   CA12(12 'Batal')
     A          R LAYARUTAMA
     A                                   CF05(05 'Segarkan')
     A                                 1  2'Inquiry Rekening'
     A                                   DSPATR(HI)
     A                                 1 70DATE
     A                                   EDTCDE(Y)
     A                                 3  2'Nomor Rekening . :'
     A            INOREK        16A  B  3 22
     A                                   CHECK(RZ)
     A  40                               DSPATR(RI PC)
     A                                 5  2'Nama . . . . . . :'
     A            ONAMA         60A  O  5 22
     A                                 6  2'Saldo  . . . . . :'
     A            OSALDO        17Y 20  6 22
     A                                   EDTCDE(1)
     A                                23  2'F3=Keluar  F5=Segarkan'
     A                                   COLOR(BLU)
```

```
     A          R PESAN
     A            OPESAN        78A  O  22 2
     A                                   DSPATR(HI)
```

97.2 Penggunaan field

Kode	Arti
`I`	Input saja
`O`	Output saja
`B`	Both, bisa diisi dan menampilkan
`H`	Hidden, tidak tampil tetapi ikut dikirim
`M`	Message
`P`	Program to system field

97.3 Kata kunci yang paling sering ditemui

Kata kunci	Fungsi
`DSPATR(HI)`	Tampilan terang
`DSPATR(RI)`	Terbalik, untuk menandai kesalahan
`DSPATR(PC)`	Posisi kursor ke field ini
`DSPATR(ND)`	Tidak ditampilkan, untuk password
`COLOR(RED)`	Warna
`EDTCDE(1)`	Format angka dengan pemisah ribuan
`EDTCDE(Y)`	Format tanggal
`EDTWRD`	Format kustom
`CHECK(RZ)`	Nol di kanan untuk angka
`CHECK(ME)`	Field wajib diisi
`VALUES`	Nilai yang diizinkan
`RANGE`	Rentang yang diizinkan
`COMP`	Perbandingan
`ERRMSG`	Pesan kesalahan
`OVERLAY`	Menumpang di atas record lain
`SFL/SFLCTL`	Subfile

97.4 Subfile — daftar bergulir

Ini pola yang mendominasi aplikasi perbankan: daftar transaksi, daftar rekening, daftar nasabah hasil pencarian.

```
     A          R DAFTARSFL                  SFL
     A            STGL            L  0  8  3
     A            SKET          30A  0  8 14
     A            SDEBET        15Y 20  8 46
     A                                     EDTCDE(1)
     A            SKREDIT       15Y 20  8 63
     A                                     EDTCDE(1)
     A          R DAFTARCTL                 SFLCTL(DAFTARSFL)
     A                                     SFLSIZ(0051)
     A                                     SFLPAG(0010)
     A                                     OVERLAY
     A  50                                 SFLDSP
     A  51                                 SFLDSPCTL
     A  52                                 SFLCLR
     A  53                                 SFLEND(*MORE)
     A                                 6  3'Tanggal'
     A                                 6 14'Keterangan'
     A                                 6 46'Debet'
     A                                 6 63'Kredit'
```

Indikator	Fungsi
SFLDSP	Tampilkan isi subfile
SFLDSPCTL	Tampilkan record kontrol, yaitu judul kolom
SFLCLR	Kosongkan subfile
SFLEND	Tandai akhir daftar atau tampilkan penanda lanjut

Urutan penggunaannya di program: kosongkan dengan SFLCLR, isi record satu per satu, lalu tampilkan dengan SFLDSP dan SFLDSPCTL menyala.

```
**free
dcl-s wBaris int(5) inz(0);

// 1. Kosongkan subfile
*in50 = *off;
*in51 = *off;
*in52 = *on;
write DAFTARCTL;
*in52 = *off;

// 2. Isi
exec sql OPEN curMutasi;
dou sqlcode <> 0 or wBaris >= 500;
   exec sql FETCH curMutasi INTO :STGL, :SKET, :SDEBET, :SKREDIT;
   if sqlcode <> 0;
      leave;
   endif;
   wBaris += 1;
   write DAFTARSFL;
enddo;
exec sql CLOSE curMutasi;

// 3. Tampilkan
if wBaris > 0;
   *in50 = *on;
endif;
*in51 = *on;
*in53 = *on;
exfmt DAFTARCTL;
```

Batas yang sering menjadi masalah. SFLSIZ maksimal 9999 record. Aplikasi yang mencoba memuat seluruh

hasil pencarian ke subfile akan gagal pada pencarian yang luas. Solusinya: pemuatan bertahap saat pengguna menggulir, atau batasi hasil pencarian sejak kuerinya.

97.5 Membaca display file yang sudah ada

```
/* Melihat definisi lengkap */
DSPFD FILE(APPPGM/LAYARTELLER) TYPE(*ALL)
DSPFFD FILE(APPPGM/LAYARTELLER)

/* Mengambil kembali sumber DDS bila hilang */
RTVDDSSRC FILE(DTALIB/REKENING) +
          SRCFILE(APPSRC/QDDSSRC) +
          SRCMBR(REKENING)
```

`RTVDDSSRC` menyelamatkan situasi ketika objek ada tetapi sumbernya hilang. Ini lebih sering terjadi daripada yang diakui orang, dan layak dijalankan sebagai inventarisasi preventif.

```
-- Record format dan field dalam sebuah display file
SELECT RECORD_FORMAT_NAME, FIELD_NAME, DATA_TYPE,
       LENGTH, DECIMAL_POSITIONS, FIELD_USE,
       ROW_NUMBER, COLUMN_NUMBER
  FROM QSYS2.DISPLAY_FILE_INFO
 WHERE FILE_SCHEMA = 'APPPGM' AND FILE_NAME = 'LAYARTELLER';
```

97.6 Masalah layar yang sering dilaporkan

Keluhan	Penyebab yang lazim
Layar kosong atau kacau	`DSPSIZ` tidak cocok dengan emulator
Field tidak bisa diisi	Penggunaan `O` bukan `B` atau `I`
Angka tampil tanpa pemisah	`EDTCDE` hilang
Karakter aneh	CCSID sesi tidak cocok
Tombol fungsi tidak bekerja	`CAnn` atau `CFnn` tidak didefinisikan
Daftar berhenti di 9999	Batas `SFLSIZ`
Layar tampak beku	Program menunggu, bukan masalah layar
Pesan tidak hilang	Message subfile tidak dibersihkan

Baris ketujuh penting bagi System Engineer: layar yang tampak beku hampir selalu berarti program di belakangnya sedang menunggu sesuatu — kunci record, job lain, atau sistem eksternal. Periksa status job, bukan konfigurasi layar.

```
-- Job interaktif yang tampak beku bagi penggunanya
SELECT JOB_NAME, AUTHORIZATION_NAME, JOB_STATUS,
       FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_TYPE_FILTER => '*INTERACT'))
 WHERE JOB_STATUS IN ('LCKW','MSGW','DEQW','RUN')
   AND ELAPSED_TIME > 30
 ORDER BY ELAPSED_TIME DESC;
```

Bab 98. Printer File

98.1 Tata letak dasar

```
     A                                       REF(DTALIB/REKENING)
     A                                       CHRSIZ(1 1)
     A          R JUDUL                      SKIPB(001)
     A                                  1  1'Bank Contoh'
     A                                  1 40'LAPORAN SALDO HARIAN'
     A                                  1 70DATE
     A                                       EDTCDE(Y)
     A                                  2 70'Hal.'
     A                                  2 75PAGNBR
     A                                       EDTCDE(Z)
     A                                  4  1'Nomor Rekening'
     A                                  4 20'Nama'
     A                                  4 65'Saldo'
     A          R BARIS                      SPACEB(001)
     A            NO_REK    R                1
     A            NAMA      R               20
     A            SALDO     R               60
     A                                       EDTCDE(1)
     A          R TOTAL                      SPACEB(002)
     A                                  40'T O T A L'
     A            GTOTAL        19  2       60
     A                                       EDTCDE(1)
```

Kata kunci	Fungsi
SKIPB	Lompat ke baris sebelum mencetak
SKIPA	Lompat setelah mencetak
SPACEB	Spasi sebelum
SPACEA	Spasi setelah
PAGNBR	Nomor halaman otomatis
REF	Ambil definisi field dari tabel
OVERLAY	Cetak menumpang
CHRSIZ	Ukuran karakter

98.2 Mengendalikan keluaran dari CL

```
/* Mengarahkan keluaran tanpa mengubah program */
OVRPRTF FILE(LAPSALDO) +
        OUTQ(APPPGM/REGULATOR) +
        HOLD(*YES) +
        SAVE(*YES) +
        USRDTA('LAPORAN-HARIAN') +
        PAGESIZE(66 132) +
        LPI(6) CPI(10) +
        OVRFLW(60)

CALL PGM(APPPGM/CETAKSALDO)

DLTOVR FILE(LAPSALDO)
```

OVRPRTF adalah alat yang sangat berguna bagi System Engineer: ia mengubah tujuan, ukuran, dan perlakuan keluaran tanpa menyentuh program sama sekali. Untuk mengarahkan laporan ke antrean arsip atau mengubah ukuran kertas, ini jalan yang tidak menuntut perubahan kode.

98.3 Menemukan pemakai printer file

```
-- Program yang memakai printer file tertentu
SELECT WHPNAM AS program, WHLNAM AS library
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'LAPSALDO';
```

```
/* Atribut printer file yang berlaku */
DSPFD FILE(APPPGM/LAPSALDO) TYPE(*ATR)
DSPOVR FILE(*ALL)
```

`DSPOVR` menunjukkan override yang sedang aktif di job. Bila laporan muncul di antrean yang tidak diharapkan, ini tempat pertama untuk memeriksa.

🔒 Bagian XXVII — Respons Insiden Keamanan

Bagian ini tentang apa yang dilakukan System Engineer ketika kecurigaan mengarah ke penyalahgunaan, bukan kegagalan teknis. Perbedaannya penting: dalam insiden keamanan, tindakan Anda menjadi bukti, dan bukti yang rusak tidak bisa diperbaiki.

Bab 99. Prinsip Penanganan

99.1 Empat aturan yang mengubah segalanya

1. **Anda bukan penyelidik utama.** Peran Anda menyediakan data dan menjaga integritasnya. Keputusan investigasi ada pada Security Officer dan manajemen.
2. **Jangan merusak jejak.** Setiap perubahan yang Anda buat menghapus bukti dan bisa dipertanyakan kemudian.
3. **Catat segalanya, dengan waktu.** Kronologi tindakan Anda sendiri akan diminta.
4. **Jangan memberi tahu siapa pun di luar jalur eskalasi.** Termasuk rekan dekat, sampai manajemen menyatakan boleh.

Aturan keempat terasa tidak nyaman tetapi punya alasan praktis: bila dugaan mengarah ke orang dalam, peringatan dini akan menghilangkan bukti.

99.2 Urutan tindakan

Gambar 95

Satu-satunya alasan bertindak sebelum eskalasi adalah kerusakan yang sedang berlangsung. Dalam kasus lain, menunggu arahan lebih baik daripada bertindak cepat dan menghapus jejak.

99.3 Mengumpulkan bukti tanpa merusaknya

```
/* Simpan audit journal receiver SEKARANG, sebelum ia berpindah */
SAVOBJ OBJ(*ALL) LIB(QSYS) OBJTYPE(*JRNRCV) +
       DEV(*SAVF) SAVF(FORENSIK/AUDRCV01)

/* Ekstraksi ke tabel terpisah, di library khusus */
CRTLIB LIB(FORENSIK) TEXT('Bukti insiden, jangan diubah')
CHGOBJAUT OBJ(FORENSIK) OBJTYPE(*LIB) USER(*PUBLIC) AUT(*EXCLUDE)
```

```
-- Seluruh aktivitas satu profil dalam jendela yang diselidiki
CREATE TABLE FORENSIK.AKTIVITAS AS (
  SELECT ENTRY_TIMESTAMP, ENTRY_TYPE, USER_NAME, JOB_NAME,
         OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE,
         REMOTE_ADDRESS, ENTRY_DATA
    FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
           STARTING_TIMESTAMP => TIMESTAMP('2026-09-10 00:00:00'),
           ENDING_TIMESTAMP   => TIMESTAMP('2026-09-18 23:59:59')))
   WHERE USER_NAME = 'PROFIL_DISELIDIKI'
) WITH DATA;
```

Mengapa menyimpan receiver lebih dulu. Audit journal receiver berpindah dan bisa dihapus oleh proses pembersihan rutin. Bukti yang hilang karena pembersihan otomatis adalah kehilangan yang tidak bisa dijelaskan kepada auditor.

99.4 Pertanyaan yang biasanya diajukan

Pertanyaan	Sumber data
Kapan profil ini terakhir login, dari mana	`QSYS2.USER_INFO`, entri `PW` dan audit
Apa yang diaksesnya	Entri `ZR` dan `ZC` pada objek yang diaudit
Apakah ada data yang diambil keluar	Log ODBC, log FTP, `NETSTAT_JOB_INFO`
Apakah ada objek dibuat atau dihapus	Entri `CO` dan `DO`
Apakah wewenang pernah diubah	Entri `CP` dan `CA`
Apakah ada perubahan konfigurasi keamanan	Entri `SV` dan `AD`
Apakah ada restore objek	Entri `RO`, `RU`, `RZ`

```
-- Volume pembacaan objek sensitif per pengguna
SELECT USER_NAME, OBJECT_NAME,
       DATE(ENTRY_TIMESTAMP) AS tgl,
       COUNT(*) AS jml_akses
  FROM FORENSIK.AKTIVITAS
 WHERE ENTRY_TYPE IN ('ZR','ZC')
   AND OBJECT_NAME IN ('CIF_UTAMA','REKENING','KARTU')
 GROUP BY USER_NAME, OBJECT_NAME, DATE(ENTRY_TIMESTAMP)
 ORDER BY jml_akses DESC;

-- Aktivitas di luar jam kerja
SELECT ENTRY_TIMESTAMP, ENTRY_TYPE, USER_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, REMOTE_ADDRESS
  FROM FORENSIK.AKTIVITAS
 WHERE HOUR(ENTRY_TIMESTAMP) NOT BETWEEN 7 AND 18
    OR DAYOFWEEK(ENTRY_TIMESTAMP) IN (1, 7)
 ORDER BY ENTRY_TIMESTAMP;
```

99.5 Tanda yang layak diselidiki

Tanda	Kenapa mencurigakan
Lonjakan pembacaan tabel nasabah	Pengambilan data massal
Akses dari alamat IP baru	Kredensial dipakai orang lain
Login pada jam tidak biasa untuk profil itu	Perubahan pola perilaku
Pembuatan profil baru berwenang tinggi	Pembuatan pintu belakang
Perubahan konfigurasi audit	Upaya menutup jejak
Penghapusan journal receiver	Idem
Program baru di library produksi tanpa tiket	Kode yang tidak melalui kendali

Tanda	Kenapa mencurigakan
Koneksi ODBC dengan kueri sangat besar	Eksfiltrasi
Berkas besar baru di IFS	Persiapan pemindahan data

```
-- Objek baru di library produksi dalam 7 hari terakhir
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJCREATED, OBJTEXT
  FROM TABLE(QSYS2.OBJECT_STATISTICS('APPPGM','*ALL'))
 WHERE OBJCREATED > CURRENT TIMESTAMP - 7 DAYS
 ORDER BY OBJCREATED DESC;

-- Berkas besar baru di IFS
SELECT PATH_NAME, DATA_SIZE / 1048576 AS mb,
       OBJECT_OWNER, CREATE_TIMESTAMP
  FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
         START_PATH_NAME => '/',
         SUBTREE_DIRECTORIES => 'YES'))
 WHERE CREATE_TIMESTAMP > CURRENT TIMESTAMP - 7 DAYS
   AND DATA_SIZE > 104857600
 ORDER BY DATA_SIZE DESC;
```

💾 Bab 100. Penahanan dan Pemulihan

100.1 Pilihan penahanan

Tindakan	Dampak	Kapan tepat
Nonaktifkan profil	Menghentikan akses, terlihat pelaku	Ada bukti kuat
Blokir alamat IP di exit program	Tidak terlihat pelaku	Sedang dikumpulkan bukti
Akhiri sesi aktif	Menghentikan aktivitas berjalan	Eksfiltrasi sedang terjadi
Matikan layanan tertentu	Menghentikan seluruh jalur	Jalur itu jelas disalahgunakan
Isolasi jaringan	Memutus semuanya	Kompromi meluas

```
/* Nonaktifkan profil, hanya atas arahan */
CHGUSRPRF USRPRF(PROFIL_TERSANGKA) STATUS(*DISABLED)

/* Akhiri sesi yang sedang berjalan */
ENDJOB JOB(123456/PROFIL/QPADEV0001) OPTION(*IMMED)

/* Tingkatkan audit pada profil tertentu tanpa memblokir */
CHGUSRAUD USRPRF(PROFIL_DIPANTAU) +
          AUDLVL(*CMD *CREATE *DELETE *OBJMGT *SAVRST *SECURITY)
```

`CHGUSRAUD` adalah pilihan yang sering paling tepat di tahap awal: aktivitas tercatat lengkap, pelaku tidak tahu sedang diamati, dan operasi tidak terganggu.

100.2 Setelah kompromi kredensial

Urutan yang harus dijalankan:

1. Nonaktifkan profil yang terkompromi
2. Identifikasi seluruh tempat kredensial itu dipakai — skrip, aplikasi, penjadwal
3. Ganti kredensial di semua tempat itu secara terkoordinasi
4. Periksa apakah profil membuat profil lain
5. Periksa apakah ada exit program atau program yang diubah
6. Periksa apakah ada objek baru dengan wewenang adopsi

7. Tinjau seluruh akses yang dilakukan profil itu sejak kompromi diperkirakan terjadi

```
-- Profil yang dibuat oleh profil tersangka
SELECT ENTRY_TIMESTAMP, USER_NAME AS pembuat, OBJECT_NAME AS profil_baru
  FROM FORENSIK.AKTIVITAS
 WHERE ENTRY_TYPE = 'CO'
   AND OBJECT_TYPE = '*USRPRF';

-- Program yang mengadopsi wewenang, dibuat baru-baru ini
SELECT PROGRAM_LIBRARY, PROGRAM_NAME, PROGRAM_OWNER,
       USER_PROFILE, CREATION_TIMESTAMP
  FROM QSYS2.PROGRAM_INFO
 WHERE USER_PROFILE = '*OWNER'
   AND CREATION_TIMESTAMP > CURRENT TIMESTAMP - 30 DAYS
 ORDER BY CREATION_TIMESTAMP DESC;
```

Kueri kedua mencari pintu belakang klasik: program kecil yang dimiliki profil berwenang tinggi dan mengadopsi wewenangnya, sehingga siapa pun yang memanggilnya memperoleh wewenang itu sesaat.

100.3 Memverifikasi integritas sistem

```
/* Periksa tanda tangan objek sistem */
CHKOBJITG USRPRF(*ALL) CHKDMN(*YES) CHKPGMSIG(*ALL) +
          OUTFILE(FORENSIK/INTEGRITAS)
```

```
SELECT * FROM FORENSIK.INTEGRITAS;
```

`CHKOBJITG` memeriksa apakah ada objek program yang dimodifikasi tanpa melalui kompilasi yang sah, objek yang melanggar batas domain, atau tanda tangan digital yang tidak cocok. Ia memakan waktu lama pada sistem besar, tetapi merupakan pemeriksaan integritas paling menyeluruh yang tersedia.

100.4 Laporan insiden

```
LAPORAN INSIDEN KEAMANAN [nomor]

1. RINGKASAN
   Apa yang terjadi, dalam tiga kalimat.

2. KRONOLOGI
   Waktu        Peristiwa                      Sumber bukti
   ---------------------------------------------------------

3. CAKUPAN
   Sistem terdampak:
   Data yang mungkin terpapar:
   Jumlah subjek data terdampak:
   Periode paparan:

4. AKAR MASALAH
   Bagaimana ini bisa terjadi.
   Kontrol apa yang gagal atau tidak ada.

5. TINDAKAN PENAHANAN
   Yang dilakukan, oleh siapa, pukul berapa.

6. PEMULIHAN

7. PELAPORAN EKSTERNAL
```

```
   Kewajiban lapor ke regulator: ya atau tidak, dasar hukumnya.
   Kewajiban memberitahu subjek data: ya atau tidak.
   Tanggal pelaporan.

8. PERBAIKAN
   Tindakan          Penanggung jawab   Tenggat
   ---------------------------------------------

9. LAMPIRAN BUKTI
```

Bagian tujuh menuntut keterlibatan tim hukum dan kepatuhan. Pelaporan insiden yang melibatkan data pribadi punya tenggat yang diatur, dan keterlambatan pelaporan adalah pelanggaran tersendiri, terpisah dari insidennya.

100.5 Setelah insiden

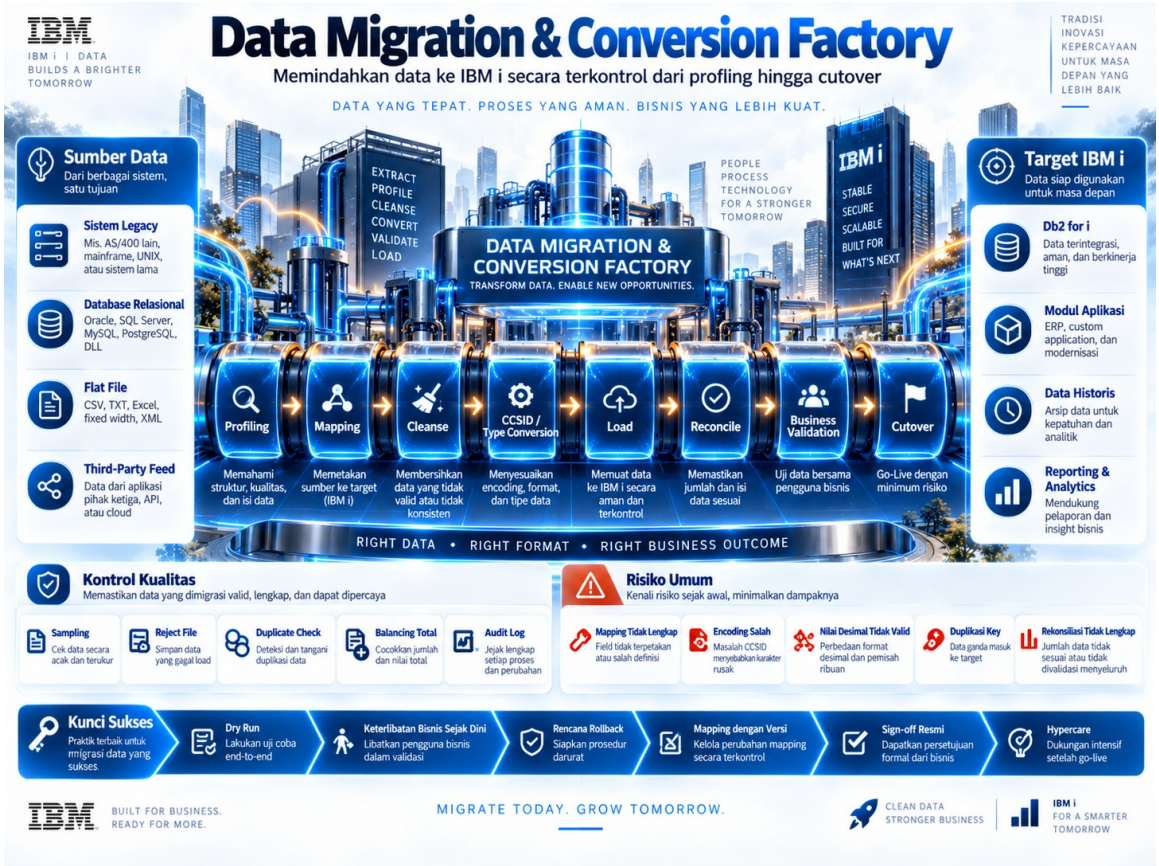
Tindakan	Alasan
Tambahkan deteksi yang akan menangkapnya lebih awal	Insiden serupa akan terulang
Tinjau apakah pola yang sama ada di tempat lain	Satu kelemahan jarang berdiri sendiri
Perbarui runbook respons	Pengalaman ini akan terlupakan
Latih tim dengan skenario ini	Pengetahuan harus menyebar
Tinjau ulang matriks wewenang	Sering insiden mengungkap wewenang berlebih

```
-- Deteksi berkelanjutan: lonjakan akses tabel sensitif
CREATE OR REPLACE VIEW SECLIB.V_DETEKSI_AKSES AS
WITH harian AS (
  SELECT USER_NAME, DATE(ENTRY_TIMESTAMP) AS tgl, COUNT(*) AS jml
    FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
           JOURNAL_ENTRY_TYPES => 'ZR',
           STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS))
   WHERE OBJECT_NAME IN ('CIF_UTAMA','KARTU')
   GROUP BY USER_NAME, DATE(ENTRY_TIMESTAMP)
),
baseline AS (
  SELECT USER_NAME, AVG(jml) AS rata, MAX(jml) AS puncak
    FROM harian
   WHERE tgl < CURRENT DATE
   GROUP BY USER_NAME
)
SELECT h.USER_NAME, h.tgl, h.jml,
       DECIMAL(b.rata, 9, 1) AS rata_biasa,
       DECIMAL(h.jml / NULLIF(b.rata, 0), 9, 1) AS kali_lipat
  FROM harian h JOIN baseline b ON h.USER_NAME = b.USER_NAME
 WHERE h.tgl = CURRENT DATE
   AND h.jml > b.rata * 3
   AND h.jml > 100;
```

Deteksi berbasis penyimpangan dari kebiasaan sendiri jauh lebih efektif daripada ambang tetap. Petugas layanan nasabah yang biasa membuka lima puluh data per hari lalu tiba-tiba membuka dua ribu adalah sinyal yang jelas, sementara ambang tetap seribu akan melewatkannya atau membanjiri Anda dengan peringatan palsu.

📦 Bagian XXVIII — Migrasi dan Konversi Data

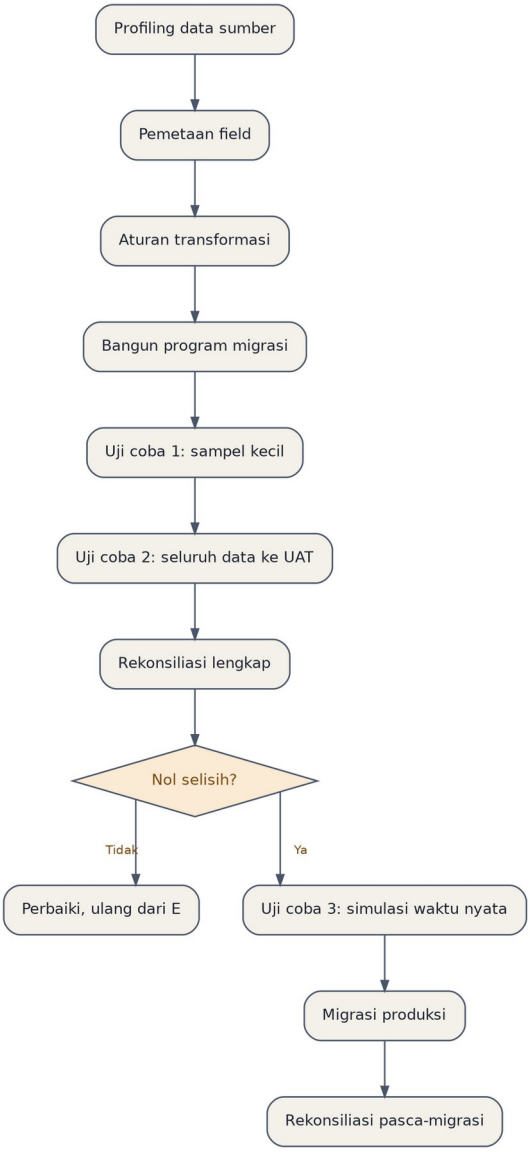
📦 Bab 101. Merancang Migrasi Data



Gambar 96 — Alur migrasi dan konversi data ke IBM i

Migrasi data adalah proyek dengan tingkat kegagalan tertinggi dalam pekerjaan ini, karena kesalahan kecil terlipat jutaan kali dan sering baru terlihat berminggu-minggu kemudian.

101.1 Tahapan



Gambar 97

Tiga uji coba bukan berlebihan. Uji pertama menangkap kesalahan logika, kedua menangkap kasus data pinggiran, ketiga menangkap masalah waktu dan kapasitas.

101.2 Profiling data sumber

Dilakukan sebelum menulis satu baris kode migrasi.

```
-- Profil satu kolom: nilai unik, kosong, ekstrem
SELECT COUNT(*) AS total,
       COUNT(DISTINCT KD_PRODUK) AS nilai_unik,
       SUM(CASE WHEN KD_PRODUK IS NULL THEN 1 ELSE 0 END) AS jml_null,
       SUM(CASE WHEN TRIM(KD_PRODUK) = '' THEN 1 ELSE 0 END) AS kosong,
       MIN(LENGTH(TRIM(KD_PRODUK))) AS panjang_min,
       MAX(LENGTH(TRIM(KD_PRODUK))) AS panjang_maks
  FROM SUMBER.REKENING_LAMA;
```

```
-- Distribusi nilai, untuk menemukan yang tidak terduga
SELECT KD_PRODUK, COUNT(*) AS jml
  FROM SUMBER.REKENING_LAMA
 GROUP BY KD_PRODUK
 ORDER BY jml DESC;

-- Nilai yang tidak ada di tabel referensi
SELECT DISTINCT s.KD_PRODUK
  FROM SUMBER.REKENING_LAMA s
 WHERE NOT EXISTS (SELECT 1 FROM APPPRM.PRODUK p
                   WHERE p.KD_PRODUK = s.KD_PRODUK);

-- Tanggal di luar rentang wajar
SELECT MIN(TGL_BUKA) AS terlama, MAX(TGL_BUKA) AS terbaru,
       SUM(CASE WHEN TGL_BUKA < DATE('1960-01-01')
                THEN 1 ELSE 0 END) AS terlalu_lama,
       SUM(CASE WHEN TGL_BUKA > CURRENT DATE
                THEN 1 ELSE 0 END) AS masa_depan
  FROM SUMBER.REKENING_LAMA;
```

Yang selalu ditemukan dan selalu mengejutkan:

Temuan	Contoh
Kode yang tidak ada di tabel referensi	Produk yang sudah dihapus tetapi rekeningnya masih ada
Tanggal mustahil	Tanggal lahir tahun 1900 atau 2099
Field yang dipakai untuk dua hal	Kolom catatan berisi kode tersembunyi
Spasi versus null	Dua cara menyatakan kosong
Nilai yang dipadatkan	Beberapa informasi dalam satu kolom
Data duplikat	Nasabah yang sama berkali-kali
Karakter tak terduga	Tab, baris baru, karakter kontrol di kolom teks

Baris ketiga adalah yang paling berbahaya. Kolom keterangan yang secara tidak resmi dipakai menyimpan penanda tertentu akan hilang maknanya setelah migrasi, dan proses hilir yang bergantung padanya akan gagal diam-diam.

101.3 Dokumen pemetaan

```
PEMETAAN: SUMBER.REKENING_LAMA -> DTALIB.REKENING

Kolom sumber  Tipe      Kolom tujuan  Tipe       Transformasi
---------------------------------------------------------------
REKNO         CHAR 12   NO_REK        CHAR 16    Padding nol di kiri
NAMALENGKAP   CHAR 40   NAMA          VARCHAR 60 Trim, upper, bersihkan
KDPRD         CHAR 3    KD_PRODUK     CHAR 4     Tabel konversi K01
SALDOAKHIR    PACK 15,2 SALDO         DEC 17,2   Langsung
FLAG          CHAR 1    STATUS        CHAR 1     A=A T=C B=B lain=X
TGLBUKA       NUM 8     TGL_BUKA      DATE       YYYYMMDD, 0 jadi NULL
(tidak ada)   -         KD_CABANG     CHAR 4     Tabel silang cabang
CATATAN       CHAR 50   (tidak dipakai) -        PERIKSA: berisi kode?

BARIS YANG DIKECUALIKAN
  - STATUS = X setelah konversi: laporkan, jangan migrasikan
  - SALDO negatif pada produk deposito: laporkan untuk ditinjau
```

Baris terakhir pada tabel pemetaan — kolom yang tidak dipakai tetapi ditandai untuk diperiksa — adalah kebiasaan yang menyelamatkan proyek. Jangan pernah membuang kolom tanpa memastikan isinya benar-benar tidak bermakna.

101.4 Program migrasi yang bisa diulang

```
**free
ctl-opt dftactgrp(*no) actgrp('MIGGRP') option(*srcstmt);

dcl-s wBatch  int(10);
dcl-s wProses int(10) inz(0);
dcl-s wLolos  int(10) inz(0);
dcl-s wTolak  int(10) inz(0);

// Idempoten: tolak bila batch ini sudah selesai
exec sql
  SELECT COUNT(*) INTO :wBatch
    FROM MIGRASI.KONTROL
   WHERE KD_BATCH = :pKdBatch AND STATUS = 'S';

if wBatch > 0;
   kirimEscape('Batch ini sudah dimigrasikan');
endif;

exec sql
  INSERT INTO MIGRASI.KONTROL (KD_BATCH, MULAI, STATUS)
  VALUES (:pKdBatch, CURRENT TIMESTAMP, 'R');
exec sql COMMIT;

exec sql OPEN curSumber;

dou sqlcode <> 0;
   exec sql FETCH curSumber INTO :dsSumber;
   if sqlcode <> 0;
      leave;
   endif;

   wProses += 1;
   clear dsTujuan;

   monitor;
      transformasi(dsSumber: dsTujuan);
      validasi(dsTujuan);

      exec sql INSERT INTO DTALIB.REKENING VALUES(:dsTujuan);

      // Jejak: kunci lama jadi kunci baru
      exec sql
        INSERT INTO MIGRASI.JEJAK
          (KD_BATCH, KUNCI_SUMBER, KUNCI_TUJUAN, WAKTU)
        VALUES (:pKdBatch, :dsSumber.reknoLama,
                :dsTujuan.noRek, CURRENT TIMESTAMP);

      wLolos += 1;
   on-error *all;
      wTolak += 1;
      exec sql
        INSERT INTO MIGRASI.TOLAK
          (KD_BATCH, KUNCI_SUMBER, ALASAN)
        VALUES (:pKdBatch, :dsSumber.reknoLama, :wAlasan);
   endmon;

   if %rem(wProses: 5000) = 0;
      exec sql COMMIT;
   endif;
```

```
enddo;

exec sql CLOSE curSumber;
exec sql COMMIT;

exec sql
  UPDATE MIGRASI.KONTROL
     SET STATUS = 'S', SELESAI = CURRENT TIMESTAMP,
         JML_PROSES = :wProses, JML_LOLOS = :wLolos,
         JML_TOLAK = :wTolak
   WHERE KD_BATCH = :pKdBatch;
exec sql COMMIT;

*inlr = *on;
```

Tiga unsur yang membuat program ini layak produksi: penanda idempotensi, tabel jejak yang memetakan kunci lama ke kunci baru, dan tabel penolakan agar data bermasalah bisa diperbaiki lalu dimigrasikan ulang.

Tabel jejak sangat penting. Ketika enam bulan kemudian seseorang bertanya rekening lama nomor sekian sekarang menjadi nomor berapa, tanpa tabel ini jawabannya tidak ada.

101.5 Rekonsiliasi

Ini bagian yang menentukan apakah migrasi berhasil, dan tidak boleh diringkas.

```
-- 1. Jumlah record
SELECT (SELECT COUNT(*) FROM SUMBER.REKENING_LAMA) AS sumber,
       (SELECT COUNT(*) FROM DTALIB.REKENING)      AS tujuan,
       (SELECT COUNT(*) FROM MIGRASI.TOLAK)        AS ditolak
  FROM SYSIBM.SYSDUMMY1;

-- 2. Total nilai keuangan, harus persis sama
SELECT (SELECT SUM(SALDOAKHIR) FROM SUMBER.REKENING_LAMA
          WHERE REKNO NOT IN (SELECT KUNCI_SUMBER FROM MIGRASI.TOLAK))
         AS total_sumber,
       (SELECT SUM(SALDO) FROM DTALIB.REKENING) AS total_tujuan
  FROM SYSIBM.SYSDUMMY1;

-- 3. Distribusi per kategori, harus sebanding
SELECT 'sumber' AS asal, KDPRD AS kode, COUNT(*) AS jml,
       SUM(SALDOAKHIR) AS total
  FROM SUMBER.REKENING_LAMA GROUP BY KDPRD
UNION ALL
SELECT 'tujuan', KD_PRODUK, COUNT(*), SUM(SALDO)
  FROM DTALIB.REKENING GROUP BY KD_PRODUK
 ORDER BY kode, asal;

-- 4. Uji petik record demi record
SELECT s.REKNO, s.SALDOAKHIR, t.NO_REK, t.SALDO
  FROM SUMBER.REKENING_LAMA s
  JOIN MIGRASI.JEJAK j ON j.KUNCI_SUMBER = s.REKNO
  JOIN DTALIB.REKENING t ON t.NO_REK = j.KUNCI_TUJUAN
 WHERE s.SALDOAKHIR <> t.SALDO;
```

Toleransi untuk nilai keuangan adalah nol. Selisih satu rupiah pada jutaan record berarti ada kesalahan pembulatan sistematis yang akan terus berlipat. Jangan pernah menerima alasan bahwa selisihnya kecil.

101.6 Menangani data yang ditolak

```
SELECT ALASAN, COUNT(*) AS jml
  FROM MIGRASI.TOLAK
 GROUP BY ALASAN
 ORDER BY jml DESC;
```

Jenis penolakan	Penanganan
Kode referensi tidak dikenal	Tambahkan ke tabel konversi, migrasikan ulang
Data wajib kosong	Keputusan bisnis: isi default atau kecualikan
Nilai di luar rentang	Tinjau satu per satu, sering data rusak
Duplikat	Keputusan bisnis: gabungkan atau pilih satu
Kesalahan format tanggal	Perbaiki aturan transformasi

Setiap kategori penolakan menuntut keputusan bisnis yang tercatat, bukan keputusan teknis yang diambil pengembang saat larut malam. Bawa daftar ini ke pemilik proses.

🗄️ Bab 102. Konversi Objek Basis Data

102.1 Menghasilkan DDL dari objek yang ada

```
-- DDL untuk satu tabel
CALL QSYS2.GENERATE_SQL(
  DATABASE_OBJECT_NAME => 'REKENING',
  DATABASE_OBJECT_LIBRARY_NAME => 'DTALIB',
  DATABASE_OBJECT_TYPE => 'TABLE',
  CREATE_OR_REPLACE_OPTION => '1',
  SYSTEM_NAMING_OPTION => '0');

-- DDL untuk banyak objek sekaligus
CALL QSYS2.GENERATE_SQL_OBJECTS(
  SYSTEM_TABLE_NAME => 'OBJLIST',
  SYSTEM_TABLE_SCHEMA => 'QTEMP',
  CREATE_OR_REPLACE_OPTION => '1');
```

Ini cara tercepat mendapatkan definisi tabel dalam bentuk SQL untuk dimasukkan ke kendali versi, termasuk untuk tabel yang aslinya dibuat dengan DDS.

102.2 Mengubah struktur tabel besar

Perubahan	Dampak	Pendekatan
Tambah kolom nullable	Cepat, metadata saja	Langsung
Tambah kolom dengan default	Menulis ulang seluruh tabel	Jendela pemeliharaan
Perbesar panjang kolom	Menulis ulang	Jendela pemeliharaan
Perkecil panjang kolom	Menulis ulang, risiko pemotongan	Hindari
Ubah tipe data	Menulis ulang, risiko konversi	Tabel bayangan
Tambah constraint	Validasi seluruh baris	Jendela, atau check pending

```
-- Pola tabel bayangan untuk perubahan besar
CREATE TABLE DTALIB.REKENING_BARU LIKE DTALIB.REKENING;

ALTER TABLE DTALIB.REKENING_BARU
  ALTER COLUMN NAMA SET DATA TYPE VARCHAR(100);

INSERT INTO DTALIB.REKENING_BARU SELECT * FROM DTALIB.REKENING;

-- Verifikasi sebelum menukar
SELECT (SELECT COUNT(*) FROM DTALIB.REKENING) AS jml_lama,
       (SELECT COUNT(*) FROM DTALIB.REKENING_BARU) AS jml_baru,
       (SELECT SUM(SALDO) FROM DTALIB.REKENING) AS saldo_lama,
       (SELECT SUM(SALDO) FROM DTALIB.REKENING_BARU) AS saldo_baru
  FROM SYSIBM.SYSDUMMY1;
```

```
/* Tukar dalam jendela pendek */
RNMOBJ OBJ(DTALIB/REKENING) OBJTYPE(*FILE) NEWOBJ(REKENINGOLD)
RNMOBJ OBJ(DTALIB/REKENING_BARU) OBJTYPE(*FILE) NEWOBJ(REKENING)

/* Mulai journaling pada tabel baru, JANGAN SAMPAI TERLEWAT */
STRJRNPF FILE(DTALIB/REKENING) JRN(JRNLIB/JRNDTA) IMAGES(*BOTH)
```

Kesalahan paling sering pada pola tabel bayangan: lupa memulai journaling pada tabel baru. Tabel pengganti tidak mewarisi konfigurasi journaling, dan sistem berjalan tanpa jurnal sampai seseorang menyadarinya — biasanya saat jurnal itu dibutuhkan untuk pemulihan.

Daftar periksa setelah penukaran tabel:

- ☐ Journaling aktif pada tabel baru
- ☐ Seluruh index dibuat ulang
- ☐ Constraint dan trigger dipasang kembali
- ☐ Wewenang objek sama dengan tabel lama
- ☐ RCAC dan mask diaktifkan kembali
- ☐ Program yang memakainya dikompilasi ulang bila format berubah
- ☐ Tabel lama disimpan sampai verifikasi selesai

102.3 Memindahkan data antar sistem

```
/* Daftarkan basis data jarak jauh */
ADDRDBDIRE RDB(SISTEMDR) RMTLOCNAME('10.20.31.50' *IP)
```

```
-- Penyalinan lewat koneksi jarak jauh
CONNECT TO SISTEMDR;
CREATE TABLE QTEMP.SALINAN AS
  (SELECT * FROM DTALIB.REKENING) WITH DATA;
CONNECT RESET;
```

Untuk volume besar, save file lalu pemindahan berkas biasanya lebih cepat dan lebih mudah dipantau daripada penyalinan lewat SQL.

```
/* Pola save file untuk pemindahan besar */
CRTSAVF FILE(QTEMP/PINDAH)
SAVOBJ OBJ(REKENING) LIB(DTALIB) DEV(*SAVF) SAVF(QTEMP/PINDAH)

/* Di sistem tujuan, setelah berkas dipindahkan */
RSTOBJ OBJ(REKENING) SAVLIB(DTALIB) DEV(*SAVF) +
       SAVF(QTEMP/PINDAH) RSTLIB(DTALIB)
```

Setelah restore, periksa kembali journaling, wewenang, dan index — ketiganya adalah hal yang paling sering tertinggal dalam pemindahan antar sistem.

Bagian XXIX — Integrasi Berbasis Pesan

Bab 103. Data Queue

Data queue adalah mekanisme antrean bawaan IBM i. Ringan, cepat, dan sering cukup untuk koordinasi di dalam satu sistem.

103.1 Dua jenis

Jenis	Perilaku	Kegunaan
Standard	FIFO, LIFO, atau berdasarkan kunci	Koordinasi antar job
DDM	Menunjuk data queue di sistem lain	Komunikasi antar partisi

```
/* FIFO sederhana */
CRTDTAQ DTAQ(APPWRK/EODQ) MAXLEN(256) SEQ(*FIFO) +
        TEXT('Sinyal antar fase EOD')

/* Berdasarkan kunci, untuk mengambil pesan tertentu */
CRTDTAQ DTAQ(APPWRK/TRXQ) MAXLEN(512) SEQ(*KEYED) +
        KEYLEN(16) TEXT('Antrean transaksi berkunci')

/* Yang tetap ada setelah IPL */
CRTDTAQ DTAQ(APPWRK/AUDITQ) MAXLEN(1024) SEQ(*FIFO) +
        FORCE(*YES) TEXT('Antrean audit, tahan IPL')
```

`FORCE(*YES)` menulis isi antrean ke penyimpanan permanen. Tanpa itu, isi antrean hilang saat sistem berhenti tidak normal — yang bisa berarti transaksi hilang.

103.2 Mengirim dan menerima

```
**free
dcl-pr sndDtaQ extpgm('QSNDDTAQ');
  pNamaQ   char(10) const;
  pLibQ    char(10) const;
  pPanjang packed(5:0) const;
  pData    char(512) const;
end-pr;

dcl-pr rcvDtaQ extpgm('QRCVDTAQ');
  pNamaQ   char(10) const;
  pLibQ    char(10) const;
  pPanjang packed(5:0);
  pData    char(512);
  pTunggu  packed(5:0) const;
end-pr;
```

```
// Mengirim
sndDtaQ('EODQ': 'APPWRK': 256: wPesan);

// Menerima dengan batas waktu 300 detik
dcl-s wPanjang packed(5:0);
wPanjang = 512;
rcvDtaQ('EODQ': 'APPWRK': wPanjang: wPesan: 300);

if wPanjang = 0;
   // timeout: tidak ada pesan dalam 300 detik
   kirimEscape('Timeout menunggu sinyal dari fase sebelumnya');
endif;
```

Nilai pTunggu	Perilaku
`0`	Ambil bila ada, langsung kembali bila kosong
Angka positif	Tunggu sekian detik
`-1`	Tunggu selamanya

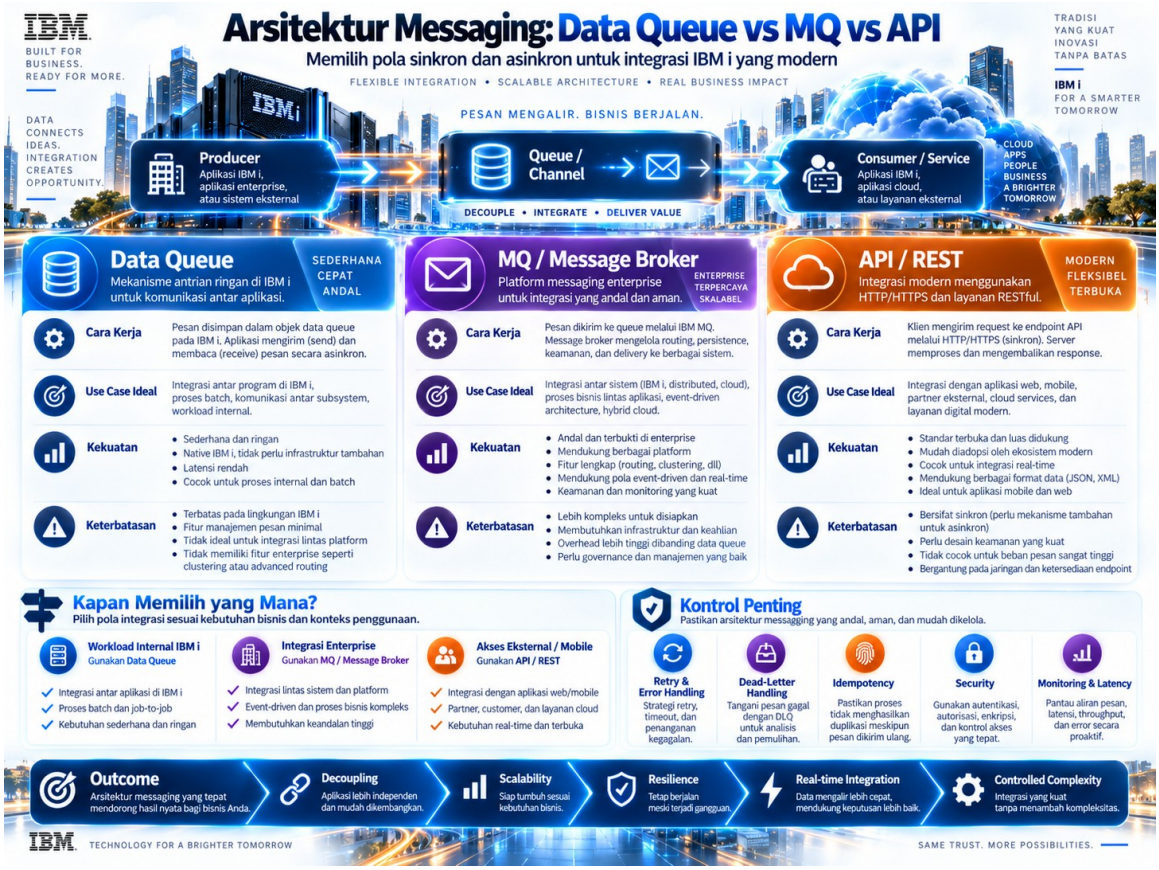
Nilai `-1` tidak boleh ada di program batch produksi. Inilah penyebab kasus 7 di Bab 91: job menggantung empat jam tanpa pesan karena menunggu sinyal dari job yang sudah mati.

103.3 Memantau data queue

```
SELECT DATA_QUEUE_LIBRARY, DATA_QUEUE_NAME,
       DATA_QUEUE_TYPE, MAXIMUM_MESSAGE_LENGTH,
       CURRENT_MESSAGES, MAXIMUM_MESSAGES,
       FORCE, TEXT_DESCRIPTION
  FROM QSYS2.DATA_QUEUE_INFO
 WHERE DATA_QUEUE_LIBRARY = 'APPWRK'
 ORDER BY CURRENT_MESSAGES DESC;

-- Mengintip isi tanpa mengambilnya
SELECT ORDINAL_POSITION, MESSAGE_DATA, MESSAGE_ENQUEUE_TIMESTAMP
  FROM TABLE(QSYS2.DATA_QUEUE_ENTRIES(
         DATA_QUEUE => 'EODQ',
         DATA_QUEUE_LIBRARY => 'APPWRK'))
 ORDER BY ORDINAL_POSITION
 FETCH FIRST 20 ROWS ONLY;
```

Antrean yang terus bertambah adalah tanda bahwa pengirim lebih cepat daripada penerima, atau penerima sudah mati. Pantau `CURRENT_MESSAGES` sebagai metrik rutin.

Bab 104. MQ dan Middleware



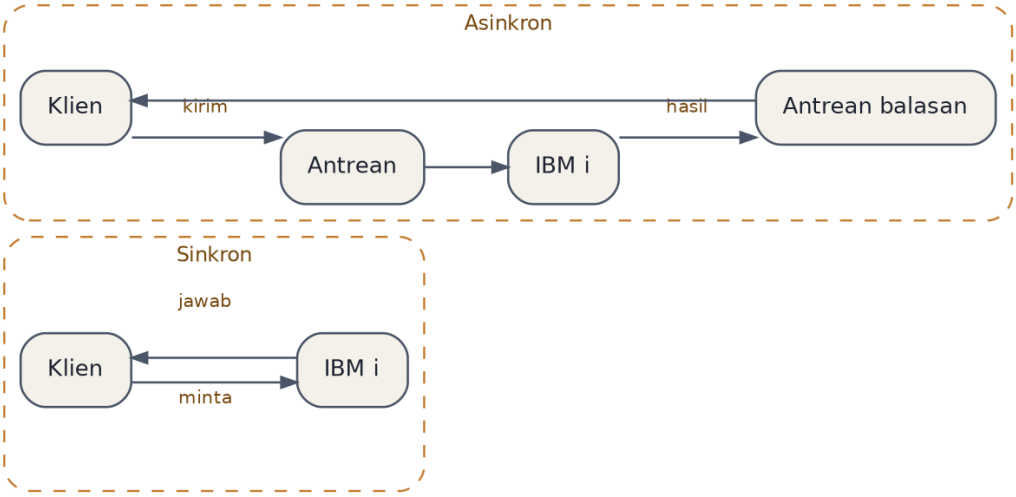
Gambar 98 — Memilih antara data queue, MQ, dan API

104.1 Kapan MQ layak

Kebutuhan	Data queue	MQ
Antar job di satu sistem	Cukup	Berlebihan
Antar partisi IBM i	DDM data queue	Bisa
Antar platform berbeda	Tidak	Ya
Jaminan pengiriman transaksional	Tidak	Ya
Persistensi setelah kegagalan	Terbatas	Ya
Pemantauan dan alat bantu	Minim	Lengkap

Untuk pertukaran dengan sistem non-IBM i yang menuntut jaminan pengiriman — pembayaran antar bank, misalnya — MQ atau padanannya adalah pilihan yang tepat. Untuk koordinasi internal, data queue lebih sederhana dan lebih cepat.

104.2 Pola integrasi yang perlu dipahami



Gambar 99

Pola	Cocok untuk	Risiko
Permintaan dan jawaban sinkron	Inquiry saldo, otorisasi	Timeout memblokir klien
Kirim dan lupakan	Notifikasi, log	Kehilangan tanpa disadari
Permintaan dengan balasan asinkron	Transfer, kliring	Korelasi pesan rumit
Publikasi dan langganan	Peristiwa ke banyak konsumen	Urutan tidak terjamin
Batch berkas	Rekonsiliasi, pelaporan	Latensi tinggi

104.3 Korelasi pesan

Dalam pola asinkron, setiap pesan harus membawa pengenal yang memungkinkan jawabannya dicocokkan dengan permintaannya.

```
CREATE TABLE DTALIB.PESAN_KORELASI (
    KORELASI_ID  VARCHAR(64)  NOT NULL,
    JENIS        VARCHAR(20)  NOT NULL,
    WAKTU_KIRIM  TIMESTAMP    NOT NULL,
    WAKTU_TERIMA TIMESTAMP,
    STATUS       CHAR(1)      NOT NULL,
    NO_REF       VARCHAR(40),
    PRIMARY KEY (KORELASI_ID)
);

CREATE INDEX DTALIB.KORELASI_IX1
    ON DTALIB.PESAN_KORELASI (STATUS, WAKTU_KIRIM);
```

```
-- Pesan yang dikirim tetapi belum berbalas
SELECT JENIS, COUNT(*) AS jml,
       MIN(WAKTU_KIRIM) AS tertua,
       TIMESTAMPDIFF(4, CHAR(CURRENT TIMESTAMP - MIN(WAKTU_KIRIM)))
         AS menit_tertua
  FROM DTALIB.PESAN_KORELASI
 WHERE STATUS = 'K'
   AND WAKTU_KIRIM < CURRENT TIMESTAMP - 5 MINUTES
 GROUP BY JENIS
 ORDER BY menit_tertua DESC;
```

Kueri ini adalah pemantauan terpenting untuk integrasi asinkron. Pesan yang tidak pernah berbalas adalah transaksi yang menggantung, dan di perbankan itu berarti dana nasabah dalam keadaan tidak jelas.

104.4 Ketahanan

Mekanisme	Fungsi
Batas waktu pada setiap panggilan keluar	Mencegah job menggantung
Percobaan ulang bertingkat	1 detik, 5 detik, 30 detik, lalu berhenti
Circuit breaker	Berhenti mencoba saat lawan jelas mati
Antrean surat mati	Pesan yang gagal berulang tidak hilang
Idempotency key	Percobaan ulang tidak menghasilkan dobel
Rekonsiliasi terjadwal	Menangkap yang lolos dari semuanya

```
**free
// Circuit breaker sederhana
dcl-s wGagalBeruntun int(5);
dcl-s wStatusSirkuit char(1);

exec sql
  SELECT GAGAL_BERUNTUN, STATUS INTO :wGagalBeruntun, :wStatusSirkuit
    FROM APPWRK.SIRKUIT WHERE KD_MITRA = :pKdMitra;

if wStatusSirkuit = 'O';   // terbuka: jangan coba
   pResp.kodeHasil = 'E503';
   pResp.pesan = 'Layanan mitra sedang tidak tersedia';
   return;
endif;

monitor;
   panggilMitra(pKdMitra: pData: pResp);
   // berhasil: setel ulang penghitung
   exec sql
     UPDATE APPWRK.SIRKUIT
        SET GAGAL_BERUNTUN = 0, STATUS = 'T'
      WHERE KD_MITRA = :pKdMitra;
on-error *all;
   wGagalBeruntun += 1;
   exec sql
     UPDATE APPWRK.SIRKUIT
        SET GAGAL_BERUNTUN = :wGagalBeruntun,
            STATUS = CASE WHEN :wGagalBeruntun >= 5 THEN 'O' ELSE 'T' END,
            WAKTU_BUKA = CASE WHEN :wGagalBeruntun >= 5
                              THEN CURRENT TIMESTAMP ELSE WAKTU_BUKA END
      WHERE KD_MITRA = :pKdMitra;
   pResp.kodeHasil = 'E503';
endmon;
```

Sirkuit yang terbuka harus punya mekanisme menutup kembali — biasanya setelah jeda tertentu, satu permintaan percobaan dikirim untuk menguji apakah mitra sudah pulih.

104.5 Memantau kesehatan integrasi

```
CREATE OR REPLACE VIEW APPWRK.V_KESEHATAN_INTEGRASI AS
SELECT KD_MITRA,
       COUNT(*) AS total_1jam,
       SUM(CASE WHEN STATUS = 'S' THEN 1 ELSE 0 END) AS berhasil,
       SUM(CASE WHEN STATUS = 'G' THEN 1 ELSE 0 END) AS gagal,
       SUM(CASE WHEN STATUS = 'K' THEN 1 ELSE 0 END) AS menggantung,
       DECIMAL(AVG(DURASI_MS), 9, 1) AS rata_ms,
       MAX(DURASI_MS) AS terlama_ms,
       CASE
         WHEN SUM(CASE WHEN STATUS = 'G' THEN 1 ELSE 0 END) * 100.0
              / NULLIF(COUNT(*), 0) > 10 THEN 'KRITIS'
         WHEN SUM(CASE WHEN STATUS = 'K' THEN 1 ELSE 0 END) > 0 THEN 'PERHATIAN'
         ELSE 'NORMAL'
       END AS kondisi
  FROM DTALIB.PESAN_KORELASI
 WHERE WAKTU_KIRIM > CURRENT TIMESTAMP - 1 HOUR
 GROUP BY KD_MITRA;
```

🤝 Bagian XXX — Manajemen Vendor dan Dukungan

Bab 105. Dukungan IBM

105.1 Sebelum membuka kasus

Waktu yang dihabiskan mengumpulkan data sebelum membuka kasus akan kembali berlipat dalam bentuk penyelesaian yang lebih cepat.

- ☐ Nomor seri dan model mesin
- ☐ Rilis OS dan level PTF group
- ☐ Pesan lengkap termasuk teks tingkat kedua
- ☐ Job log lengkap, bukan potongan
- ☐ Waktu kejadian yang persis
- ☐ Apa yang berubah sebelum masalah muncul
- ☐ Apakah bisa diulang, dan bagaimana caranya
- ☐ Dampak bisnis, dinyatakan dalam kalimat yang jelas

```
/* Mengumpulkan data untuk IBM */
DSPPTF LICPGM(5770SS1) OUTPUT(*PRINT)
WRKPTFGRP OUTPUT(*PRINT)
DSPJOBLOG JOB(nomor/user/nama) OUTPUT(*PRINT)

/* Paket data servis bila diminta */
SNDPTFORD PTFID((SF99999))
```

105.2 Tingkat keparahan

Severity	Definisi IBM	Ekspektasi respons
1	Dampak kritis pada bisnis, sistem	24 jam terus-menerus

Severity	Definisi IBM	Ekspektasi respons
	tidak beroperasi	
2	Dampak signifikan, sebagian fungsi hilang	Jam kerja, prioritas tinggi
3	Dampak kecil, ada jalan pintas	Jam kerja
4	Dampak minimal, pertanyaan	Jam kerja

Severity 1 menuntut komitmen dari pihak Anda juga. IBM akan bekerja terus-menerus, dan mengharapkan orang di pihak Anda tersedia terus-menerus juga. Membuka severity 1 lalu tidak ada yang menjawab telepon pada pukul dua pagi akan menurunkan tingkat keparahan kasus.

105.3 Eskalasi

Bila kasus tidak bergerak, jalur eskalasi ada dan sah untuk dipakai. Yang perlu disiapkan:

1. Nomor kasus dan kronologi lengkap
2. Dampak bisnis dalam angka, bukan adjektiva
3. Apa yang sudah dicoba
4. Apa yang Anda butuhkan, secara spesifik

Kalimat "ini berdampak besar" jauh kurang efektif dibanding "sejak Selasa, 40.000 transaksi per hari gagal, dan kami menjalankan proses manual dengan 12 orang tambahan".

105.4 Menyimpan riwayat kasus

```
CREATE TABLE APPWRK.KASUS_VENDOR (
    NO_KASUS     VARCHAR(30)  NOT NULL,
    VENDOR       VARCHAR(30)  NOT NULL,
    DIBUKA       DATE         NOT NULL,
    DITUTUP      DATE,
    SEVERITY     SMALLINT,
    RINGKASAN    VARCHAR(200),
    AKAR_MASALAH VARCHAR(500),
    SOLUSI       VARCHAR(500),
    PTF_TERKAIT  VARCHAR(100),
    PRIMARY KEY (NO_KASUS, VENDOR)
);
```

Riwayat ini menjawab pertanyaan yang sering muncul: apakah masalah ini pernah terjadi sebelumnya, dan apa solusinya? Tanpa pencatatan, tim yang sama menyelesaikan masalah yang sama dua kali dalam tiga tahun.

🤝 Bab 106. Vendor Aplikasi

106.1 Klausul kontrak yang menentukan

Klausul	Kenapa penting
Tingkat layanan dengan angka	Tanpa angka, tidak ada yang bisa ditagih
Dukungan untuk rilis OS target	Menentukan apakah Anda bisa upgrade
Kewajiban memberi tahu akhir dukungan	Waktu untuk merencanakan
Escrow kode sumber	Perlindungan bila vendor berhenti

Klausul	Kenapa penting
Hak audit	Verifikasi keamanan dan kepatuhan
Kepemilikan data	Data tetap milik bank
Kewajiban keamanan dan pelaporan insiden	Rantai tanggung jawab jelas
Prosedur keluar	Bagaimana berpindah bila diperlukan

Dua klausul terakhir sering diabaikan saat kontrak ditandatangani dan paling mahal ketika dibutuhkan.

106.2 Batas akses vendor

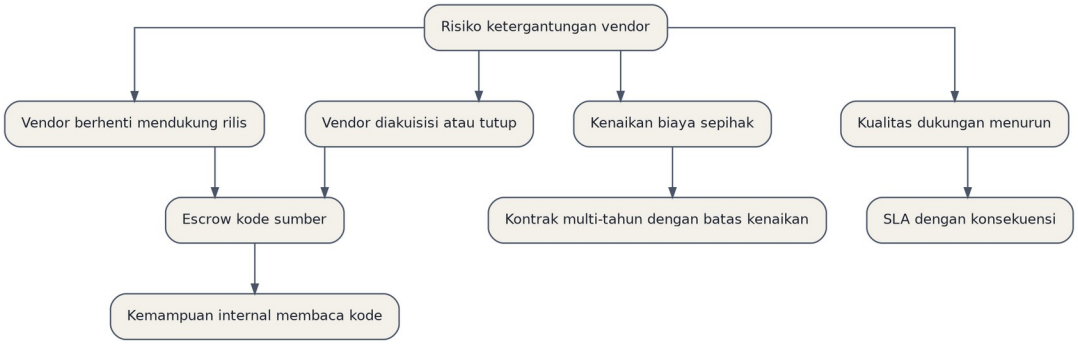
Akses	Aturan
Produksi	Hanya dengan pendampingan, sesi terekam
Data nasabah	Tidak, kecuali di-mask
Wewenang khusus	Tidak pernah `*ALLOBJ`
Jendela akses	Terbatas waktu, dicabut otomatis
Pencatatan	`CHGUSRAUD` penuh
Koneksi jarak jauh	Lewat jalur terkendali, bukan akses langsung

```
/* Profil vendor dengan pembatasan */
CRTUSRPRF USRPRF(VENDOR01) PASSWORD(*NONE) +
          STATUS(*DISABLED) +
          USRCLS(*USER) SPCAUT(*NONE) +
          LMTCPB(*YES) +
          INLMNU(*SIGNOFF) +
          TEXT('Vendor - aktifkan hanya saat dibutuhkan')

/* Aktifkan saat sesi dijadwalkan */
CHGUSRPRF USRPRF(VENDOR01) STATUS(*ENABLED) +
          PWDEXPITV(1)
CHGUSRAUD USRPRF(VENDOR01) +
          AUDLVL(*CMD *CREATE *DELETE *OBJMGT *SAVRST *SECURITY)

/* Nonaktifkan setelah selesai, jangan menunggu */
CHGUSRPRF USRPRF(VENDOR01) STATUS(*DISABLED)
```

```
-- Apa yang dikerjakan vendor selama sesi
SELECT ENTRY_TIMESTAMP, ENTRY_TYPE,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE,
       ENTRY_DATA
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 8 HOURS))
 WHERE USER_NAME = 'VENDOR01'
 ORDER BY ENTRY_TIMESTAMP;
```

Jalankan kueri ini setelah setiap sesi vendor dan simpan hasilnya. Ini bukan soal ketidakpercayaan; ini catatan yang melindungi kedua pihak ketika terjadi perselisihan tentang apa yang diubah.

106.3 Ketergantungan yang perlu dikelola



Gambar 100

Kotak terakhir sering terlupakan. Escrow kode sumber tidak berguna bila tidak ada seorang pun di organisasi Anda yang bisa membaca dan memelihara kode itu. Kemampuan internal membaca RPG adalah bagian dari strategi mitigasi risiko vendor, bukan sekadar keterampilan teknis.

106.4 Tinjauan vendor berkala

Yang ditinjau	Frekuensi	Sumber data
Kepatuhan terhadap SLA	Triwulanan	Riwayat kasus
Jumlah dan tingkat keparahan insiden	Triwulanan	Sistem tiket
Waktu penyelesaian rata-rata	Triwulanan	Riwayat kasus
Roadmap produk	Tahunan	Pertemuan vendor
Dukungan untuk rilis mendatang	Tahunan	Konfirmasi tertulis
Kesehatan finansial vendor	Tahunan	Laporan publik

Baris terakhir terdengar di luar wilayah Anda, tetapi vendor yang bermasalah secara finansial adalah risiko operasional yang nyata. Bila vendor Anda perusahaan publik, laporan tahunannya tersedia untuk dibaca.

✍️ Bagian XXXI — Latihan dan Studi Mandiri

Bagian ini dirancang untuk pelatihan internal. Soal disusun berdasarkan situasi nyata, bukan hafalan definisi. Kunci jawaban ada di Bab 110.

✍️ Bab 107. Soal Pemahaman Dasar

L1.1 Sebuah program gagal dengan `CPF9801`. Sebutkan tiga penyebab yang paling mungkin, dan perintah apa yang Anda jalankan pertama untuk masing-masing.

L1.2 Jelaskan perbedaan `*CAT`, `*TCAT`, dan `*BCAT` dalam CL, dan berikan satu contoh kesalahan yang muncul bila tertukar.

L1.3 Apa yang terjadi bila sebuah program RPG melakukan `CHAIN` pada file yang dibuka `usage(*update)` lalu tidak melakukan `update`, `delete`, maupun `unlock`?

L1.4 Sebutkan urutan pemeriksaan wewenang IBM i dari awal sampai akhir. Kenapa mencabut wewenang privat seorang pengguna belum tentu memblokir aksesnya?

L1.5 Sebuah job berstatus `LCKW`. Tulis satu kueri SQL untuk menemukan siapa yang menahan kuncinya.

L1.6 Jelaskan perbedaan `JOBPTY` dan `RUNPTY`, dan mana yang Anda ubah bila sebuah job batch memperlambat sesi interaktif.

L1.7 Kenapa `ACTGRP(*NEW)` berbahaya bila program dipanggil di dalam perulangan jutaan kali?

L1.8 Sistem Anda menunjukkan pemakaian ASP 87%. Sebutkan lima tindakan berurutan dari yang paling aman ke yang paling berisiko.

✍️ Bab 108. Soal Situasi Operasional

L2.1 Pukul 02:30, fase akrual EOD berstatus gagal. Seorang operator menghubungi Anda dan mengusulkan menjalankan ulang fase itu karena "biasanya berhasil kalau diulang". Apa jawaban Anda, dan apa yang Anda lakukan?

L2.2 Pemeriksaan pagi menunjukkan tanggal buku aplikasi tertinggal satu hari dari tanggal sistem. Apa artinya, dan apa tiga langkah pertama Anda?

L2.3 Backup semalam gagal dengan `CPF3793`. EOD sudah berjalan normal dan layanan sudah buka. Apakah Anda menunggu malam berikutnya? Jelaskan alasannya.

L2.4 Sebuah cabang melaporkan setiap transaksi memakan 10 detik. Cabang lain normal. Tulis urutan diagnosis Anda, minimal lima langkah, dengan perintah yang dijalankan.

L2.5 Setelah uji DR, seluruh kanal berfungsi kecuali mobile banking. Sebutkan penyebab yang paling mungkin dan cara memverifikasinya.

L2.6 EOD berjalan normal, tidak ada pesan kesalahan, tetapi keesokan harinya 400 nasabah mengeluh bunga deposito tidak masuk. Bagaimana Anda menyelidikinya?

L2.7 Anda diminta memberi akses produksi kepada konsultan vendor untuk memperbaiki masalah aplikasi yang mendesak. Sebutkan syarat minimum yang Anda tetapkan sebelum menyetujui.

L2.8 Job EOD berstatus berjalan selama tiga jam tanpa progres dan tanpa pesan. Tulis urutan pemeriksaan Anda.

✍️ Bab 109. Soal Rancangan dan Analisis

L3.1 Anda diminta merancang strategi backup untuk sistem core banking dengan RPO 15 menit dan RTO 2 jam. Jelaskan rancangan Anda, termasuk apa yang di-backup, kapan, dan mekanisme apa yang dipakai untuk mencapai RPO tersebut.

L3.2 Sebuah tabel transaksi berisi 800 juta baris dan tumbuh 2 juta per hari. Kueri laporan bulanan memakan 4 jam. Rancang perbaikannya, dan jelaskan mengapa pendekatan Anda lebih baik daripada sekadar menambah index.

L3.3 Manajemen ingin menerapkan deploy otomatis ke produksi seperti yang dilakukan tim aplikasi web. Tulis tanggapan Anda dalam bentuk argumen yang berimbang: apa yang layak diambil, apa yang tidak, dan alasannya.

L3.4 Rancang mekanisme agar fase akrual EOD tidak mungkin dijalankan dua kali pada tanggal buku yang sama. Tulis strukturnya, termasuk tabel dan potongan kode.

L3.5 Sebuah rekening penampungan menerima 200 transaksi per detik pada jam sibuk, dan menjadi titik kontensi. Sebutkan dua pendekatan penyelesaian beserta konsekuensi masing-masing.

L3.6 Tim keamanan menemukan bahwa seorang administrator bisa melihat seluruh data nasabah. Rancang kontrol yang membatasi ini tanpa menghentikan kemampuannya mengelola sistem.

L3.7 Anda menemukan 340 program yang masih ditulis dalam fixed format RPG III. Manajemen bertanya apakah semuanya harus dimodernisasi. Susun kriteria keputusan Anda.

L3.8 Rancang pemantauan yang akan mendeteksi interface yang berhenti berjalan sama sekali, bukan hanya yang gagal. Jelaskan mengapa keduanya berbeda.

Bab 110. Kunci Jawaban

110.1 Pemahaman dasar

L1.1 Tiga penyebab: objek benar-benar tidak ada, library list tidak memuat library yang berisi objek itu, atau pengguna tidak berwenang sehingga objek tampak tidak ada. Perintah pertama: `DSPLIBL` untuk memeriksa library list, `CHKOBJ` dengan kualifikasi penuh untuk memastikan keberadaan, dan `DSPOBJAUT` untuk memeriksa wewenang.

L1.2 `*CAT` menyambung apa adanya termasuk spasi kanan. `*TCAT` memotong spasi kanan variabel pertama sebelum menyambung. `*BCAT` memotong lalu menyisipkan satu spasi. Kesalahan lazim: membangun nama objek dengan `*CAT` menghasilkan nama berisi spasi di tengah, sehingga `CHKOBJ` gagal dengan pesan yang membingungkan.

L1.3 Record tetap terkunci sampai record berikutnya dibaca pada file yang sama atau program berakhir. Bila program terus berjalan tanpa membaca record lain, kunci bertahan dan job lain yang membutuhkannya akan berstatus `LCKW`. Ini penyebab paling umum kontensi di EOD.

L1.4 Urutannya: wewenang khusus `*ALLOBJ`, wewenang privat atas objek, wewenang lewat group profile, authorization list, lalu `*PUBLIC`. Mencabut wewenang privat tidak memblokir akses bila `*PUBLIC` masih memberikan wewenang yang cukup, atau bila pengguna anggota group yang berwenang.

L1.5

```
SELECT JOB_NAME, LOCK_STATE, TABLE_SCHEMA, TABLE_NAME, RRN
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'DTALIB';
```

L1.6 `JOBPTY` adalah prioritas job saat menunggu di job queue; ia menentukan urutan job diambil. `RUNPTY` adalah prioritas saat job sudah berjalan; ia menentukan pembagian CPU. Untuk job batch yang

memperlambat sesi interaktif, yang diubah adalah `RUNPTY` dengan `CHGJOB RUNPTY(50)` — angka lebih besar berarti prioritas lebih rendah.

L1.7 `ACTGRP(*NEW)` membuat activation group baru pada setiap pemanggilan dan menghancurkannya saat program berakhir. Pembuatan dan penghancuran itu mahal. Dalam perulangan jutaan kali, biaya ini mendominasi waktu eksekusi dan bisa membuat proses berkali-kali lebih lambat daripada memakai activation group bernama.

L1.8 Berurutan dari paling aman: hapus spooled file yang melewati retensi; hapus save file yang sudah dipindahkan ke media; simpan lalu lepas journal receiver lama yang sudah ter-backup; arsipkan dan hapus data tabel riwayat yang melewati retensi; pindahkan library arsip ke ASP lain atau media eksternal. `RCLSTG` disebut terakhir karena menuntut restricted state.

110.2 Situasi operasional

L2.1 Jawaban: tidak. Fase akrual tidak idempoten — menjalankan ulang tanpa membalik hasil sebelumnya akan membukukan bunga dua kali. Yang dilakukan: hentikan seluruh job EOD yang masih berjalan, catat jumlah record yang sudah terproses dari tabel log bunga, eskalasi ke manajer TI dan Application Support karena ini keputusan dua orang, dan siapkan opsi restore dari Checkpoint A. Dokumentasikan setiap langkah dengan stempel waktu.

L2.2 Artinya EOD semalam tidak selesai. Tiga langkah pertama: periksa tabel kontrol EOD untuk melihat fase mana yang berhenti dan dengan status apa; periksa job log fase tersebut untuk pesan kesalahan; tentukan apakah layanan sudah terlanjur dibuka dengan tanggal buku lama, karena itu mengubah seluruh pilihan penanganan.

L2.3 Tidak menunggu. Jalankan ulang backup segera meskipun di jam layanan, memakai `SAVACT(*SYNCLIB)` agar tidak mengganggu operasi. Alasannya: setiap jam tanpa titik pulih adalah risiko yang tidak perlu, dan menunggu berarti dua hari data tanpa perlindungan. Bila gagal dua malam berturut-turut, ini insiden tingkat tinggi yang harus dilaporkan.

L2.4 Karena hanya satu cabang yang terdampak, masalah kemungkinan besar pada jalur, bukan pada sistem. Urutan: verifikasi beban sistem normal dengan `WRKSYSSTS`; ping cabang dengan paket kecil; ping dengan `PKTLEN(1472)` untuk menguji MTU; `TRACEROUTE` untuk melihat di hop mana waktu melonjak; periksa `NETSTAT_JOB_INFO` untuk koneksi dari cabang itu; periksa statistik kesalahan interface. Kegagalan pada paket besar sementara paket kecil berhasil menunjuk masalah MTU.

L2.5 Penyebab paling mungkin: middleware mobile memakai daftar putih alamat IP, dan partisi DR punya alamat berbeda. Verifikasi: coba koneksi dari sisi middleware ke alamat DR, periksa log penolakan di middleware, dan bandingkan alamat yang terdaftar dengan alamat DR yang sebenarnya. Perbaikan jangka panjang: virtual IP yang berpindah saat failover.

L2.6 Ini kegagalan diam, bukan kegagalan yang berteriak. Selidiki dengan: bandingkan jumlah record di tabel log bunga dengan jumlah rekening aktif yang seharusnya diproses; kelompokkan rekening yang terdampak untuk menemukan pola, kemungkinan besar satu kode produk; periksa apakah kode produk itu punya rate

terdaftar dan berlaku pada tanggal tersebut; periksa apakah program mengembalikan nol saat rate tidak ditemukan. Perbaikan: program harus membedakan rate nol yang sah dari rate tidak ditemukan, dan validasi pra-akrual harus memastikan setiap produk aktif punya rate.

L2.7 Syarat minimum: tiket perubahan yang disetujui; profil vendor terpisah dengan `LMTCPB(*YES)` dan tanpa wewenang khusus; `CHGUSRAUD` penuh diaktifkan; jendela waktu terbatas dengan pencabutan otomatis; pendampingan oleh staf internal selama sesi; sesi terekam; data nasabah di-mask bila memungkinkan; dan tinjauan audit journal setelah sesi berakhir.

L2.8 Urutan: periksa status job dengan `ACTIVE_JOB_INFO`; bila `MSGW`, baca job log dan jawab pesan dengan benar; bila `LCKW`, cari job penahan dengan `RECORD_LOCK_INFO`; bila `DEQW`, periksa call stack dengan `WRKJOB OPTION(*PGMSTK)` untuk melihat apakah menunggu data queue; bila `RUN` dengan CPU tinggi, curigai perulangan atau table scan; bila `RUN` dengan CPU rendah, periksa respons disk dan seize wait.

110.3 Rancangan dan analisis

L3.1 RPO 15 menit tidak bisa dicapai dengan backup tape saja. Rancangan: journaling aktif pada seluruh tabel produksi dengan receiver di ASP terpisah; replikasi journal ke sistem DR secara berkelanjutan; save harian dengan `SAVACT(*SYNCLIB)` sebelum dan sesudah EOD; save mingguan seluruh library pengguna; `SAVSYS` bulanan. RPO 15 menit dicapai dari replikasi journal, bukan dari save. RTO 2 jam menuntut prosedur restore yang sudah dilatih dan media yang tersedia di lokasi, serta kemungkinan besar PowerHA atau replikasi logis agar tidak bergantung pada restore penuh.

L3.2 Menambah index tidak menolong karena kueri mengagregasi hampir seluruh tabel; index justru menambah beban tulis pada tabel yang menerima 2 juta insert per hari. Rancangan yang lebih baik: tabel ringkasan saldo harian yang diisi sekali per hari sebagai bagian dari EOD, sehingga laporan bulanan menjumlahkan 30 baris ringkasan alih-alih memindai ratusan juta baris. Tambahkan pemisahan tabel aktif dan arsip agar tabel transaksi aktif hanya memuat periode berjalan. Pertimbangkan EVI dengan `INCLUDE` untuk agregat yang sering diminta.

L3.3 Yang layak diambil: kendali versi untuk kode sumber, pipeline build otomatis, uji otomatis, artefak berversi, dan rollback yang teruji. Semuanya meningkatkan keandalan tanpa menambah risiko. Yang tidak layak: deploy otomatis ke produksi tanpa persetujuan, karena pemisahan tugas dan persetujuan perubahan adalah kewajiban regulatoris, bukan preferensi proses. Frekuensi deploy mingguan sudah agresif untuk sistem yang memindahkan uang. Argumen kuncinya: praktik DevOps lahir dari lingkungan di mana biaya kesalahan rendah dan rollback murah; di core banking keduanya tidak berlaku.

L3.4 Tabel penanda dengan kunci gabungan tanggal buku dan kode fase, kolom status. Program akrual memeriksa di awal apakah ada baris dengan status selesai untuk tanggal buku berjalan, dan melempar escape message bila ada. Penanda ditulis dalam transaksi yang sama dengan pemrosesan. Contoh strukturnya ada di Bab 59.5. Tambahkan kontrol kedua berupa rekonsiliasi pasca-akrual yang membandingkan total log bunga dengan total posting GL.

L3.5 Pendekatan pertama: pecah saldo menjadi beberapa slot yang dijumlahkan, sehingga transaksi menulis

ke slot berbeda berdasarkan hash. Konsekuensinya saldo tidak lagi tersedia sebagai satu nilai dan setiap pembacaan menuntut agregasi. Pendekatan kedua: catat mutasi ke tabel terpisah dan hitung saldo sebagai agregat dengan tabel ringkasan berkala. Konsekuensinya saldo waktu nyata menuntut penjumlahan mutasi sejak ringkasan terakhir. Keduanya menukar kesederhanaan dengan throughput, dan hanya layak untuk rekening yang terbukti menjadi titik panas.

L3.6 Row and Column Access Control adalah jawabannya, karena RCAC berlaku bahkan untuk profil dengan `*ALLOBJ`. Pasang column mask pada kolom sensitif seperti NIK dan nomor kartu, dan row permission yang membatasi berdasarkan peran. Administrator tetap bisa mengelola objek, menjalankan backup, dan menyetel kinerja, tetapi tidak bisa membaca isi data nasabah. Lengkapi dengan audit objek tingkat `*ALL` pada tabel tersebut, sehingga setiap pembacaan tercatat.

L3.7 Kriteria untuk membiarkan kode apa adanya: berjalan stabil bertahun-tahun tanpa insiden, tidak akan berubah karena kebutuhan bisnis, kinerjanya memadai, dan tidak menghalangi perubahan lain. Kode yang memenuhi keempatnya lebih baik didokumentasikan daripada disentuh. Prioritaskan modernisasi pada kode yang sering berubah, yang pernah menyebabkan insiden, atau yang menghalangi integrasi baru. Jawaban untuk manajemen: tidak, tidak semuanya, dan modernisasi tanpa tujuan bisnis adalah cara memperkenalkan cacat baru ke sistem yang stabil.

L3.8 Keduanya berbeda karena kegagalan menghasilkan pesan sedangkan berhenti berjalan tidak menghasilkan apa pun. Pemantauan berbasis pesan kesalahan tidak akan pernah mendeteksi proses yang tidak pernah dimulai. Rancangan: tabel jadwal yang mendaftarkan setiap interface beserta jam harapannya, dan kueri terjadwal yang mencari interface yang jam harapannya sudah lewat tetapi belum ada catatan keberhasilan hari itu. Contoh kuerinya ada di Bab 64.4.

✍️ Bab 111. Proyek Latihan

Untuk pelatihan yang lebih dalam, berikan proyek berikut kepada anggota tim baru. Masing-masing menghasilkan sesuatu yang benar-benar dipakai.

Proyek	Hasil	Perkiraan waktu
Bangun pemantauan kesehatan harian	Program CL dan tabel riwayat	1 minggu
Tulis runbook untuk tiga insiden tersering	Tiga runbook teruji	2 minggu
Bangun baseline konfigurasi otomatis	Program dan laporan pembanding	1 minggu
Lakukan restore satu library ke UAT	Berita acara dengan durasi aktual	3 hari
Audit seluruh profil berwenang khusus	Laporan temuan dan rekomendasi	1 minggu
Analisis satu fase EOD yang melambat	Laporan studi kinerja	2 minggu
Petakan seluruh interface dan kontraknya	Dokumen inventaris	2 minggu
Buat view kepatuhan otomatis	View SQL siap audit	1 minggu

Proyek keempat sengaja diberi waktu pendek: tujuannya bukan kesempurnaan, melainkan memastikan

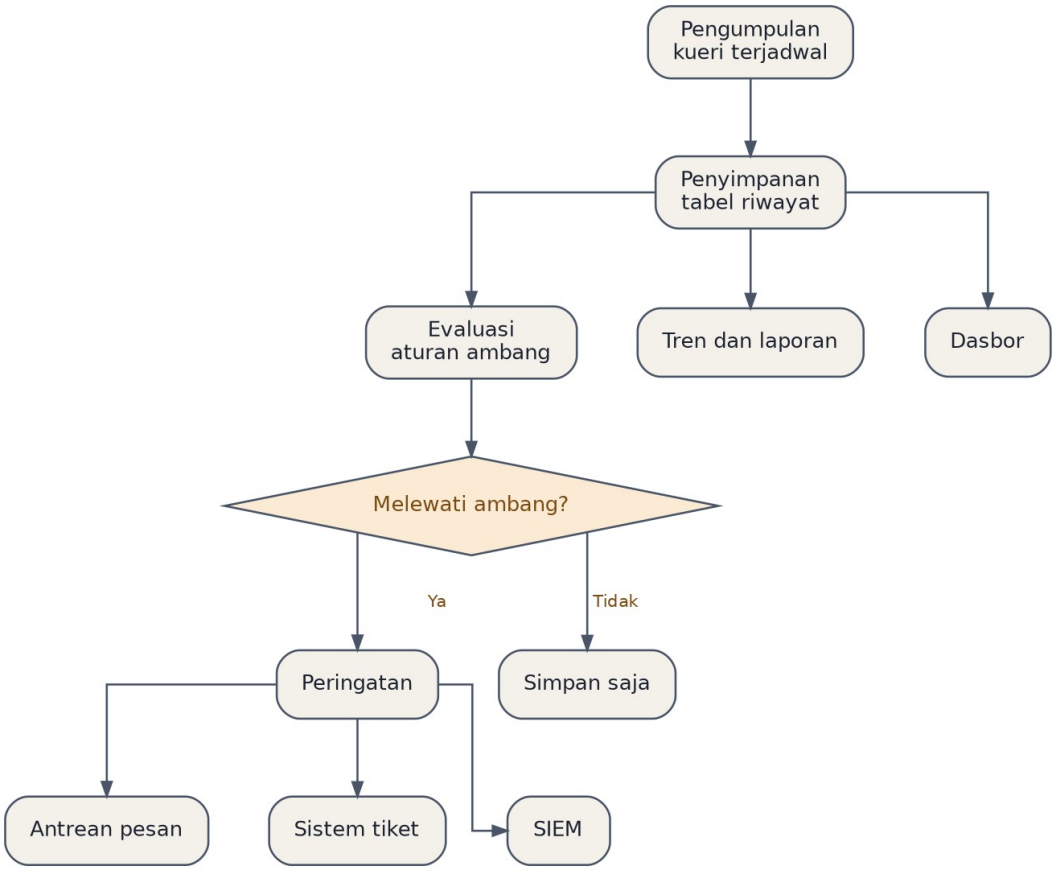
orang itu pernah benar-benar melakukan restore sebelum ia dibutuhkan melakukannya sungguhan.

📊 Bagian XXXII — Membangun Pemantauan Terpadu

Bab-bab sebelumnya memberi potongan kueri pemantauan. Bagian ini merangkainya menjadi satu sistem yang berjalan sendiri dan memberi tahu Anda sebelum orang lain mengeluh.

📊 Bab 112. Arsitektur Pemantauan

112.1 Tiga lapisan



Gambar 101

Memisahkan pengumpulan dari evaluasi itu penting: data yang tidak memicu peringatan hari ini tetap berguna sebagai baseline enam bulan kemudian.

112.2 Skema penyimpanan

```
-- Satu tabel untuk seluruh metrik numerik
CREATE TABLE MONITOR.METRIK (
    WAKTU      TIMESTAMP   NOT NULL,
    KATEGORI   VARCHAR(20) NOT NULL,
    NAMA       VARCHAR(40) NOT NULL,
    NILAI      DECIMAL(19,4),
```

```
    SATUAN     VARCHAR(10),
    KETERANGAN VARCHAR(200)
);

CREATE INDEX MONITOR.METRIK_IX1 ON MONITOR.METRIK (WAKTU, KATEGORI);
CREATE INDEX MONITOR.METRIK_IX2 ON MONITOR.METRIK (NAMA, WAKTU);

-- Definisi ambang, sebagai data bukan kode
CREATE TABLE MONITOR.AMBANG (
    NAMA        VARCHAR(40) NOT NULL,
    AMBANG_WARN DECIMAL(19,4),
    AMBANG_CRIT DECIMAL(19,4),
    ARAH        CHAR(1) NOT NULL,
    AKTIF       CHAR(1) NOT NULL DEFAULT 'Y',
    JAM_MULAI   TIME,
    JAM_SELESAI TIME,
    KETERANGAN  VARCHAR(200),
    PRIMARY KEY (NAMA)
);

-- ARAH: N berarti naik itu buruk, T berarti turun itu buruk

-- Riwayat peringatan
CREATE TABLE MONITOR.PERINGATAN (
    ID       BIGINT GENERATED ALWAYS AS IDENTITY,
    WAKTU    TIMESTAMP   NOT NULL,
    NAMA     VARCHAR(40) NOT NULL,
    TINGKAT  CHAR(1)     NOT NULL,
    NILAI    DECIMAL(19,4),
    AMBANG   DECIMAL(19,4),
    PESAN    VARCHAR(300),
    DIKIRIM  CHAR(1) DEFAULT 'N',
    PRIMARY KEY (ID)
);
```

Menyimpan ambang sebagai data, bukan sebagai angka di dalam program, berarti penyesuaian ambang tidak menuntut perubahan kode dan bisa dilakukan lewat perubahan data yang disetujui.

Kolom `JAM_MULAI` dan `JAM_SELESAI` memungkinkan ambang berbeda antara jam layanan dan jendela batch. Pemakaian CPU 95% pada pukul dua pagi adalah normal; pada pukul sepuluh pagi itu masalah.

112.3 Program pengumpul

```
PGM
   /* --- Kategori SISTEM --- */
   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''SISTEM'', ''CPU_PERSEN'', +
            AVERAGE_CPU_UTILIZATION, ''%'' +
       FROM TABLE(QSYS2.SYSTEM_STATUS())') COMMIT(*NONE)

   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''SISTEM'', ''ASP_PERSEN'', +
            SYSTEM_ASP_USED, ''%'' +
       FROM TABLE(QSYS2.SYSTEM_STATUS())') COMMIT(*NONE)

   /* --- Kategori JOB --- */
   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
```

```
     SELECT CURRENT TIMESTAMP, ''JOB'', ''JOB_MSGW'', COUNT(*), ''job'' +
       FROM TABLE(QSYS2.ACTIVE_JOB_INFO( +
            JOB_STATUS_FILTER => ''*MSGW''))') COMMIT(*NONE)

   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''JOB'', ''JOB_LCKW'', COUNT(*), ''job'' +
       FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) +
      WHERE JOB_STATUS = ''LCKW''') COMMIT(*NONE)

   /* --- Kategori JARINGAN --- */
   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''JARINGAN'', ''CONN_ESTABLISHED'', +
            COUNT(*), ''koneksi'' +
       FROM QSYS2.NETSTAT_INFO +
      WHERE TCP_STATE = ''ESTABLISHED''') COMMIT(*NONE)

   /* --- Kategori APLIKASI --- */
   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''APLIKASI'', ''EOD_FASE_GAGAL'', +
            COUNT(*), ''fase'' +
       FROM APPWRK.EOD_KONTROL +
      WHERE TGL_BUKU = CURRENT DATE AND STATUS = ''E''') COMMIT(*NONE)

   RUNSQL SQL('INSERT INTO MONITOR.METRIK +
     (WAKTU, KATEGORI, NAMA, NILAI, SATUAN) +
     SELECT CURRENT TIMESTAMP, ''APLIKASI'', ''PESAN_MENGGANTUNG'', +
            COUNT(*), ''pesan'' +
       FROM DTALIB.PESAN_KORELASI +
      WHERE STATUS = ''K'' +
        AND WAKTU_KIRIM < CURRENT TIMESTAMP - 10 MINUTES') COMMIT(*NONE)

   /* Evaluasi ambang setelah seluruh metrik terkumpul */
   CALL PGM(MONITOR/EVALUASI)
ENDPGM
```

112.4 Evaluasi ambang

```
INSERT INTO MONITOR.PERINGATAN (WAKTU, NAMA, TINGKAT, NILAI, AMBANG, PESAN)
WITH terbaru AS (
  SELECT m.NAMA, m.NILAI, m.SATUAN, m.WAKTU,
         ROW_NUMBER() OVER (PARTITION BY m.NAMA ORDER BY m.WAKTU DESC) AS urut
    FROM MONITOR.METRIK m
   WHERE m.WAKTU > CURRENT TIMESTAMP - 15 MINUTES
),
dinilai AS (
  SELECT t.NAMA, t.NILAI, t.SATUAN,
         a.AMBANG_WARN, a.AMBANG_CRIT, a.ARAH,
         CASE WHEN a.ARAH = 'N' AND t.NILAI >= a.AMBANG_CRIT THEN 'K'
              WHEN a.ARAH = 'T' AND t.NILAI <= a.AMBANG_CRIT THEN 'K'
              ELSE 'W' END AS tingkat
    FROM terbaru t
    JOIN MONITOR.AMBANG a ON a.NAMA = t.NAMA
   WHERE t.urut = 1
     AND a.AKTIF = 'Y'
     AND (a.JAM_MULAI IS NULL
          OR CURRENT TIME BETWEEN a.JAM_MULAI AND a.JAM_SELESAI)
     AND ((a.ARAH = 'N' AND t.NILAI >= a.AMBANG_WARN)
       OR (a.ARAH = 'T' AND t.NILAI <= a.AMBANG_WARN))
```

```
)
SELECT CURRENT TIMESTAMP, NAMA, tingkat, NILAI,
       CASE WHEN tingkat = 'K' THEN AMBANG_CRIT ELSE AMBANG_WARN END,
       NAMA || ' = ' || VARCHAR(NILAI) || ' ' || COALESCE(SATUAN, '')
  FROM dinilai;
```

112.5 Menekan peringatan berulang

Satu masalah yang berlangsung tiga jam tidak boleh menghasilkan tiga puluh enam peringatan identik. Tim yang dibanjiri peringatan akan berhenti membacanya, dan itu lebih berbahaya daripada tidak punya pemantauan sama sekali.

```
-- Kirim hanya bila belum ada peringatan sama dalam 60 menit terakhir
SELECT p.ID, p.NAMA, p.TINGKAT, p.PESAN
  FROM MONITOR.PERINGATAN p
 WHERE p.DIKIRIM = 'N'
   AND NOT EXISTS (
       SELECT 1 FROM MONITOR.PERINGATAN q
        WHERE q.NAMA = p.NAMA
          AND q.TINGKAT = p.TINGKAT
          AND q.DIKIRIM = 'Y'
          AND q.WAKTU > p.WAKTU - 60 MINUTES
          AND q.ID < p.ID);
```

Pengecualian: peringatan yang naik tingkat dari perhatian menjadi kritis harus selalu dikirim, karena itu informasi baru.

112.6 Ambang sebagai titik awal

```
INSERT INTO MONITOR.AMBANG
  (NAMA, AMBANG_WARN, AMBANG_CRIT, ARAH, JAM_MULAI, JAM_SELESAI, KETERANGAN)
VALUES
  ('CPU_PERSEN',        70,  85, 'N', '07:00:00', '17:00:00', 'Jam layanan'),
  ('ASP_PERSEN',        75,  85, 'N', NULL, NULL, 'Sepanjang waktu'),
  ('JOB_MSGW',           1,   3, 'N', NULL, NULL, 'Job menunggu jawaban'),
  ('JOB_LCKW',           3,  10, 'N', NULL, NULL, 'Kontensi kunci'),
  ('EOD_FASE_GAGAL',     1,   1, 'N', NULL, NULL, 'Satu pun tidak boleh'),
  ('PESAN_MENGGANTUNG',  5,  20, 'N', NULL, NULL, 'Integrasi asinkron'),
  ('CONN_ESTABLISHED', 500, 800, 'N', NULL, NULL, 'Sesuaikan kapasitas');
```

Sesuaikan setelah mengumpulkan data dua minggu. Ambang yang ditetapkan tanpa baseline akan salah, dan ambang yang salah menghasilkan peringatan palsu yang merusak kepercayaan pada sistem.

112.7 Deteksi berbasis penyimpangan

Ambang tetap gagal menangkap masalah yang besarnya relatif terhadap kebiasaan. Tambahkan lapisan kedua.

```
CREATE OR REPLACE VIEW MONITOR.V_PENYIMPANGAN AS
WITH baseline AS (
  SELECT NAMA, AVG(NILAI) AS rata, STDDEV(NILAI) AS simpangan
    FROM MONITOR.METRIK
   WHERE WAKTU BETWEEN CURRENT TIMESTAMP - 14 DAYS
                   AND CURRENT TIMESTAMP - 1 DAY
     AND HOUR(WAKTU) = HOUR(CURRENT TIMESTAMP)
   GROUP BY NAMA
  HAVING COUNT(*) > 20
),
```

```
sekarang AS (
  SELECT NAMA, NILAI, WAKTU,
         ROW_NUMBER() OVER (PARTITION BY NAMA ORDER BY WAKTU DESC) AS urut
    FROM MONITOR.METRIK
   WHERE WAKTU > CURRENT TIMESTAMP - 15 MINUTES
)
SELECT s.NAMA, s.NILAI AS nilai_kini,
       DECIMAL(b.rata, 15, 2) AS rata_biasa,
       DECIMAL(b.simpangan, 15, 2) AS simpangan,
       DECIMAL((s.NILAI - b.rata) / NULLIF(b.simpangan, 0), 9, 2) AS skor
  FROM sekarang s
  JOIN baseline b ON b.NAMA = s.NAMA
 WHERE s.urut = 1
   AND b.simpangan > 0
   AND ABS((s.NILAI - b.rata) / b.simpangan) > 3;
```

Membandingkan terhadap jam yang sama pada hari-hari sebelumnya itu penting: beban pukul sepuluh pagi dan pukul dua pagi sangat berbeda, dan baseline tunggal akan mengaburkan keduanya.

⚙️ Bab 113. Integrasi dengan Sistem Luar

113.1 Mengirim ke SIEM

Bank umumnya punya SIEM terpusat. IBM i harus ikut mengirim ke sana, bukan menjadi pulau yang terpantau terpisah.

Data yang dikirim	Sumber	Frekuensi
Kegagalan wewenang	Audit journal `AF`	Tiap 5 menit
Login gagal	Audit journal `PW`	Tiap 5 menit
Perubahan profil	Audit journal `CP`	Tiap 15 menit
Perubahan system value	Audit journal `SV`	Tiap 15 menit
Objek dihapus di produksi	Audit journal `DO`	Tiap 15 menit
Peringatan operasional	Tabel peringatan	Waktu nyata

```
-- Ekstraksi bertahap dengan penanda posisi
CREATE TABLE MONITOR.SIEM_POSISI (
    JENIS  VARCHAR(20) NOT NULL,
    POSISI TIMESTAMP   NOT NULL,
    PRIMARY KEY (JENIS)
);

-- Ambil hanya yang baru sejak ekstraksi terakhir
SELECT ENTRY_TIMESTAMP, ENTRY_TYPE, USER_NAME, JOB_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, REMOTE_ADDRESS
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'AF,PW,CP,SV,DO',
         STARTING_TIMESTAMP => (SELECT POSISI FROM MONITOR.SIEM_POSISI
                                WHERE JENIS = 'AUDIT')))
 ORDER BY ENTRY_TIMESTAMP;
```

```
#!/QOpenSys/pkgs/bin/bash
# Kirim sebagai syslog ke kolektor SIEM
KOLEKTOR=10.20.50.100
PORT=514

while IFS='|' read -r waktu jenis pengguna objek; do
  logger -n $KOLEKTOR -P $PORT -t IBMi-AUDIT \
    "type=$jenis user=$pengguna object=$objek time=$waktu"
done < /monitor/out/audit-baru.txt
```

Prinsip yang penting. Jejak audit harus keluar dari sistem yang diauditnya. Seseorang yang mengompromikan IBM i bisa menghapus audit journal lokal, tetapi tidak bisa menghapus apa yang sudah terkirim ke SIEM.

113.2 Dasbor

```
CREATE OR REPLACE VIEW MONITOR.V_DASBOR AS
WITH terbaru AS (
  SELECT NAMA, NILAI, SATUAN, WAKTU,
         ROW_NUMBER() OVER (PARTITION BY NAMA ORDER BY WAKTU DESC) AS urut
    FROM MONITOR.METRIK
   WHERE WAKTU > CURRENT TIMESTAMP - 30 MINUTES
)
SELECT t.NAMA, t.NILAI, t.SATUAN, t.WAKTU,
       a.AMBANG_WARN, a.AMBANG_CRIT,
       CASE
         WHEN a.NAMA IS NULL THEN 'INFO'
         WHEN a.ARAH = 'N' AND t.NILAI >= a.AMBANG_CRIT THEN 'KRITIS'
         WHEN a.ARAH = 'N' AND t.NILAI >= a.AMBANG_WARN THEN 'PERHATIAN'
         WHEN a.ARAH = 'T' AND t.NILAI <= a.AMBANG_CRIT THEN 'KRITIS'
         WHEN a.ARAH = 'T' AND t.NILAI <= a.AMBANG_WARN THEN 'PERHATIAN'
         ELSE 'NORMAL'
       END AS kondisi
  FROM terbaru t
  LEFT JOIN MONITOR.AMBANG a ON a.NAMA = t.NAMA
 WHERE t.urut = 1;
```

View ini bisa dibaca lewat ODBC oleh alat dasbor apa pun yang sudah dipakai institusi Anda. Tidak perlu membangun antarmuka sendiri.

113.3 Laporan tren bulanan

```
SELECT NAMA,
       VARCHAR_FORMAT(WAKTU, 'YYYY-MM') AS bulan,
       DECIMAL(AVG(NILAI), 15, 2) AS rata,
       MAX(NILAI) AS puncak,
       DECIMAL(
         PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY NILAI), 15, 2
       ) AS persentil_95
  FROM MONITOR.METRIK
 WHERE NAMA IN ('CPU_PERSEN','ASP_PERSEN','CONN_ESTABLISHED')
   AND WAKTU > CURRENT DATE - 12 MONTHS
 GROUP BY NAMA, VARCHAR_FORMAT(WAKTU, 'YYYY-MM')
 ORDER BY NAMA, bulan;
```

Persentil 95 lebih berguna daripada rata-rata untuk perencanaan kapasitas. Rata-rata menyembunyikan puncak, dan sistem harus dirancang untuk puncak.

113.4 Retensi data pemantauan

Usia data	Resolusi	Alasan
0–7 hari	Tiap 5 menit	Investigasi insiden
8–90 hari	Rata-rata per jam	Tren jangka pendek
91 hari–2 tahun	Rata-rata harian	Perencanaan kapasitas
Di atas 2 tahun	Rata-rata bulanan	Laporan jangka panjang

```
-- Peringkasan terjadwal, dari 5 menit menjadi per jam
INSERT INTO MONITOR.METRIK_JAM
SELECT TIMESTAMP(DATE(WAKTU), TIME(CHAR(HOUR(WAKTU)) || ':00:00')) AS jam,
       KATEGORI, NAMA,
       AVG(NILAI) AS rata, MAX(NILAI) AS puncak, MIN(NILAI) AS terendah,
       SATUAN
  FROM MONITOR.METRIK
 WHERE WAKTU < CURRENT TIMESTAMP - 7 DAYS
 GROUP BY DATE(WAKTU), HOUR(WAKTU), KATEGORI, NAMA, SATUAN;

DELETE FROM MONITOR.METRIK WHERE WAKTU < CURRENT TIMESTAMP - 7 DAYS;
```

Tanpa peringkasan, tabel metrik akan tumbuh menjadi puluhan juta baris dalam setahun dan pemantauan Anda sendiri menjadi masalah kapasitas.

113.5 Memantau pemantauan

Hal terakhir yang perlu dipantau adalah apakah pemantauan itu sendiri berjalan.

```
-- Metrik yang seharusnya terkumpul tetapi tidak ada
SELECT a.NAMA, MAX(m.WAKTU) AS terakhir_terkumpul,
       TIMESTAMPDIFF(4, CHAR(CURRENT TIMESTAMP - MAX(m.WAKTU)))
         AS menit_lalu
  FROM MONITOR.AMBANG a
  LEFT JOIN MONITOR.METRIK m ON m.NAMA = a.NAMA
 WHERE a.AKTIF = 'Y'
 GROUP BY a.NAMA
HAVING MAX(m.WAKTU) IS NULL
    OR MAX(m.WAKTU) < CURRENT TIMESTAMP - 30 MINUTES;
```

Sistem pemantauan yang mati diam-diam memberi rasa aman palsu yang lebih berbahaya daripada tidak ada pemantauan sama sekali. Kueri ini harus dijalankan oleh mekanisme terpisah — penjadwal di sistem lain, atau sekurangnya job terpisah yang tidak bergantung pada program pengumpul yang sama.

💰 Bagian XXXIII — Lisensi, Biaya, dan Anggaran

System Engineer yang bisa menjelaskan biaya dalam bahasa manajemen mendapat anggaran. Yang tidak bisa, mendapat penolakan yang dikira keputusan teknis.

💰 Bab 114. Memahami Struktur Lisensi

114.1 Komponen biaya

Komponen	Dasar penagihan	Catatan
Perangkat keras	Satu kali, plus pemeliharaan	Siklus 5–7 tahun

Komponen	Dasar penagihan	Catatan
IBM i sistem operasi	Berdasarkan kelompok prosesor	Melekat pada mesin
Pengguna berlisensi	Jumlah pengguna	Periksa definisi yang berlaku
Licensed Program Product	Per produk	Hanya yang dipakai
Software maintenance	Tahunan	Wajib untuk dapat PTF
Hardware maintenance	Tahunan	Menentukan waktu tanggap
Perangkat lunak pihak ketiga	Beragam	Sering terikat serial mesin
Aplikasi core banking	Kontrak vendor	Biasanya komponen terbesar

Struktur lisensi IBM berubah dari waktu ke waktu dan berbeda antar wilayah. Verifikasi rincian yang berlaku dengan perwakilan IBM atau mitra bisnis Anda sebelum menyusun anggaran — angka dari dokumen lama sering menyesatkan.

114.2 Memeriksa apa yang terpasang

```
/* Licensed program yang terpasang */
DSPSFWRSC OUTPUT(*PRINT)

/* Informasi lisensi per produk */
WRKLICINF

/* Detail satu produk */
DSPLICKEY PRDID(5770SS1)
```

```
-- Produk terpasang dan statusnya
SELECT PRODUCT_ID, PRODUCT_OPTION, RELEASE_LEVEL,
       LOAD_ID, SYMBOLIC_LOAD_STATE,
       LOAD_ERROR_INDICATOR, TEXT_DESCRIPTION
  FROM QSYS2.SOFTWARE_PRODUCT_INFO
 ORDER BY PRODUCT_ID, PRODUCT_OPTION;

-- Informasi kunci lisensi
SELECT PRODUCT_ID, FEATURE, LICENSE_TERM,
       USAGE_LIMIT, CURRENT_USAGE, PEAK_USAGE,
       EXPIRATION_DATE, USAGE_TYPE
  FROM QSYS2.LICENSE_INFO
 ORDER BY PRODUCT_ID;
```

Kueri kedua layak dijadwalkan bulanan. Kolom `PEAK_USAGE` menunjukkan apakah Anda mendekati batas lisensi — informasi yang jauh lebih baik diketahui sekarang daripada saat pengguna ditolak masuk.

114.3 Produk yang terpasang tetapi tidak dipakai

```
-- Bandingkan produk terpasang dengan yang benar-benar dipakai
SELECT PRODUCT_ID, TEXT_DESCRIPTION
  FROM QSYS2.SOFTWARE_PRODUCT_INFO
 WHERE SYMBOLIC_LOAD_STATE = 'INSTALLED'
 ORDER BY PRODUCT_ID;
```

Setiap produk berlisensi yang terpasang tetapi tidak dipakai adalah biaya pemeliharaan tahunan yang terbuang. Tinjauan tahunan atas daftar ini hampir selalu menemukan sesuatu — produk yang dipasang untuk proyek yang batal lima tahun lalu, atau fitur yang digantikan oleh fungsi bawaan.

Sebelum mencabut, pastikan benar-benar tidak dipakai. Periksa objek yang bergantung padanya, dan

jalankan periode pengamatan sebelum pencabutan permanen.

💰 Bab 115. Menyusun Anggaran

115.1 Siklus anggaran tahunan

Bulan sebelum tahun anggaran	Kegiatan
6	Kumpulkan data kapasitas dan tren
5	Proyeksikan kebutuhan 3 tahun
4	Minta penawaran indikatif dari vendor
3	Susun usulan dengan opsi
2	Presentasi dan pembahasan
1	Revisi dan finalisasi

Kumpulkan data enam bulan sebelumnya, bukan saat diminta. Usulan anggaran yang disusun dalam satu minggu selalu terlihat seperti tebakan, karena memang begitu.

115.2 Menyusun argumen

Ada tiga jenis pengeluaran, dan masing-masing menuntut argumen berbeda.

Jenis	Argumen yang bekerja
Wajib	Konsekuensi bila tidak dilakukan, dalam bahasa risiko
Kapasitas	Proyeksi berbasis data, dengan titik kehabisan yang jelas
Perbaikan	Nilai yang dihasilkan, dalam angka

Contoh untuk pengeluaran wajib. Bukan: "kami perlu memperpanjang maintenance". Melainkan: "tanpa perpanjangan maintenance, kami tidak dapat menerima PTF keamanan. Tiga PTF keamanan terbit kuartal lalu, dua di antaranya memperbaiki kerentanan yang dapat dieksploitasi dari jaringan. Ini akan menjadi temuan audit dan pelanggaran ketentuan manajemen risiko TI."

Contoh untuk kapasitas. Bukan: "disk hampir penuh". Melainkan: "pemakaian ASP naik dari 62% menjadi 79% dalam 14 bulan, laju rata-rata 1,2% per bulan. Pada laju ini kami mencapai ambang kritis 85% pada bulan kelima tahun depan. Lead time pengadaan enam bulan. Keputusan harus diambil kuartal ini."

Perbedaan keduanya adalah angka dan tanggal. Manajemen tidak menolak permintaan yang datang dengan tanggal habis waktu yang jelas.

115.3 Data yang mendukung usulan

```
-- Tren kapasitas 24 bulan untuk lampiran usulan
SELECT VARCHAR_FORMAT(WAKTU, 'YYYY-MM') AS bulan,
       DECIMAL(AVG(CASE WHEN NAMA = 'ASP_PERSEN' THEN NILAI END), 5, 1)
         AS asp_rata,
       DECIMAL(AVG(CASE WHEN NAMA = 'CPU_PERSEN' THEN NILAI END), 5, 1)
         AS cpu_rata,
       DECIMAL(MAX(CASE WHEN NAMA = 'CPU_PERSEN' THEN NILAI END), 5, 1)
         AS cpu_puncak
  FROM MONITOR.METRIK
 WHERE WAKTU > CURRENT DATE - 24 MONTHS
 GROUP BY VARCHAR_FORMAT(WAKTU, 'YYYY-MM')
 ORDER BY bulan;
```

```
-- Proyeksi sederhana: kapan ambang tercapai
WITH tren AS (
  SELECT VARCHAR_FORMAT(WAKTU, 'YYYY-MM') AS bulan,
         AVG(NILAI) AS nilai
    FROM MONITOR.METRIK
   WHERE NAMA = 'ASP_PERSEN' AND WAKTU > CURRENT DATE - 12 MONTHS
   GROUP BY VARCHAR_FORMAT(WAKTU, 'YYYY-MM')
),
laju AS (
  SELECT (MAX(nilai) - MIN(nilai)) / (COUNT(*) - 1) AS per_bulan,
         MAX(nilai) AS sekarang
    FROM tren
)
SELECT DECIMAL(sekarang, 5, 1) AS persen_sekarang,
       DECIMAL(per_bulan, 5, 2) AS kenaikan_per_bulan,
       DECIMAL((85 - sekarang) / NULLIF(per_bulan, 0), 5, 1)
         AS bulan_sampai_kritis
  FROM laju;
```

Kueri terakhir menghasilkan satu angka yang bisa diletakkan di halaman pertama usulan: berapa bulan tersisa sebelum ambang kritis tercapai.

115.4 Menyajikan opsi

Usulan dengan satu pilihan terasa seperti tuntutan. Usulan dengan tiga pilihan terasa seperti nasihat.

Opsi	Isi	Konsekuensi
Minimal	Hanya yang wajib	Risiko tetap, kapasitas habis dalam 8 bulan
Disarankan	Wajib + kapasitas	Aman 3 tahun, risiko utama tertangani
Lengkap	Ditambah perbaikan struktural	Aman 5 tahun, beban operasional turun

Sertakan rekomendasi yang jelas. Menyajikan tiga opsi tanpa menyebut mana yang Anda sarankan memindahkan keputusan teknis ke orang yang tidak punya dasar teknis untuk mengambilnya.

115.5 Biaya yang sering terlupakan

Pos	Kenapa terlupakan
Lisensi pihak ketiga terikat serial	Baru ketahuan saat mesin diganti
Biaya migrasi dan jasa	Dianggap bagian dari pembelian
Pelatihan tim untuk teknologi baru	Dianggap bisa belajar sendiri
Perpanjangan dukungan aplikasi	Kontrak vendor terpisah
Kapasitas lingkungan non-produksi	Hanya produksi yang dihitung
Media backup dan penyimpanan offsite	Biaya berulang yang kecil tapi tetap
Biaya jaringan untuk replikasi DR	Bandwidth tidak gratis
Waktu tim selama proyek	Orang yang sama tetap harus operasi harian

Baris terakhir adalah yang paling sering menyebabkan proyek meleset dari jadwal. Tim yang dibebani proyek besar tanpa tambahan tenaga akan mengorbankan salah satu: proyek terlambat, atau operasi harian memburuk. Sampaikan ini di depan, bukan setelah terjadi.

115.6 Menghitung biaya tidak bertindak

Argumen yang paling kuat sering bukan biaya melakukan, melainkan biaya tidak melakukan.

Skenario	Cara memperkirakan
Layanan berhenti sehari	Volume transaksi harian, biaya penanganan manual, dampak reputasi
Temuan audit berulang	Biaya remediasi, potensi sanksi
Kehilangan data satu hari	Biaya rekonstruksi, kewajiban pelaporan
Kehilangan satu ahli kunci	Biaya rekrutmen, waktu sampai produktif, risiko selama kosong
EOD gagal sekali sebulan	Biaya lembur, keterlambatan layanan, kepercayaan

Angka-angka ini tidak perlu presisi. Yang penting adalah urutan besarannya, dan bahwa perbandingannya dibuat secara eksplisit alih-alih dibiarkan menjadi asumsi tak tertulis.

💰 Bab 116. Total Biaya Kepemilikan

116.1 Komponen lima tahun

Kategori	Sifat	Porsi khas
Perangkat keras	Satu kali di awal	Menengah
Lisensi sistem operasi	Melekat perangkat keras	Menengah
Pemeliharaan tahunan	Berulang	Besar kumulatif
Aplikasi core banking	Kontrak berjalan	Terbesar
Tenaga kerja	Berulang	Besar
Fasilitas dan daya	Berulang	Kecil sampai menengah
Jaringan	Berulang	Kecil

Dalam perhitungan lima tahun, biaya berulang hampir selalu melampaui biaya perangkat keras awal. Ini sering mengejutkan orang yang terbiasa membandingkan harga mesin.

116.2 Membandingkan dengan alternatif

Ketika muncul usulan mengganti platform, perbandingan yang jujur harus memasukkan:

1. Biaya penulisan ulang aplikasi, bukan hanya biaya platform baru
2. Durasi proyek dan biaya menjalankan dua sistem selama transisi
3. Risiko kegagalan proyek, yang di perbankan berarti biaya sangat besar
4. Biaya pelatihan ulang seluruh tim
5. Kehilangan pengetahuan yang tertanam dalam kode lama
6. Biaya dukungan platform baru, yang belum tentu lebih murah

Perbandingan yang hanya melihat biaya lisensi tahunan platform lama versus platform baru menyesatkan, dan itulah bentuk perbandingan yang paling sering disajikan.

Ini bukan argumen untuk selalu bertahan. Ada situasi di mana penggantian adalah keputusan yang benar. Tetapi keputusan itu harus diambil dengan angka yang lengkap, bukan dengan perbandingan sebagian.

116.3 Menyiapkan angka Anda sendiri

```
-- Inventaris untuk perhitungan biaya
SELECT 'Jumlah partisi' AS pos, COUNT(*) AS nilai
  FROM QSYS2.CLUSTER_INFO
UNION ALL
SELECT 'Total penyimpanan GB',
       SUM(UNIT_STORAGE_CAPACITY) / 1073741824
  FROM QSYS2.SYSDISKSTAT
UNION ALL
SELECT 'Jumlah pengguna aktif',
       COUNT(*) FROM QSYS2.USER_INFO WHERE STATUS = '*ENABLED'
UNION ALL
SELECT 'Jumlah produk berlisensi',
       COUNT(DISTINCT PRODUCT_ID) FROM QSYS2.SOFTWARE_PRODUCT_INFO
 WHERE SYMBOLIC_LOAD_STATE = 'INSTALLED'
UNION ALL
SELECT 'Objek aplikasi',
       COUNT(*) FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'));
```

Simpan hasil kueri ini setiap tahun. Deret angkanya menjadi dasar percakapan tentang pertumbuhan yang jauh lebih meyakinkan daripada pernyataan bahwa sistem semakin besar.

📖 Bagian XXXIV — Referensi Perintah CL

Disusun per kategori tugas, bukan menurut abjad, supaya bisa dipakai saat Anda tahu apa yang ingin dilakukan tetapi lupa nama perintahnya.

Bab 117. Perintah per Kategori

117.1 Pola penamaan

Perintah CL mengikuti pola kata kerja dan objek yang konsisten. Memahami polanya berarti Anda bisa menebak nama perintah yang belum pernah dipakai.

Awalan	Arti	Contoh
`WRK`	Work with, layar interaktif	`WRKACTJOB`
`DSP`	Display, tampilkan saja	`DSPLIB`
`CRT`	Create	`CRTLIB`
`CHG`	Change	`CHGJOB`
`DLT`	Delete	`DLTF`
`RTV`	Retrieve, ambil ke variabel	`RTVJOBA`
`ADD`	Tambah entri	`ADDLIBLE`
`RMV`	Hapus entri	`RMVLIBLE`
`STR`	Start	`STRSBS`
`END`	End	`ENDJOB`
`SAV`	Save	`SAVLIB`
`RST`	Restore	`RSTLIB`
`CPY`	Copy	`CPYF`
`MOV`	Move	`MOVOBJ`

Awalan	Arti	Contoh
RNM	Rename	RNMOBJ
CHK	Check	CHKOBJ
PRT	Print	PRTCMNTRC
SND	Send	SNDMSG
RCV	Receive	RCVMSG

Singkatan objek yang sering muncul: OBJ objek, LIB library, PF physical file, LF logical file, SBS subsystem, JOBQ job queue, OUTQ output queue, USRPRF user profile, JRN journal, SPLF spooled file, MSG message, PGM program.

Gabungkan keduanya dan Anda mendapat perintah yang benar tanpa mencari: menampilkan wewenang objek adalah DSPOBJAUT, mengubah deskripsi job adalah CHGJOBD, menyimpan objek yang berubah adalah SAVCHGOBJ.

117.2 Objek dan library

Perintah	Fungsi
CRTLIB / DLTLIB	Buat dan hapus library
CHGLIB	Ubah atribut library
DSPLIB	Tampilkan isi library
CLRLIB	Kosongkan library
DSPOBJD	Deskripsi objek, bisa ke outfile
CHGOBJD	Ubah deskripsi objek
MOVOBJ	Pindahkan objek antar library
RNMOBJ	Ubah nama objek
DLTOBJ	Hapus objek
CHKOBJ	Periksa keberadaan dan wewenang
ALCOBJ / DLCOBJ	Alokasikan dan lepaskan objek
WRKOBJLCK	Siapa mengunci objek
DSPOBJAUT	Wewenang atas objek
GRTOBJAUT / RVKOBJAUT	Beri dan cabut wewenang
CHGOBJOWN	Ubah pemilik objek
CHGOBJAUD	Atur audit objek
DSPPGMREF	Objek yang dipakai program
RTVDDSSRC	Ambil kembali sumber DDS
RTVCLSRC	Ambil kembali sumber CL
CHKOBJITG	Periksa integritas objek

117.3 Library list

Perintah	Fungsi
DSPLIBL	Tampilkan library list job
CHGLIBL	Ganti seluruh bagian pengguna
ADDLIBLE	Tambah satu library
RMVLIBLE	Hapus satu library

Perintah	Fungsi
CHGCURLIB	Ubah current library
EDTLIBL	Sunting library list interaktif
CHGSYSLIBL	Ubah bagian sistem, butuh wewenang khusus

117.4 Job dan work management

Perintah	Fungsi
WRKACTJOB	Job aktif
WRKUSRJOB	Job milik pengguna tertentu
WRKSBMJOB	Job yang dikirim dari sesi ini
WRKJOB	Detail satu job
DSPJOB	Tampilkan atribut job
CHGJOB	Ubah atribut job berjalan
ENDJOB	Akhiri job
HLDJOB / RLSJOB	Tahan dan lepaskan job
SBMJOB	Kirim job batch
RTVJOBA	Ambil atribut job ke variabel
DSPJOBLOG	Job log
WRKJOBQ	Job queue
CRTJOBQ / DLTJOBQ	Buat dan hapus job queue
HLDJOBQ / RLSJOBQ	Tahan dan lepaskan job queue
CLRJOBQ	Kosongkan job queue
WRKSBS	Subsystem aktif
STRSBS / ENDSBS	Mulai dan hentikan subsystem
CRTSBSD / CHGSBSD	Buat dan ubah deskripsi subsystem
DSPSBSD	Tampilkan deskripsi subsystem
ADDJOBQE / CHGJOBQE	Kaitkan job queue ke subsystem
ADDRTGE / CHGRTGE	Routing entry
ADDPJE / CHGPJE	Prestart job entry
ADDWSE / ADDCMNE	Workstation dan communication entry
CRTJOBD / CHGJOBD	Job description
CRTCLS / CHGCLS	Class, prioritas dan time slice
WRKSYSSTS	Status sistem dan pool
CHGSHRPOOL	Ubah shared pool
WRKSHRPOOL	Kelola shared pool
WRKSYSACT	Aktivitas sistem waktu nyata

117.5 Penjadwalan

Perintah	Fungsi
WRKJOBSCDE	Kelola jadwal job
ADDJOBSCDE	Tambah jadwal
CHGJOBSCDE	Ubah jadwal
RMVJOBSCDE	Hapus jadwal

Perintah	Fungsi
HLDJOBSCDE / RLSJOBSCDE	Tahan dan lepaskan jadwal
DLYJOB	Tunda job

117.6 File dan basis data

Perintah	Fungsi
CRTPF / CRTLF	Buat physical dan logical file
CRTSRCPF	Buat source physical file
ADDPFM / RMVM	Tambah dan hapus member
CLRPFM	Kosongkan member
CPYF	Salin file
DSPPFM	Tampilkan isi member
DSPFD	Deskripsi file
DSPFFD	Deskripsi field
RGZPFM	Reorganisasi member
RUNQRY	Jalankan kueri cepat
RUNSQL	Jalankan satu statement SQL
RUNSQLSTM	Jalankan skrip SQL
STRSQL	Sesi SQL interaktif
CPYTOIMPF / CPYFRMIMPF	Ekspor dan impor berkas teks
CPYTOSTMF / CPYFRMSTMF	Salin antara member dan IFS
STRJRNPF / ENDJRNPF	Mulai dan hentikan journaling
CRTJRN / CRTJRNRCV	Buat journal dan receiver
WRKJRN / WRKJRNA	Kelola journal dan atributnya
DSPJRN	Tampilkan entri journal
APYJRNCHG / RMVJRNCHG	Terapkan dan batalkan perubahan
CHGJRN	Ganti receiver
STRCMTCTL / ENDCMTCTL	Commitment control
WRKCMTDFN	Kelola definisi commit

117.7 Keamanan

Perintah	Fungsi
CRTUSRPRF / CHGUSRPRF	Buat dan ubah profil
DSPUSRPRF	Tampilkan profil
DLTUSRPRF	Hapus profil
WRKUSRPRF	Kelola profil
CHGUSRAUD	Atur audit per pengguna
CRTAUTL	Buat authorization list
ADDAUTLE / RMVAUTLE	Kelola entri authorization list
DSPAUTL	Tampilkan authorization list
GRTUSRAUT	Salin wewenang antar pengguna
DSPAUTUSR	Daftar pengguna berwenang
WRKSYSVAL	Kelola system value

Perintah	Fungsi
DSPSYSVAL / CHGSYSVAL	Tampilkan dan ubah system value
WRKREGINF	Exit point terdaftar
ADDEXITPGM / RMVEXITPGM	Kelola exit program
WRKFCNUSG	Kelola function usage
PRTPVTAUT	Cetak wewenang privat
PRTPUBAUT	Cetak wewenang publik
PRTUSRPRF	Cetak laporan profil
ANZDFTPWD	Cari password sama dengan nama profil

ANZDFTPWD layak dijalankan sekarang bila Anda belum pernah. Ia mencari profil yang passwordnya sama dengan nama profilnya — temuan audit klasik yang sering masih ada di sistem lama.

117.8 Backup dan pemulihan

Perintah	Fungsi
SAVSYS	Simpan sistem operasi
SAVLIB	Simpan library
SAVOBJ	Simpan objek tertentu
SAVCHGOBJ	Simpan objek yang berubah
SAV	Simpan objek IFS
SAVSECDTA	Simpan data keamanan
SAVCFG	Simpan konfigurasi
SAVDLO	Simpan document library object
RSTLIB / RSTOBJ	Pulihkan library dan objek
RST	Pulihkan objek IFS
RSTUSRPRF	Pulihkan profil pengguna
RSTAUT	Pulihkan wewenang, selalu terakhir
RSTCFG	Pulihkan konfigurasi
DSPSAVF	Tampilkan isi save file
CRTSAVF / CLRSAVF	Kelola save file
STRBKUBRM	Jalankan control group BRMS
WRKMEDBRM / WRKMEDIBRM	Kelola media dan inventaris
STRRCYBRM	Pemulihan terbantu BRMS
DSPLOGBRM	Log BRMS

117.9 Jaringan

Perintah	Fungsi
CFGTCP	Menu konfigurasi TCP/IP
STRTCP / ENDTCP	Mulai dan hentikan TCP/IP
STRTCPSVR / ENDTCPSVR	Mulai dan hentikan server
ADDTCPIFC / CHGTCPIFC	Kelola interface
STRTCPIFC / ENDTCPIFC	Aktifkan dan matikan interface
ADDTCPRTE / CHGTCPRTE	Kelola rute
CHGTCPDMN	Domain dan DNS

Perintah	Fungsi
ADDTCPHTE / CHGTCPHTE	Tabel host
NETSTAT	Status jaringan
PING	Uji jangkauan
TRACEROUTE	Telusuri jalur
CRTLINETH / CHGLINETH	Line description Ethernet
VRYCFG	Aktifkan dan matikan konfigurasi
WRKCFGSTS	Status konfigurasi
STRCMNTRC / ENDCMNTRC	Penelusuran komunikasi
PRTCMNTRC	Cetak hasil penelusuran
CHGTELNA / CHGFTPA	Atribut Telnet dan FTP

117.10 Keluaran dan cetak

Perintah	Fungsi
WRKSPLF	Kelola spooled file
WRKOUTQ	Kelola output queue
CRTOUTQ / DLTOUTQ	Buat dan hapus output queue
CHGSPLFA	Ubah atribut spooled file
HLDSPLF / RLSSPLF	Tahan dan lepaskan
DLTSPLF	Hapus spooled file
CPYSPLF	Salin ke berkas
STRPRTWTR / ENDWTR	Kelola writer
WRKWTR	Kelola writer
OVRPRTF	Override printer file
DLTOVR / DSPOVR	Kelola override

117.11 Pemeliharaan sistem

Perintah	Fungsi
PWRDWNSYS	Matikan atau restart sistem
WRKPTFGRP	PTF group
DSPPTF	PTF terpasang
LODPTF / APYPTF / RMVPTF	Kelola PTF
SNDPTFORD	Pesan PTF dari IBM
INSPTF	Pasang PTF
DSPSFWRSC	Produk terpasang
WRKLICINF	Informasi lisensi
RCLSTG	Klaim ruang penyimpanan
RCLSPLSTG	Klaim ruang spooled file
STRCLNUP / CHGCLNUP	Pembersihan otomatis
WRKDSKSTS	Status disk
STRASPBAL / ENDASPBAL	Penyeimbangan ASP
ANZOBJCVN	Analisis konversi objek

Perintah	Fungsi
DSPLOG	History log
WRKPRB	Log masalah
STRPFRCOL / ENDPFRCOL	Collection Services
CRTPFRDTA	Bangun data kinerja
STRJWMON / ENDJWMON	Job Watcher
STRDW / ENDDW	Disk Watcher

117.12 Pengembangan

Perintah	Fungsi
CRTBNDRPG	Kompilasi RPG jadi program
CRTRPGMOD	Kompilasi RPG jadi modul
CRTSQLRPGI	Kompilasi RPG dengan embedded SQL
CRTBNDCL / CRTCLMOD	Kompilasi CL
CRTPGM	Bind modul jadi program
CRTSRVPGM	Buat service program
CRTBNDDIR / ADDBNDDIRE	Binding directory
DSPPGM / DSPSRVPGM	Tampilkan informasi program
DSPMOD	Tampilkan informasi modul
UPDPGM / UPDSRVPGM	Perbarui modul dalam program
STRDBG / ENDDBG	Sesi debug
STRSRVJOB / ENDSRVJOB	Layani job lain
STRSEU	Editor sumber lama
CRTDSPF / CRTPRTF	Buat display dan printer file
STRPDM	Programming Development Manager

117.13 Pesan

Perintah	Fungsi
SNDMSG	Kirim pesan ke pengguna
SNDBRKMSG	Kirim pesan interupsi
SNDPGMMSG	Kirim pesan dari program
RCVMSG	Terima pesan dalam program
SNDUSRMSG	Kirim dan tunggu jawaban
DSPMSG	Tampilkan antrean pesan
WRKMSGQ	Kelola antrean pesan
CRTMSGQ / DLTMSGQ	Buat dan hapus antrean
CHGMSGQ	Ubah antrean, atur mode
CRTMSGF / ADDMSGD	Message file dan deskripsi
DSPMSGD	Tampilkan deskripsi pesan
WRKMSGD	Kelola deskripsi pesan
MONMSG	Pantau pesan dalam CL
STRWCH / ENDWCH	Pantau kemunculan pesan
WRKWCH	Kelola watch aktif

117.14 IFS dan PASE

Perintah	Fungsi
`WRKLNK`	Kelola objek IFS
`CRTDIR` / `RMVDIR`	Buat dan hapus direktori
`CPY`	Salin objek IFS
`MOV`	Pindahkan objek IFS
`RNM`	Ubah nama
`CHGAUT`	Ubah wewenang IFS
`CHGOWN`	Ubah pemilik
`DSPLNK`	Tampilkan objek IFS
`QSH`	Shell Qshell
`CALL QP2TERM`	Terminal PASE
`STRQSH`	Mulai Qshell
`MKDIR` / `RMDIR`	Direktori dari Qshell

📖 Bab 118. Indeks IBM i Services

Daftar view dan fungsi tabel yang paling sering dipakai, dikelompokkan per keperluan. Ketersediaan sebagian bergantung pada rilis dan Technology Refresh — verifikasi di sistem Anda.

118.1 Sistem dan kapasitas

Objek	Isi
`QSYS2.SYSTEM_STATUS()`	CPU, ASP, jumlah job, penyimpanan sementara
`QSYS2.SYSTEM_STATUS_INFO`	View atas informasi status sistem
`QSYS2.MEMORY_POOL()`	Pool, ukuran, faulting, activity level
`QSYS2.MEMORY_POOL_INFO`	Informasi pool
`QSYS2.SYSDISKSTAT`	Statistik per unit disk
`QSYS2.ASP_INFO`	Informasi ASP
`QSYS2.SYSTEM_VALUE_INFO`	Seluruh system value
`QSYS2.GROUP_PTF_INFO`	Level PTF group
`QSYS2.PTF_INFO`	PTF individual
`QSYS2.SOFTWARE_PRODUCT_INFO`	Produk berlisensi terpasang
`QSYS2.LICENSE_INFO`	Batas dan pemakaian lisensi

118.2 Job dan work management

Objek	Isi
`QSYS2.ACTIVE_JOB_INFO()`	Job aktif dengan detail
`QSYS2.JOB_INFO()`	Job termasuk yang di antrean
`QSYS2.JOBLOG_INFO()`	Isi job log sebagai tabel
`QSYS2.JOB_QUEUE_INFO`	Job queue dan isinya
`QSYS2.SUBSYSTEM_INFO`	Subsystem dan statusnya
`QSYS2.SUBSYSTEM_POOL_INFO`	Pool per subsystem
`QSYS2.SCHEDULED_JOB_INFO`	Jadwal job

Objek	Isi
QSYS2.OUTPUT_QUEUE_ENTRIES	Spooled file
QSYS2.OUTPUT_QUEUE_INFO	Output queue
QSYS2.HISTORY_LOG_INFO()	History log sebagai tabel
QSYS2.MESSAGE_QUEUE_INFO	Isi antrean pesan
QSYS2.MESSAGE_FILE_DATA	Deskripsi pesan

118.3 Objek dan penyimpanan

Objek	Isi
QSYS2.OBJECT_STATISTICS()	Objek, ukuran, pemakaian terakhir
QSYS2.IFS_OBJECT_STATISTICS()	Objek IFS
QSYS2.IFS_READ()	Baca isi berkas IFS
QSYS2.IFS_WRITE()	Tulis berkas IFS
QSYS2.OBJECT_LOCK_INFO	Penguncian objek
QSYS2.OBJECT_PRIVILEGES	Wewenang atas objek
QSYS2.LIBRARY_INFO()	Informasi library termasuk ukuran
QSYS2.DATA_AREA_INFO	Isi data area
QSYS2.DATA_QUEUE_INFO	Data queue
QSYS2.DATA_QUEUE_ENTRIES()	Isi data queue

118.4 Basis data

Objek	Isi
QSYS2.SYSTABLES	Daftar tabel
QSYS2.SYSCOLUMNS	Kolom termasuk CCSID
QSYS2.SYSINDEXES	Index
QSYS2.SYSTABLESTAT	Statistik tabel, jumlah baris dan terhapus
QSYS2.SYSPARTITIONINDEXSTAT	Statistik index termasuk pemakaian
QSYS2.SYSIXADV	Index yang disarankan optimizer
QSYS2.SYSCOLUMNSTAT	Statistik kolom
QSYS2.SYSCST	Constraint
QSYS2.SYSTRIGGERS	Trigger
QSYS2.SYSVIEWS	View
QSYS2.SYSROUTINES	Prosedur dan fungsi
QSYS2.RECORD_LOCK_INFO	Penguncian record
QSYS2.ACTIVE_QUERY_INFO	Kueri yang sedang berjalan
QSYS2.DB_TRANSACTION_INFO	Transaksi commitment control
QSYS2.JOURNAL_INFO	Journal
QSYS2.JOURNAL_RECEIVER_INFO	Receiver
QSYS2.JOURNALED_OBJECTS	Objek yang dijurnal
QSYS2.DISPLAY_JOURNAL()	Entri journal sebagai tabel

118.5 Keamanan

Objek	Isi
`QSYS2.USER_INFO`	Profil pengguna
`QSYS2.USER_INFO_BASIC`	Versi ringan, lebih cepat
`QSYS2.GROUP_PROFILE_ENTRIES`	Keanggotaan group
`QSYS2.AUTHORIZATION_LIST_INFO`	Authorization list
`QSYS2.AUTHORIZATION_LIST_USER_INFO`	Entri authorization list
`QSYS2.FUNCTION_INFO`	Fungsi yang dikendalikan
`QSYS2.FUNCTION_USAGE`	Siapa berwenang atas fungsi
`QSYS2.EXIT_PROGRAM_INFO`	Exit program terdaftar
`QSYS2.CERTIFICATE_INFO()`	Sertifikat di DCM
`QSYS2.SECURITY_INFO`	Ringkasan konfigurasi keamanan
`SYSTOOLS.GROUP_PTF_CURRENCY`	Perbandingan PTF dengan level IBM

118.6 Jaringan

Objek	Isi
`QSYS2.NETSTAT_INFO`	Koneksi dan port
`QSYS2.NETSTAT_JOB_INFO`	Koneksi beserta job pemiliknya
`QSYS2.NETSTAT_INTERFACE_INFO`	Interface dan statistiknya
`QSYS2.NETSTAT_ROUTE_INFO`	Tabel rute
`QSYS2.TCPIP_INFO`	Konfigurasi TCP/IP
`QSYS2.TCPIP_HOST_TABLE`	Tabel host
`QSYS2.SERVER_SHARE_INFO`	Server dan share
`QSYS2.ENVIRONMENT_VARIABLE_INFO`	Variabel lingkungan

118.7 Prosedur yang berguna

Prosedur	Fungsi
`QSYS2.QCMDEXC()`	Jalankan perintah CL dari SQL
`QSYS2.OVERRIDE_QAQQINI()`	Setel opsi kueri untuk job
`QSYS2.ENSURE_STATISTICS()`	Perbarui statistik tabel
`QSYS2.DUMP_PLAN_CACHE()`	Buang plan cache ke tabel
`QSYS2.GENERATE_SQL()`	Hasilkan DDL dari objek
`QSYS2.GENERATE_SQL_OBJECTS()`	DDL untuk banyak objek
`SYSTOOLS.HTTPGETCLOB()`	Panggilan HTTP GET
`SYSTOOLS.HTTPPOSTCLOB()`	Panggilan HTTP POST
`SYSTOOLS.SPLIT()`	Pecah teks jadi baris
`SYSTOOLS.LPRINTF()`	Tulis ke job log dari SQL

118.8 Pola pemakaian yang sering berguna

```
-- Menjalankan perintah CL dari sesi SQL
CALL QSYS2.QCMDEXC('ADDLIBLE LIB(APPPGM) POSITION(*FIRST)');

-- Menulis catatan ke job log untuk penelusuran
CALL SYSTOOLS.LPRINTF('Mulai proses rekonsiliasi');

-- Membaca berkas IFS sebagai tabel
SELECT LINE_NUMBER, LINE
  FROM TABLE(QSYS2.IFS_READ(
         PATH_NAME => '/interface/in/kontrol.txt'))
 ORDER BY LINE_NUMBER;

-- Memecah teks berdelimiter jadi baris
SELECT ORDINAL_POSITION, ELEMENT
  FROM TABLE(SYSTOOLS.SPLIT('A,B,C,D', ','));
```

118.9 Membangun perpustakaan kueri sendiri

Simpan kueri yang sering dipakai sebagai view bernama, bukan sebagai potongan yang disalin berulang.

```
CREATE SCHEMA TOOLS;

CREATE OR REPLACE VIEW TOOLS.V_JOB_BERMASALAH AS
SELECT JOB_NAME, AUTHORIZATION_NAME, SUBSYSTEM,
       JOB_STATUS, FUNCTION, ELAPSED_TIME,
       ELAPSED_CPU_PERCENTAGE
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 WHERE JOB_STATUS IN ('MSGW','LCKW','DEQW')
    OR ELAPSED_CPU_PERCENTAGE > 20;

CREATE OR REPLACE VIEW TOOLS.V_PROFIL_BERISIKO AS
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       PREVIOUS_SIGNON,
       DAYS(CURRENT DATE) - DAYS(DATE(PREVIOUS_SIGNON)) AS hari_diam
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES <> ''
   AND STATUS = '*ENABLED';

CREATE OR REPLACE VIEW TOOLS.V_TABEL_PERLU_REORG AS
SELECT TABLE_SCHEMA, TABLE_NAME, NUMBER_ROWS, NUMBER_DELETED_ROWS,
       DECIMAL(NUMBER_DELETED_ROWS * 100.0 /
               NULLIF(NUMBER_ROWS + NUMBER_DELETED_ROWS, 0), 5, 1)
         AS persen_terhapus
  FROM QSYS2.SYSTABLESTAT
 WHERE NUMBER_DELETED_ROWS > 50000;
```

Setelah beberapa bulan, schema `TOOLS` menjadi aset tim yang nilainya melampaui kueri individual di dalamnya: ia merekam cara tim Anda memandang sistemnya.

Bagian XXXV — Penutup

Bab 119. Peta Jalan Pengembangan Diri

119.1 Tiga tahun pertama

Tahun	Fokus	Penanda kemampuan
1	Operasi dan diagnosis dasar	Bisa jaga malam dengan cadangan
2	Kinerja, keamanan, pemulihan	Bisa memimpin penyelesaian insiden
3	Arsitektur dan proyek	Bisa memimpin upgrade atau migrasi

119.2 Keterampilan yang nilainya naik

Keterampilan	Kenapa naik
SQL untuk administrasi	Menggantikan hampir seluruh perintah `DSP`
Keamanan dan audit	Tuntutan regulator terus meningkat
Integrasi dan API	IBM i makin jadi mesin di belakang kanal
Otomasi dan kendali versi	Ekspektasi tata kelola berubah
Kemampuan menjelaskan ke non-teknis	Menentukan apakah Anda dapat anggaran
Pemahaman domain perbankan	Membedakan teknisi dari mitra bisnis

Baris terakhir sering diremehkan. System Engineer yang memahami mengapa akrual tidak boleh diulang jauh lebih berharga daripada yang hanya tahu cara menjalankan ulang job.

119.3 Sumber belajar

Sumber	Untuk apa
IBM Documentation	Rujukan resmi, selalu terkini
IBM Redbooks	Panduan mendalam per topik
MC Press Online	Artikel praktis
IT Jungle	Berita dan analisis platform
RPGPGM.com	Tutorial teknis
COMMON	Konferensi dan komunitas
Sistem uji Anda sendiri	Yang paling efektif

Baris terakhir bukan basa-basi. Membaca tentang `APYJRNCHG` dan pernah menjalankannya dua kali adalah dua tingkat kemampuan yang berbeda.

Bab 120. Penutup

Buku ini dibuka dengan pernyataan bahwa IBM i bertahan bukan karena inersia. Setelah seratus dua puluh bab, saya ingin menutup dengan alasan yang lebih spesifik.

Platform ini bertahan karena ia dirancang oleh orang yang menganggap integritas data sebagai kewajiban, bukan fitur. Single-level storage, arsitektur berbasis objek, journaling terintegrasi, dan commitment control bukan tambahan yang ditempelkan kemudian — semuanya ada sejak awal, dan itulah sebabnya sistem yang memindahkan uang masih berjalan di atasnya empat dekade kemudian.

Tugas seorang System Engineer di sini bukan sekadar menjaga mesin tetap hidup. Tugasnya adalah menjaga agar sifat-sifat itu tetap utuh ketika sistem berubah: memastikan tabel baru tetap dijurnal, memastikan transaksi tetap atomik, memastikan pemulihan masih mungkin, memastikan jejak audit masih lengkap. Sebagian besar isi buku ini, bila diringkas, adalah tentang itu.

Tiga hal yang saya harap tertinggal setelah buku ini ditutup:

Kegagalan yang berteriak lebih baik daripada kegagalan yang diam. Sistem yang berhenti dengan pesan jelas menyelamatkan Anda; sistem yang terus berjalan dengan data salah menghancurkan Anda perlahan. Setiap kali Anda menulis kode atau merancang proses, tanyakan apa yang terjadi bila asumsinya keliru.

Yang tidak diuji tidak ada. Backup yang tidak pernah direstore bukan backup. Runbook yang tidak pernah dijalankan orang lain bukan runbook. Prosedur DR yang tidak pernah dilatih bukan prosedur, melainkan harapan tertulis.

Pengetahuan yang hanya ada di kepala satu orang adalah risiko, bukan aset. Di platform dengan sedikit praktisi, ini bukan soal kenyamanan melainkan soal keberlangsungan. Tuliskan, ajarkan, rotasikan.

Selebihnya adalah detail — dan detail itulah yang mengisi halaman-halaman sebelumnya, tersedia untuk dibuka ketika Anda membutuhkannya pada pukul tiga pagi.

🔍 Bagian XXXVI — Studi Kasus Lanjutan

Sepuluh kasus tambahan, disusun dengan struktur yang sama seperti Bagian XXIV. Bagian ini sengaja diperluas karena studi kasus adalah bagian yang paling sering dibuka orang, dan pola yang dikenali dari kasus orang lain menghemat berjam-jam penyelidikan sendiri.

📊 Bab 121. Kasus Penyimpanan dan Kapasitas

121.1 Kasus 11 — ASP penuh dalam semalam

Gejala. Pemakaian ASP naik dari 71% menjadi 94% dalam delapan jam. Tidak ada penambahan data bisnis yang menjelaskan.

Jalan buntu. Tim mencari tabel yang membesar. Seluruh tabel aplikasi normal.

Diagnosis yang benar. Yang tumbuh bukan tabel, melainkan sesuatu di luar library aplikasi.

```
SELECT OBJLONGSCHEMA AS library,
       SUM(OBJSIZE) / 1048576 AS mb,
       COUNT(*) AS jml_objek
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'))
 GROUP BY OBJLONGSCHEMA
 ORDER BY mb DESC
 FETCH FIRST 10 ROWS ONLY;

-- Penyimpanan sementara per job
SELECT JOB_NAME, AUTHORIZATION_NAME, TEMPORARY_STORAGE,
       FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY TEMPORARY_STORAGE DESC
 FETCH FIRST 20 ROWS ONLY;
```

Satu job menahan 340 GB penyimpanan sementara.

Akar masalah. Seorang analis menjalankan kueri pelaporan tanpa kondisi join yang benar, menghasilkan hasil kali silang antara dua tabel besar. Optimizer membangun file sementara raksasa.

Perbaikan. Job diakhiri, penyimpanan kembali. Jangka panjang: `QUERY_TIME_LIMIT` dipasang di QAQQINI untuk lingkungan pelaporan, dan akses kueri ad hoc dipindahkan ke partisi MIS.

Pelajaran. Pemakaian ASP yang melonjak cepat hampir selalu penyimpanan sementara, bukan data permanen. Periksa `TEMPORARY_STORAGE` lebih dulu sebelum mencari tabel yang membesar.

121.2 Kasus 12 — Journal receiver memenuhi ASP

Gejala. ASP journal naik terus meskipun volume transaksi tetap.

Diagnosis.

```
SELECT JOURNAL_LIBRARY, JOURNAL_NAME, RECEIVER_NAME,
       RECEIVER_SIZE / 1048576 AS mb, ATTACHED_TIMESTAMP, STATUS
  FROM QSYS2.JOURNAL_RECEIVER_INFO
 ORDER BY RECEIVER_SIZE DESC;
```

Puluhan receiver berstatus terlepas tetapi belum dihapus, yang tertua berusia empat belas bulan.

Akar masalah. Journal dibuat dengan `DLTRCV(*NO)` — yang benar, karena receiver adalah bahan pemulihan. Tetapi proses pembersihan yang seharusnya menghapus receiver setelah tersimpan di backup tidak pernah dibuat.

Perbaikan. Prosedur terjadwal yang memverifikasi receiver sudah tersimpan di BRMS, lalu menghapusnya. Verifikasi lebih dulu, hapus kemudian — tidak pernah sebaliknya.

```
-- Receiver yang sudah aman dihapus
SELECT r.RECEIVER_LIBRARY, r.RECEIVER_NAME,
       r.RECEIVER_SIZE / 1048576 AS mb
  FROM QSYS2.JOURNAL_RECEIVER_INFO r
 WHERE r.STATUS = 'DETACHED'
   AND r.ATTACHED_TIMESTAMP < CURRENT DATE - 35 DAYS
   AND EXISTS (SELECT 1 FROM QUSRBRM.SAVE_HISTORY s
               WHERE s.OBJECT_NAME = r.RECEIVER_NAME);
```

Pelajaran. `DLTRCV(*NO)` memindahkan tanggung jawab pembersihan kepada Anda. Pilihan itu benar, tetapi tidak lengkap tanpa prosedur pendampingnya.

121.3 Kasus 13 — Backup makin lama setiap minggu

Gejala. Backup mingguan tumbuh dari 4 jam menjadi 9 jam dalam setahun, sementara data hanya tumbuh 18%.

Diagnosis.

```
-- Objek terbanyak per library, bukan yang terbesar
SELECT OBJLONGSCHEMA, COUNT(*) AS jml_objek,
       SUM(OBJSIZE) / 1048576 AS mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'))
 GROUP BY OBJLONGSCHEMA
 ORDER BY jml_objek DESC
 FETCH FIRST 10 ROWS ONLY;
```

Satu library kerja berisi 1,9 juta objek berukuran kecil.

Akar masalah. Proses harian membuat objek sementara di library permanen dan tidak pernah menghapusnya. Ukuran totalnya kecil, tetapi backup memproses objek satu per satu, sehingga jumlah objek menentukan durasi lebih dari ukuran total.

Perbaikan. Objek sementara dipindahkan ke QTEMP, dan pembersihan terjadwal untuk yang sudah terlanjur ada.

Pelajaran. Durasi backup berkorelasi dengan jumlah objek, bukan hanya ukuran total. Pantau keduanya.

Bab 122. Kasus Aplikasi dan Data

122.1 Kasus 14 — Laporan bulanan berbeda dari laporan harian

Gejala. Total yang dilaporkan bulanan tidak sama dengan jumlah laporan harian sepanjang bulan itu.

Diagnosis. Bandingkan definisi kedua laporan. Keduanya memakai kueri berbeda yang ditulis oleh orang berbeda pada waktu berbeda.

```
-- Harian: memakai tanggal transaksi
SELECT SUM(NOMINAL) FROM DTALIB.TRXHIST
 WHERE TGL_TRX = CURRENT DATE;

-- Bulanan: memakai tanggal buku
SELECT SUM(NOMINAL) FROM DTALIB.TRXHIST
 WHERE TGL_BUKU BETWEEN :awal AND :akhir;
```

Akar masalah. Transaksi yang masuk setelah cut-off punya tanggal transaksi hari ini tetapi tanggal buku hari berikutnya. Laporan harian menghitungnya di hari ini, laporan bulanan di bulan berikutnya bila jatuh di akhir bulan.

Perbaikan. Seluruh pelaporan keuangan distandarkan memakai tanggal buku. Definisi dibungkus sebagai view bersama sehingga tidak ada lagi dua kueri yang seharusnya sama.

Pelajaran. Ketika dua laporan seharusnya cocok tetapi tidak, periksa definisinya sebelum mencurigai datanya. Perbedaan biasanya ada di kueri, bukan di tabel.

122.2 Kasus 15 — Pembulatan yang menumpuk

Gejala. Selisih GL sebesar Rp 340 pada akhir bulan. Kecil, tetapi berulang setiap bulan dengan nilai berbeda.

Diagnosis.

```
-- Cari transaksi dengan pecahan yang mencurigakan
SELECT NO_REF, SUM(DEBET) AS d, SUM(KREDIT) AS k,
       SUM(DEBET) - SUM(KREDIT) AS selisih
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU BETWEEN :awal AND :akhir
 GROUP BY NO_REF
HAVING SUM(DEBET) <> SUM(KREDIT);
```

Ribuan transaksi dengan selisih satu rupiah, sebagian positif sebagian negatif, totalnya Rp 340.

Akar masalah. Program menghitung pajak sebagai persentase lalu membulatkan, dan membukukan sisi debet dan kredit dengan pembulatan terpisah. Ketika keduanya dibulatkan sendiri-sendiri, hasilnya kadang berbeda satu rupiah.

Perbaikan. Hitung satu sisi, lalu turunkan sisi lain dari hasil pembulatan sisi pertama, bukan menghitung ulang.

```
**free
// Salah: dua pembulatan terpisah
wDebet  = %dech(wPokok * wRate: 17: 2);
wKredit = %dech(wPokok * wRate: 17: 2);   // bisa berbeda bila rumusnya beda

// Benar: hitung sekali, turunkan
wDebet  = %dech(wPokok * wRate: 17: 2);
wKredit = wDebet;
```

Pelajaran. Di keuangan, dua perhitungan yang seharusnya menghasilkan angka sama harus dihitung sekali dan disalin, bukan dihitung dua kali. Selisih satu rupiah yang dianggap kecil adalah gejala kesalahan sistematis.

122.3 Kasus 16 — Data hilang setelah perubahan struktur

Gejala. Setelah penambahan kolom pada tabel besar, sebagian kolom keterangan menjadi kosong.

Diagnosis. Perbandingan dengan backup sebelum perubahan menunjukkan data memang hilang, bukan sekadar tidak tampil.

Akar masalah. Perubahan dilakukan dengan pola tabel bayangan. Perintah `INSERT INTO baru SELECT * FROM lama` memakai urutan kolom, dan kolom baru disisipkan di tengah pada tabel bayangan. Akibatnya nilai bergeser satu kolom, dan kolom terakhir kehilangan isinya.

Perbaikan. Restore dari backup, ulangi dengan menyebut nama kolom secara eksplisit.

```
-- Berbahaya
INSERT INTO DTALIB.REKENING_BARU SELECT * FROM DTALIB.REKENING;

-- Aman
INSERT INTO DTALIB.REKENING_BARU
  (NO_REK, NAMA, KD_PRODUK, SALDO, STATUS, TGL_BUKA)
SELECT NO_REK, NAMA, KD_PRODUK, SALDO, STATUS, TGL_BUKA
  FROM DTALIB.REKENING;
```

Pelajaran. `SELECT *` dalam operasi migrasi data adalah kebiasaan berbahaya. Sebutkan kolom, selalu. Biaya mengetiknya jauh lebih murah daripada memulihkan data yang bergeser.

122.4 Kasus 17 — Nomor urut duplikat

Gejala. Dua transaksi berbeda mendapat nomor referensi sama. Terjadi tiga kali dalam sebulan, semuanya pada jam sibuk.

Diagnosis. Program pemberi nomor membaca nilai terakhir dari data area, menambah satu, lalu menulis kembali.

Akar masalah. Antara membaca dan menulis, job lain melakukan hal yang sama. Tidak ada penguncian, sehingga dua job membaca nilai yang sama.

Perbaikan. Ganti dengan mekanisme yang atomik.

```
-- Kolom identitas, dijamin unik oleh basis data
CREATE TABLE DTALIB.TRXHIST (
    NO_URUT BIGINT GENERATED ALWAYS AS IDENTITY
            (START WITH 1 INCREMENT BY 1 NO CYCLE),
    ...
);

-- Atau sequence bila nomor dibutuhkan sebelum insert
CREATE SEQUENCE DTALIB.SEQ_TRX
    START WITH 1 INCREMENT BY 1 NO CYCLE CACHE 50;

VALUES NEXT VALUE FOR DTALIB.SEQ_TRX INTO :wNoUrut;
```

Bila mekanisme berbasis data area harus dipertahankan, kuncinya harus eksklusif selama seluruh operasi baca dan tulis.

```
ALCOBJ OBJ((APPWRK/NOURUT *DTAARA *EXCL)) WAIT(30)
RTVDTAARA DTAARA(APPWRK/NOURUT) RTNVAR(&NOMOR)
CHGVAR VAR(&NOMOR) VALUE(&NOMOR + 1)
CHGDTAARA DTAARA(APPWRK/NOURUT) VALUE(&NOMOR)
DLCOBJ OBJ((APPWRK/NOURUT *DTAARA *EXCL))
```

Pelajaran. Pola baca-ubah-tulis tanpa penguncian akan gagal, dan hanya pada beban tinggi — yang berarti gagal di produksi, tidak di lingkungan uji.

⚙️ Bab 123. Kasus Sistem dan Infrastruktur

123.1 Kasus 18 — Sistem lambat setelah penambahan disk

Gejala. Setelah menambah enam unit disk untuk menambah kapasitas, kinerja justru menurun.

Diagnosis.

```
SELECT UNIT_NUMBER, PERCENT_USED, ELAPSED_PERCENT_BUSY,
       ELAPSED_IO_REQUESTS
  FROM QSYS2.SYSDISKSTAT
 ORDER BY ELAPSED_PERCENT_BUSY DESC;
```

Unit lama 68% sibuk, unit baru 3% sibuk.

Akar masalah. Disk baru ditambahkan tetapi data tidak disebarkan ulang. Data yang ada tetap di unit lama, dan unit baru hanya menerima data baru. Beban I/O tetap terkonsentrasi.

Perbaikan. Penyeimbangan ASP pada jendela akhir pekan.

```
STRASPBAL TYPE(*CAPACITY) ASP(1) TIMLMT(360)
DSPASPBAL
ENDASPBAL ASP(1)
```

Pelajaran. Menambah disk menambah kapasitas seketika, tetapi menambah throughput hanya setelah penyeimbangan. Jadwalkan penyeimbangan sebagai bagian dari prosedur penambahan disk.

123.2 Kasus 19 — Sesi terputus acak dari satu gedung

Gejala. Pengguna di satu gedung kantor mengalami sesi 5250 terputus beberapa kali sehari, acak.

Diagnosis.

```
SELECT REMOTE_ADDRESS, COUNT(*) AS jml_koneksi,
       AVG(IDLE_TIME) AS rata_idle
  FROM QSYS2.NETSTAT_JOB_INFO
 WHERE LOCAL_PORT IN (23, 992)
 GROUP BY REMOTE_ADDRESS;
```

Pemutusan selalu terjadi setelah sesi menganggur sekitar tiga puluh menit.

Akar masalah. Firewall di gedung itu memutus koneksi TCP yang menganggur lebih dari tiga puluh menit. Ini kebijakan baru yang diterapkan tim jaringan tanpa pemberitahuan.

Perbaikan. Aktifkan keepalive pada sisi IBM i agar koneksi tidak terlihat menganggur, dan koordinasikan kebijakan timeout dengan tim jaringan.

Pelajaran. Pemutusan yang berkorelasi dengan durasi menganggur hampir selalu perangkat di tengah jalur, bukan sistem di kedua ujungnya.

123.3 Kasus 20 — IPL memakan enam jam

Gejala. Setelah pemadaman listrik, IPL yang biasanya empat puluh menit memakan enam jam.

Diagnosis. SRC di panel menunjukkan kode yang berkaitan dengan klaim ruang penyimpanan, bergerak perlahan tetapi bergerak.

Akar masalah. Penghentian tidak normal memaksa sistem memeriksa dan memulihkan struktur penyimpanan. Durasinya sebanding dengan jumlah objek di sistem, dan sistem ini punya jutaan objek kecil dari kasus 13.

Perbaikan. Tidak ada yang bisa dilakukan saat itu selain menunggu. Jangka panjang: pengurangan jumlah objek, dan yang lebih penting, UPS dengan kapasitas cukup untuk penghentian terkendali.

Pelajaran. Jangan pernah melakukan IPL paksa berulang saat SRC menunjukkan progres. Setiap pengulangan memulai proses dari awal. Catat SRC, verifikasi ia bergerak, dan tunggu.

Bab 124. Pola dari Dua Puluh Kasus

124.1 Distribusi penyebab

Dari dua puluh kasus di buku ini, penyebabnya mengelompok sebagai berikut.

Kategori penyebab	Jumlah kasus
Perubahan yang tidak terkomunikasi	5
Asumsi yang tidak divalidasi dalam kode	5
Prosedur yang ada tetapi tidak lengkap	4
Konfigurasi yang benar dulu, tidak lagi sekarang	3
Kesalahan manusia yang tidak dicegah sistem	2
Kegagalan perangkat keras atau pihak luar	1

Yang menonjol: hanya satu dari dua puluh yang benar-benar kegagalan perangkat. Sembilan belas sisanya berakar pada keputusan manusia — dalam kode, dalam prosedur, atau dalam komunikasi.

124.2 Pencegahan yang berdampak paling luas

Diurutkan berdasarkan berapa banyak kasus yang akan tercegah:

1. **Validasi asumsi secara eksplisit dalam kode.** Rate tidak ditemukan, produk tidak terdaftar, panjang baris tidak sesuai — semuanya harus menghentikan proses, bukan dilewati diam-diam.
2. **Pemantauan yang mendeteksi ketiadaan, bukan hanya kegagalan.** Interface yang tidak berjalan, backup yang tidak menyimpan sebagian objek, pemantauan yang mati sendiri.
3. **Perbandingan terhadap baseline.** Durasi fase, ukuran backup, jumlah objek, distribusi kolektibilitas. Perubahan bertahap tidak memicu alarm apa pun kecuali Anda membandingkannya.
4. **Program menolak berjalan di lingkungan yang salah.** Pemeriksaan nama sistem di awal setiap program berbahaya.
5. **Batas waktu pada setiap operasi menunggu.** Tanpa pengecualian.
6. **Sebutkan kolom, jangan `SELECT *`, dalam operasi data.**
7. **Komunikasi perubahan lintas tim.** Lima kasus berakar di sini, dan tidak satu pun bisa dicegah dengan teknologi.

124.3 Cara memakai daftar ini

Ambil satu butir per triwulan. Terapkan pada seluruh sistem Anda, bukan sebagian. Catat berapa banyak temuan yang muncul.

Butir pertama saja, bila diterapkan menyeluruh pada seluruh program fase akrual dan posting, biasanya menemukan puluhan tempat di mana nilai nol dari data yang tidak ditemukan diperlakukan sama dengan nol hasil perhitungan. Setiap satu di antaranya adalah insiden yang belum terjadi.

🌙 Bagian XXXVII — Pendalaman End of Day

Bab 31 memberi kerangka EOD. Bagian ini masuk ke tingkat yang dibutuhkan ketika Anda benar-benar harus mengambil keputusan pada pukul tiga pagi.

Bab 125. Merancang Fase yang Aman Diulang

125.1 Tiga tingkat keamanan pengulangan

Tingkat	Sifat	Cara mencapainya
Idempoten alami	Menjalankan ulang menghasilkan keadaan sama	Operasi menetapkan nilai, bukan menambah
Idempoten terkendali	Aman diulang karena ada penanda	Tabel penanda memblokir pengulangan
Tidak idempoten	Pengulangan merusak data	Wajib reversal atau restore

Contoh idempoten alami. Klasifikasi kolektibilitas menetapkan nilai berdasarkan hari tunggakan. Dijalankan sepuluh kali, hasilnya sama.

```
UPDATE DTALIB.KREDIT
   SET KOLEKTIBILITAS = CASE
         WHEN HARI_TUNGGAK = 0 THEN 1
         WHEN HARI_TUNGGAK BETWEEN 1 AND 90 THEN 2
         ELSE 5 END
 WHERE STATUS = 'A';
```

Contoh tidak idempoten. Akrual menambah ke saldo. Dijalankan dua kali, bunga terbukukan dua kali.

```
UPDATE DTALIB.REKENING
   SET SALDO = SALDO + :wBunga
 WHERE NO_REK = :wNoRek;
```

Mengubah yang kedua menjadi tingkat kedua. Tulis hasil ke tabel terpisah lebih dulu, lalu posting dari tabel itu dengan penanda.

```
-- Langkah 1: hitung dan simpan, tanpa menyentuh saldo
INSERT INTO APPWRK.AKRUAL_HITUNG
  (TGL_BUKU, NO_REK, BUNGA, STATUS)
SELECT CURRENT DATE, NO_REK,
       DECIMAL(SALDO * r.RATE / 365, 17, 2), 'H'
  FROM DTALIB.REKENING k
  JOIN APPPRM.PRODUK_RATE r ON r.KD_PRODUK = k.KD_PRODUK
 WHERE k.STATUS = 'A'
   AND NOT EXISTS (SELECT 1 FROM APPWRK.AKRUAL_HITUNG a
                     WHERE a.TGL_BUKU = CURRENT DATE
                       AND a.NO_REK = k.NO_REK);

-- Langkah 2: posting hanya yang belum diposting
UPDATE DTALIB.REKENING k
   SET SALDO = SALDO + (SELECT BUNGA FROM APPWRK.AKRUAL_HITUNG a
                          WHERE a.NO_REK = k.NO_REK
                            AND a.TGL_BUKU = CURRENT DATE
                            AND a.STATUS = 'H')
 WHERE EXISTS (SELECT 1 FROM APPWRK.AKRUAL_HITUNG a
                 WHERE a.NO_REK = k.NO_REK
                   AND a.TGL_BUKU = CURRENT DATE
                   AND a.STATUS = 'H');
```

```
-- Langkah 3: tandai sudah diposting, dalam transaksi yang sama
UPDATE APPWRK.AKRUAL_HITUNG
   SET STATUS = 'P', WAKTU_POSTING = CURRENT TIMESTAMP
 WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'H';
```

Setelah dipecah begini, pengulangan langkah satu tidak menghitung ulang yang sudah ada, dan pengulangan langkah dua tidak memposting yang sudah berstatus `P`. Kegagalan di tengah bisa dilanjutkan, bukan diulang dari nol.

Syaratnya: langkah dua dan tiga harus berada dalam satu unit transaksi. Bila terpisah, kegagalan di antaranya menghasilkan saldo yang sudah bertambah tetapi penanda belum berubah — dan pengulangan akan memposting dua kali.

125.2 Reversal

Untuk fase yang sudah terlanjur berjalan salah dan restore terlalu mahal, reversal adalah jalan tengah.

```
-- Membalik akrual berdasarkan catatan yang tersimpan
UPDATE DTALIB.REKENING k
   SET SALDO = SALDO - (SELECT BUNGA FROM APPWRK.AKRUAL_HITUNG a
                         WHERE a.NO_REK = k.NO_REK
                           AND a.TGL_BUKU = :wTglBuku
                           AND a.STATUS = 'P')
 WHERE EXISTS (SELECT 1 FROM APPWRK.AKRUAL_HITUNG a
                WHERE a.NO_REK = k.NO_REK
                  AND a.TGL_BUKU = :wTglBuku
                  AND a.STATUS = 'P');

-- Tandai dibalik, jangan hapus catatannya
UPDATE APPWRK.AKRUAL_HITUNG
   SET STATUS = 'R', WAKTU_REVERSAL = CURRENT TIMESTAMP,
       ALASAN_REVERSAL = :wAlasan
 WHERE TGL_BUKU = :wTglBuku AND STATUS = 'P';

-- Entri GL penyeimbang, jangan menghapus entri asli
INSERT INTO DTALIB.GLTRX
  (TGL_BUKU, NO_REF, KD_GL, DEBET, KREDIT, KETERANGAN)
SELECT :wTglBuku, 'REV-' || :wNoRefAsli, KD_GL, KREDIT, DEBET,
       'Reversal akrual: ' || :wAlasan
  FROM DTALIB.GLTRX
 WHERE NO_REF = :wNoRefAsli;
```

Prinsip akuntansi yang menentukan bentuk kodenya. Entri yang salah tidak dihapus, melainkan dibalik dengan entri baru yang berlawanan. Jejak keduanya tetap ada. Menghapus entri GL menghancurkan jejak audit dan hampir selalu merupakan temuan serius.

Syarat reversal bisa dilakukan. Hasil perhitungan harus tersimpan. Bila program langsung menambahkan ke saldo tanpa mencatat berapa yang ditambahkan per rekening, reversal mustahil dan restore menjadi satu-satunya jalan. Inilah alasan tabel `AKRUAL_HITUNG` di atas berharga jauh melampaui fungsinya sebagai penanda.

125.3 Matriks keputusan yang diperluas

Kondisi	Rerun	Lanjut dari checkpoint	Reversal	Restore
Fase validasi gagal	Ya	—	—	—
Posting gagal sebelum commit pertama	Ya	—	—	—
Posting gagal setelah sebagian commit	—	Ya, bila ada penanda	—	Bila tidak ada penanda
Akrual gagal, hasil tersimpan	—	Ya	Ya	Alternatif
Akrual gagal, hasil tidak tersimpan	Tidak	Tidak	Tidak	Ya
Akrual selesai tetapi salah nilai	—	—	Ya	Alternatif
Klasifikasi gagal	Ya	—	—	—
GL tidak seimbang	Tidak	Tidak	Selidiki dulu	Mungkin
Pelaporan gagal	Ya	—	—	—
Date roll gagal di tengah	Tidak	Tidak	Tidak	Ya

Baris terakhir layak diperhatikan. Date roll mengubah tanggal buku dan sering memicu inisialisasi berbagai tabel kontrol. Kegagalan di tengahnya meninggalkan sistem dalam keadaan setengah maju, dan tidak ada jalan selain kembali ke titik sebelum date roll.

125.4 Mengukur seberapa jauh sebuah fase sempat berjalan

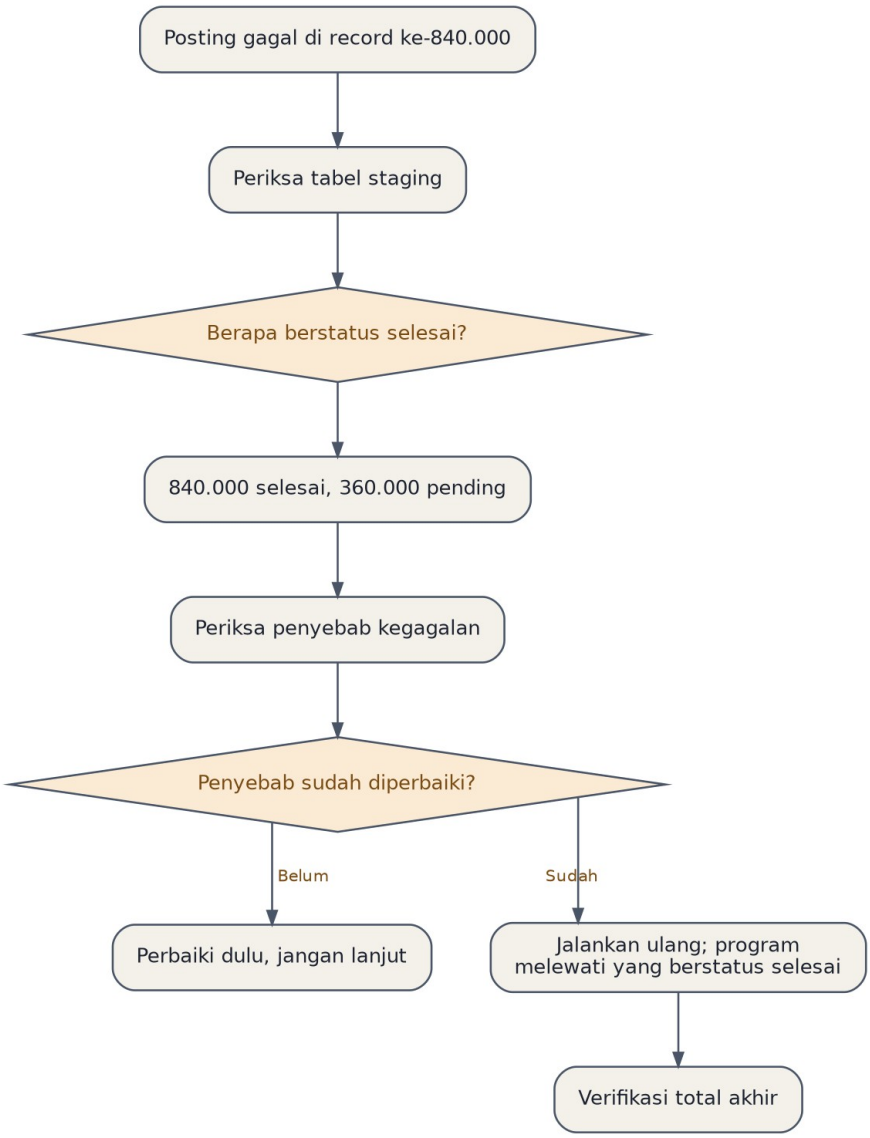
Sebelum memutuskan apa pun, Anda harus tahu persis di mana proses berhenti.

```
-- Untuk fase akrual
SELECT (SELECT COUNT(*) FROM APPWRK.AKRUAL_HITUNG
         WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'H') AS dihitung_belum_posting,
       (SELECT COUNT(*) FROM APPWRK.AKRUAL_HITUNG
         WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'P') AS sudah_posting,
       (SELECT COUNT(*) FROM DTALIB.REKENING
         WHERE STATUS = 'A') AS seharusnya,
       (SELECT SUM(BUNGA) FROM APPWRK.AKRUAL_HITUNG
         WHERE TGL_BUKU = CURRENT DATE AND STATUS = 'P') AS total_terposting
  FROM SYSIBM.SYSDUMMY1;

-- Untuk fase posting transaksi
SELECT STATUS, COUNT(*) AS jml, SUM(NOMINAL) AS total
  FROM APPWRK.STG_POSTING
 WHERE TGL_PROSES = CURRENT DATE
 GROUP BY STATUS;
```

Catat angka-angka ini sebelum melakukan tindakan apa pun. Setelah restore atau rerun, angka yang sama harus dihitung ulang dan dibandingkan. Tanpa catatan sebelum, Anda kehilangan kemampuan membuktikan bahwa pemulihan berhasil.

Bab 126. Skenario EOD Bermasalah

126.1 Skenario A — Kegagalan di fase posting, penanda ada



Gambar 102

Yang harus diperiksa sebelum melanjutkan: apakah kegagalan disebabkan data tertentu atau kondisi sistem. Bila satu record bermasalah menyebabkan program berhenti, melanjutkan tanpa menangani record itu akan berhenti di tempat yang sama.

```
-- Record yang menyebabkan kegagalan
SELECT * FROM APPWRK.STG_POSTING
 WHERE TGL_PROSES = CURRENT DATE
   AND STATUS = 'P'
 ORDER BY NO_URUT
 FETCH FIRST 5 ROWS ONLY;
```

Record pertama yang masih berstatus pending biasanya adalah yang bermasalah.

126.2 Skenario B — Kegagalan di akrual, hasil tersimpan

Urutan tindakan:

1. Hentikan seluruh job EOD
2. Catat angka kondisi: berapa dihitung, berapa terposting, total nilai
3. Eskalasi — ini keputusan dua orang
4. Periksa apakah yang terposting nilainya benar
5. Bila benar: lanjutkan dari yang belum terposting
6. Bila salah: reversal seluruhnya, perbaiki penyebab, jalankan dari awal
7. Verifikasi dengan rekonsiliasi lengkap
8. Dokumentasikan seluruh langkah dengan stempel waktu

Langkah empat sering dilewati karena tekanan waktu, dan itu kesalahan. Melanjutkan posting yang nilainya salah menggandakan masalah alih-alih menyelesaikannya.

```
-- Uji petik kebenaran nilai yang sudah terposting
SELECT a.NO_REK, a.BUNGA AS dihitung,
       k.SALDO, r.RATE,
       DECIMAL(k.SALDO * r.RATE / 365, 17, 2) AS seharusnya
  FROM APPWRK.AKRUAL_HITUNG a
  JOIN DTALIB.REKENING k ON k.NO_REK = a.NO_REK
  JOIN APPPRM.PRODUK_RATE r ON r.KD_PRODUK = k.KD_PRODUK
 WHERE a.TGL_BUKU = CURRENT DATE AND a.STATUS = 'P'
 FETCH FIRST 20 ROWS ONLY;
```

126.3 Skenario C — GL tidak seimbang

Ini bukan masalah teknis semata, dan penanganannya berbeda dari yang lain.

Jangan lakukan: date roll, membuka layanan, penyesuaian manual sebelum penyebab diketahui.

Urutan penyelidikan:

```
-- 1. Seberapa besar selisihnya
SELECT SUM(DEBET) - SUM(KREDIT) AS selisih_total
  FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE;

-- 2. Di akun mana
SELECT KD_GL, SUM(DEBET) - SUM(KREDIT) AS selisih
  FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE
 GROUP BY KD_GL HAVING SUM(DEBET) <> SUM(KREDIT);

-- 3. Transaksi mana yang tidak berpasangan
SELECT NO_REF, COUNT(*) AS jml_leg,
       SUM(DEBET) AS d, SUM(KREDIT) AS k
  FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE
 GROUP BY NO_REF HAVING SUM(DEBET) <> SUM(KREDIT)
 ORDER BY ABS(SUM(DEBET) - SUM(KREDIT)) DESC;

-- 4. Program apa yang menulisnya
SELECT DISTINCT PGM_PENULIS, COUNT(*) AS jml
  FROM DTALIB.GLTRX
 WHERE NO_REF IN (SELECT NO_REF FROM DTALIB.GLTRX
                  WHERE TGL_BUKU = CURRENT DATE
                  GROUP BY NO_REF HAVING SUM(DEBET) <> SUM(KREDIT))
 GROUP BY PGM_PENULIS;
```

Kueri keempat mensyaratkan tabel GL menyimpan nama program penulis. Bila belum ada, tambahkan — kolom sekecil itu menghemat berjam-jam penyelidikan setiap kali selisih muncul.

Pola selisih dan artinya:

Pola	Kemungkinan penyebab
Selisih pada satu nomor referensi besar	Satu transaksi gagal di tengah
Ribuan selisih kecil, campur positif negatif	Pembulatan, seperti kasus 15
Selisih persis sama dengan satu transaksi	Satu sisi tidak tertulis
Selisih hanya pada satu akun GL	Program yang menulis akun itu
Selisih muncul setelah rerun	Posting ganda

126.4 Skenario D — Batch window habis

Keputusan ini bukan milik Anda, tetapi Anda yang menyediakan datanya.

```
-- Berapa lama lagi EOD akan selesai, berdasarkan baseline
WITH baseline AS (
  SELECT URUTAN, NM_FASE,
         AVG(TIMESTAMPDIFF(4, CHAR(JAM_SELESAI - JAM_MULAI))) AS rata_menit
    FROM APPWRK.EOD_KONTROL
   WHERE STATUS = 'S' AND TGL_BUKU > CURRENT DATE - 30 DAYS
   GROUP BY URUTAN, NM_FASE
)
SELECT SUM(b.rata_menit) AS perkiraan_sisa_menit,
       CURRENT TIMESTAMP + (SUM(b.rata_menit)) MINUTES AS perkiraan_selesai
  FROM APPWRK.EOD_KONTROL e
  JOIN baseline b ON b.URUTAN = e.URUTAN
 WHERE e.TGL_BUKU = CURRENT DATE
   AND e.STATUS IN ('P','R');
```

Sajikan kepada manajemen dalam bentuk ini: fase yang tersisa, perkiraan selesai, dan pilihan yang tersedia. Bukan dalam bentuk keluhan bahwa waktu tidak cukup.

Pilihan	Konsekuensi
Lanjutkan, buka layanan terlambat	Nasabah terdampak, perlu komunikasi
Buka layanan dengan tanggal buku kemarin	Transaksi hari ini masuk besok, rekonsiliasi rumit
Lewati fase pelaporan, jalankan siang	Aman bila laporan tidak ada tenggat pagi
Hentikan, rollback, jalankan malam berikutnya	Dua hari dalam satu batch, risiko lebih besar

Bab 127. Mencegah EOD Bermasalah

127.1 Pemeriksaan pra-EOD yang lengkap

```
CREATE OR REPLACE VIEW APPWRK.V_SIAP_EOD AS
SELECT 'Backup pre-EOD selesai' AS pemeriksaan,
       CASE WHEN EXISTS (SELECT 1 FROM APPWRK.BACKUP_LOG
                          WHERE TGL = CURRENT DATE AND JENIS = 'PRE'
                            AND STATUS = 'S')
            THEN 'LULUS' ELSE 'GAGAL' END AS hasil
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'Tidak ada sesi interaktif',
       CASE WHEN (SELECT COUNT(*) FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
                     JOB_TYPE_FILTER => '*INTERACT'))) = 0
```

```
            THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'Tidak ada transaksi commit terbuka',
       CASE WHEN (SELECT COUNT(*) FROM QSYS2.DB_TRANSACTION_INFO
                  WHERE PENDING_CHANGES > 0) = 0
            THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'Seluruh produk aktif punya rate',
       CASE WHEN (SELECT COUNT(*) FROM APPPRM.PRODUK p
                  WHERE p.STATUS = 'A'
                    AND NOT EXISTS (SELECT 1 FROM APPPRM.PRODUK_RATE r
                                    WHERE r.KD_PRODUK = p.KD_PRODUK
                                      AND r.TGL_BERLAKU <= CURRENT DATE)) = 0
            THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'ASP di bawah 80 persen',
       CASE WHEN (SELECT SYSTEM_ASP_USED FROM TABLE(QSYS2.SYSTEM_STATUS()))
                 < 80 THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'Tabel kontrol EOD terinisialisasi',
       CASE WHEN (SELECT COUNT(*) FROM APPWRK.EOD_KONTROL
                  WHERE TGL_BUKU = CURRENT DATE) > 0
            THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT 'Replikasi DR tidak tertinggal',
       CASE WHEN (SELECT COUNT(*) FROM APPWRK.REPLIKASI_STATUS
                  WHERE LAG_DETIK > 300) = 0
            THEN 'LULUS' ELSE 'GAGAL' END
  FROM SYSIBM.SYSDUMMY1;
```

EOD tidak dimulai selama masih ada baris berstatus gagal. Jadikan ini gerbang teknis, bukan sekadar daftar periksa yang dibaca operator.

```
PGM
   DCL VAR(&GAGAL) TYPE(*DEC) LEN(3 0)

   RUNSQL SQL('CREATE TABLE QTEMP/CEKEOD AS ( +
      SELECT COUNT(*) AS JML FROM APPWRK.V_SIAP_EOD +
       WHERE HASIL = ''GAGAL'') WITH DATA') COMMIT(*NONE)

   /* Baca hasilnya, hentikan bila ada yang gagal */
   /* ... DCLF dan RCVF ... */

   IF COND(&GAGAL *GT 0) THEN(DO)
      SNDMSG MSG('EOD DIBATALKAN: pemeriksaan pra-EOD gagal') +
             TOUSR(EODOPS) MSGTYPE(*INQ)
      SNDPGMMSG MSG('Pemeriksaan pra-EOD gagal') MSGTYPE(*ESCAPE)
   ENDDO
ENDPGM
```

127.2 Latihan simulasi EOD gagal

Latih di UAT, dengan salinan data produksi termutakhir yang sudah di-mask.

Skenario latihan	Yang diuji
Fase posting berhenti di tengah	Kemampuan melanjutkan dari penanda
Fase akrual gagal setelah setengah terposting	Keputusan reversal versus restore
GL tidak seimbang sebesar satu transaksi	Kemampuan menelusuri penyebab
Rate produk baru belum terdaftar	Apakah validasi pra-EOD menangkapnya
Job menggantung menunggu data queue	Apakah batas waktu bekerja
ASP penuh di tengah EOD	Prosedur darurat penyimpanan
Batch window habis pada 85% progres	Proses pengambilan keputusan

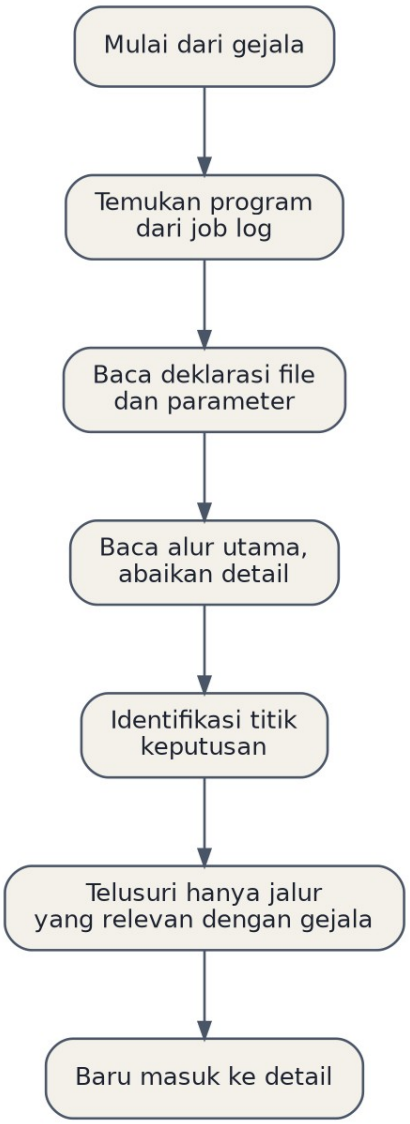
Ukuran keberhasilan bukan seberapa cepat masalah diselesaikan, melainkan apakah prosedur tertulis cukup untuk menyelesaikannya tanpa improvisasi. Setiap improvisasi adalah celah dokumentasi yang harus ditutup.

🔍 Bagian XXXVIII — Pendalaman Debug dan Analisis Program

Bab 128. Membaca Kode yang Tidak Anda Tulis

Sebagian besar waktu debug bukan dihabiskan untuk mencari kesalahan, melainkan untuk memahami apa yang seharusnya dilakukan program.

128.1 Urutan membaca



Gambar 103

Kesalahan yang paling umum adalah membaca dari baris pertama sampai terakhir. Program produksi bisa berisi tiga ribu baris, dan sembilan puluh persennya tidak relevan dengan masalah yang sedang Anda selidiki.

128.2 Pertanyaan untuk memahami program dengan cepat

Pertanyaan	Cara menjawab
File apa yang dibaca dan ditulis	Deklarasi `dcl-f`, atau `DSPPGMREF`
Parameter apa yang diterima	`dcl-pi` di prosedur utama
Program lain apa yang dipanggil	Prototipe dan `DSPPGMREF`
Di bawah commitment control?	Kata kunci `commit` pada deklarasi file
Activation group apa	`ctl-opt actgrp`
Di mana perulangan utamanya	Cari `dow`, `dou`, `for` di tingkat terluar

Pertanyaan	Cara menjawab
Apa kondisi berhentinya	Kondisi perulangan dan `leave`
Bagaimana kesalahan ditangani	`monitor`, `on-error`, indikator `(e)`

```
/* Peta ketergantungan sebelum membaca kode */
DSPPGMREF PGM(APPPGM/HITBUNGA) OUTPUT(*PRINT)
```

```
-- Modul dan service program yang terikat
SELECT * FROM TABLE(QSYS2.BOUND_MODULE_INFO('APPPGM','EODBUNGA'));
SELECT * FROM TABLE(QSYS2.BOUND_SRVPGM_INFO('APPPGM','EODBUNGA'));

-- Informasi kompilasi: kapan, dari sumber mana, dengan opsi apa
SELECT PROGRAM_NAME, PROGRAM_LIBRARY, CREATION_TIMESTAMP,
       SOURCE_FILE, SOURCE_LIBRARY, SOURCE_FILE_MEMBER,
       SOURCE_STREAM_FILE_PATH, SOURCE_TIMESTAMP,
       ACTIVATION_GROUP, USER_PROFILE, USE_ADOPTED_AUTHORITY,
       OPTIMIZATION_LEVEL, TARGET_RELEASE
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND PROGRAM_NAME = 'HITBUNGA';
```

Kolom `SOURCE_TIMESTAMP` layak diperiksa lebih dulu: bila sumber lebih baru daripada `CREATION_TIMESTAMP`, program di produksi bukan hasil kompilasi sumber yang Anda baca. Ini menjelaskan situasi membingungkan di mana kode terlihat benar tetapi perilakunya tidak.

128.3 Menemukan jalur yang relevan

Bila gejalanya adalah nilai salah pada satu field, telusuri mundur dari tempat field itu ditulis.

```
# Dari PASE, cari di seluruh sumber
cd /src
grep -rn 'BUNGA_AKRUAL' --include='*.rpgle' --include='*.sqlrpgle' .
grep -rn 'BUNGA_AKRUAL' --include='*.clle' .
```

```
-- Program yang menulis ke tabel tertentu
SELECT WHPNAM AS program, WHLNAM AS library, WHUSAG
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'KREDIT'
   AND WHUSAG IN ('2','3');
```

Bila kandidatnya banyak, persempit dengan waktu: program mana yang berjalan pada saat nilai itu berubah, menurut journal.

```
DSPJRN JRN(JRNLIB/JRNDTA) FILE((DTALIB/KREDIT)) +
       RCVRNG(*CURCHAIN) ENTTYP(*RCD) +
       FROMTIME('18/09/2026' '20:00:00') +
       OUTPUT(*OUTFILE) OUTFILE(QTEMP/JRNCARI)
```

```
SELECT DISTINCT JOPGM AS program, JOJOB, JOUSER,
       MIN(JOTSTP) AS mulai, MAX(JOTSTP) AS akhir, COUNT(*) AS jml
  FROM QTEMP.JRNCARI
 WHERE JOENTT IN ('UP','PT')
 GROUP BY JOPGM, JOJOB, JOUSER
 ORDER BY mulai;
```

Kolom `JOPGM` menunjuk program yang melakukan perubahan. Ini cara tercepat mempersempit dari dua puluh kandidat menjadi satu.

🔍 Bab 129. Teknik Debug Lanjutan

129.1 Breakpoint bersyarat yang berguna

Situasi	Bentuk breakpoint
Hanya record tertentu	`BREAK 120 WHEN NO_REK = '1234567890123456'`
Nilai tidak wajar	`BREAK 145 WHEN wBunga < 0`
Setelah sekian iterasi	`BREAK 120 WHEN wCounter > 500000`
Kombinasi	`BREAK 200 WHEN wRate = 0 AND wSaldo > 0`
Perubahan nilai	`WATCH wSaldoAkhir`

Breakpoint tanpa syarat pada program yang memproses jutaan record praktis tidak berguna. Rumuskan syaratnya dari gejala: bila keluhannya bunga nol, berhenti saat bunga nol pada saldo yang tidak nol.

129.2 Debug tanpa menghentikan program

Kadang menghentikan job produksi tidak dapat diterima. Alternatifnya adalah menambahkan jejak.

```
**free
// Jejak bersyarat, aktif hanya bila data area penanda menyala
dcl-s wJejakAktif char(1);

in *lock jejakDtaara;
wJejakAktif = jejakDtaara;
out jejakDtaara;

// Di titik yang diselidiki
if wJejakAktif = 'Y';
   exec sql
     INSERT INTO APPWRK.JEJAK_DEBUG
       (WAKTU, PROGRAM, TITIK, NO_REK, NILAI1, NILAI2, KETERANGAN)
     VALUES (CURRENT TIMESTAMP, 'HITBUNGA', 'SETELAH_GETRATE',
             :wNoRek, :wRate, :wSaldo, 'Rate diambil');
endif;
```

Penanda di data area memungkinkan jejak dinyalakan dan dimatikan tanpa mengubah program. Bila dinyalakan hanya beberapa menit, dampaknya terhadap kinerja dapat diterima.

```
/* Menyalakan */
CHGDTAARA DTAARA(APPWRK/JEJAK) VALUE('Y')
/* Mematikan setelah beberapa menit */
CHGDTAARA DTAARA(APPWRK/JEJAK) VALUE('N')
```

```
-- Analisis jejak
SELECT TITIK, COUNT(*) AS jml,
       MIN(NILAI1) AS min_rate, MAX(NILAI1) AS maks_rate,
       SUM(CASE WHEN NILAI1 = 0 THEN 1 ELSE 0 END) AS rate_nol
  FROM APPWRK.JEJAK_DEBUG
 WHERE WAKTU > CURRENT TIMESTAMP - 1 HOUR
 GROUP BY TITIK;
```

129.3 Membandingkan perilaku dua lingkungan

Situasi klasik: program berjalan benar di UAT, salah di produksi.

Yang dibandingkan	Cara
Versi objek program	`CREATION_TIMESTAMP` dan `SOURCE_TIMESTAMP`
Library list job	`DSPLIBL` di kedua sisi
Parameter aplikasi	Bandingkan tabel parameter
CCSID job dan kolom	`SYSCOLUMNS` dan atribut job
Data yang diproses	Kasus pinggiran yang hanya ada di produksi
Wewenang	`DSPOBJAUT`
Override aktif	`DSPOVR`
Service program yang terikat	`BOUND_SRVPGM_INFO`

```
-- Jalankan di kedua sistem, bandingkan hasilnya
SELECT PROGRAM_NAME, CREATION_TIMESTAMP, SOURCE_TIMESTAMP,
       ACTIVATION_GROUP, OPTIMIZATION_LEVEL,
       NUMBER_OF_MODULES, NUMBER_OF_SERVICE_PROGRAMS
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY IN ('APPPGM','UATPGM')
   AND PROGRAM_NAME = 'HITBUNGA';
```

Penyebab paling sering dari perbedaan perilaku: data. Produksi punya kasus pinggiran yang tidak ada di data uji — nilai kosong, tanggal aneh, kode yang sudah tidak dipakai, nama dengan karakter tidak lazim. Sebelum mencurigai kode, cari record produksi yang berbeda dari seluruh data uji Anda.

129.4 Menganalisis dump secara sistematis

```
WRKSPLF SELECT(EODUSER)
```

Urutan membaca dump:

Bagian	Yang dicari
Header	Message ID, program, prosedur, nomor statement
Call stack	Rantai pemanggilan dari titik masuk sampai gagal
Variabel skalar	Nilai field kunci saat gagal
Data structure	Isi struktur, sering menunjukkan data masukan
Indikator	Status indikator, untuk kode lama
File feedback	File terbuka, posisi terakhir, status I/O
Record buffer	Isi record yang sedang diproses

Record buffer adalah bagian yang paling berharga dan paling sering dilewati. Ia menunjukkan data persis yang menyebabkan kegagalan, yang kemudian bisa dicari di produksi.

```
-- Setelah tahu nomor rekening dari dump, periksa datanya
SELECT NO_REK, HEX(NO_REK) AS hex_norek,
       KD_PRODUK, HEX(KD_PRODUK) AS hex_produk,
       SALDO, STATUS, TGL_BUKA
  FROM DTALIB.REKENING
 WHERE NO_REK = '1234567890123456';
```

Tampilkan bentuk heksa untuk field yang mencurigakan. Data yang terlihat normal di layar bisa mengandung karakter tidak tercetak yang menyebabkan perbandingan gagal.

129.5 Kesalahan yang tidak menghasilkan pesan

Kelas kesalahan paling sulit: program berjalan sampai selesai, tidak ada pesan, tetapi hasilnya salah.

Gejala	Penyebab yang lazim
Sebagian record tidak diproses	Kondisi `if` yang terlalu sempit
Nilai nol yang tidak seharusnya	Data referensi tidak ditemukan
Nilai terpotong	Field hasil lebih kecil dari nilainya
Presisi hilang	Pembagian sebelum perkalian
Record terproses dua kali	Perulangan tanpa penanda
Urutan hasil berbeda	`ORDER BY` tidak deterministik
Tanggal salah	Konversi format atau tahun dua digit

Cara menemukannya: rekonsiliasi, bukan debug. Bandingkan jumlah record yang seharusnya diproses dengan yang tercatat diproses.

```
SELECT (SELECT COUNT(*) FROM DTALIB.REKENING WHERE STATUS = 'A')
         AS seharusnya,
       (SELECT COUNT(*) FROM DTALIB.BUNGALOG WHERE LOG_TGL = CURRENT DATE)
         AS terproses,
       (SELECT COUNT(*) FROM DTALIB.REKENING WHERE STATUS = 'A')
     - (SELECT COUNT(*) FROM DTALIB.BUNGALOG WHERE LOG_TGL = CURRENT DATE)
         AS selisih
  FROM SYSIBM.SYSDUMMY1;

-- Mana yang terlewat
SELECT k.NO_REK, k.KD_PRODUK, k.SALDO, k.STATUS
  FROM DTALIB.REKENING k
 WHERE k.STATUS = 'A'
   AND NOT EXISTS (SELECT 1 FROM DTALIB.BUNGALOG b
                   WHERE b.LOG_NOREK = k.NO_REK
                     AND b.LOG_TGL = CURRENT DATE);
```

Kueri kedua langsung memberi daftar record yang terlewat. Memeriksa pola di antara mereka — apakah semuanya satu kode produk, satu cabang, satu rentang tanggal — biasanya langsung menunjuk kondisi yang salah dalam kode.

129.6 Daftar periksa tinjauan kode

Untuk program yang menangani uang, tinjau terhadap daftar ini sebelum promosi.

- ☐ Setiap `CHAIN` diikuti `update`, `delete`, atau `unlock`
- ☐ Setiap pembagian diperiksa terhadap pembagi nol
- ☐ Perkalian dilakukan sebelum pembagian
- ☐ Field hasil cukup besar untuk nilai maksimum yang mungkin
- ☐ Nilai nol dari data tidak ditemukan dibedakan dari nol hasil hitung
- ☐ Setiap operasi menunggu punya batas waktu
- ☐ Commit dilakukan berkala, bukan sekali di akhir
- ☐ Program idempoten, atau ada penanda yang mencegah pengulangan
- ☐ Kesalahan ditangani dan diteruskan sebagai escape, bukan ditelan
- ☐ Jumlah record diproses dicatat untuk rekonsiliasi
- ☐ Tidak ada nilai ambang yang ditulis langsung dalam kode

- ☐ Tidak ada kredensial dalam kode sumber
- ☐ Program memeriksa lingkungan sebelum bertindak
- ☐ Dikompilasi dengan `DBGVIEW` yang memadai

Butir kelima dan kedelapan adalah dua yang paling sering menjadi akar insiden mahal. Bila Anda hanya punya waktu memeriksa dua hal, periksa keduanya.

📕 Bagian XXXIX — Katalog Runbook Lanjutan

Lima belas runbook tambahan melengkapi Bagian IX. Disusun dengan struktur yang sama agar bisa dijalankan tanpa membaca bab lain.

📕 Bab 130. Runbook Basis Data

RB-016: Tabel produksi tidak dijurnal

Pemicu. Pemeriksaan mingguan menemukan tabel di library produksi tanpa journaling, atau `SQLCODE -7008` muncul saat program berjalan di bawah commitment control.

Dampak. Tidak ada titik pulih untuk tabel itu. Replikasi ke DR tidak menyertakannya.

Diagnosis

```
SELECT o.OBJNAME, o.OBJSIZE / 1048576 AS mb, o.OBJCREATED
  FROM TABLE(QSYS2.OBJECT_STATISTICS('DTALIB','*FILE')) o
 WHERE o.OBJATTRIBUTE IN ('PF','PF38')
   AND NOT EXISTS (SELECT 1 FROM QSYS2.JOURNALED_OBJECTS j
                   WHERE j.JOURNALED_OBJECT = o.OBJNAME
                     AND j.JOURNALED_OBJECT_LIBRARY = 'DTALIB');
```

Perbaikan

```
/* Aktifkan journaling; objek boleh sedang dipakai */
STRJRNPF FILE(DTALIB/NAMATABEL) +
         JRN(JRNLIB/JRNDTA) +
         IMAGES(*BOTH) OMTJRNE(*OPNCLO)

/* Verifikasi */
DSPFD FILE(DTALIB/NAMATABEL) TYPE(*JRN)
```

PERINGATAN. Journaling dimulai dari sekarang. Data yang ada sebelumnya tidak punya entri journal, sehingga pemulihan ke titik sebelum ini tetap mustahil. Lakukan save penuh segera setelah mengaktifkan.

Pencegahan. Jadwalkan kueri diagnosis di atas mingguan. Tambahkan langkah aktivasi journaling ke daftar periksa setiap kali tabel baru dibuat di produksi.

RB-017: Constraint dalam status check pending

Pemicu. Pemeriksaan pasca-restore, atau kegagalan validasi data yang seharusnya tertangkap constraint.

Diagnosis

```
SELECT CONSTRAINT_SCHEMA, CONSTRAINT_NAME, TABLE_NAME,
       CONSTRAINT_TYPE, ENABLED, CHECK_PENDING
  FROM QSYS2.SYSCST
 WHERE CHECK_PENDING = 'YES';
```

Perbaikan

```
/* Periksa data yang melanggar sebelum mengaktifkan */
```

```
-- Untuk kunci asing, cari baris yatim
SELECT t.NO_REK, COUNT(*) AS jml
  FROM DTALIB.TRXHIST t
 WHERE NOT EXISTS (SELECT 1 FROM DTALIB.REKENING r
                    WHERE r.NO_REK = t.NO_REK)
 GROUP BY t.NO_REK;
```

```
/* Setelah data bersih, aktifkan kembali */
CHGPFCST FILE(DTALIB/TRXHIST) CST(FK_TRX_REKENING) STATE(*ENABLED)
```

Eskalasi. Bila ada baris yang melanggar, keputusan apa yang dilakukan terhadapnya adalah keputusan bisnis, bukan teknis.

RB-018: Index rusak atau jalur akses tidak valid

Pemicu. `CPF4328` atau pesan yang menyebut jalur akses tidak valid, biasanya setelah penghentian tidak normal.

Diagnosis

```
DSPFD FILE(DTALIB/*ALL) TYPE(*ACCPTH) OUTPUT(*PRINT)
EDTRBDAP
```

`EDTRBDAP` menampilkan jalur akses yang sedang dibangun ulang beserta perkiraan waktunya.

Perbaikan. Sistem membangun ulang otomatis. Yang bisa Anda lakukan adalah mengatur urutannya.

```
/* Naikkan prioritas pembangunan ulang untuk tabel kritikal */
/* Lewat layar EDTRBDAP, ubah sequence menjadi 1-99 */
```

PERINGATAN. Jangan memulai EOD selama jalur akses tabel yang terlibat masih dibangun ulang. Program akan berjalan tetapi jauh lebih lambat, dan bisa melewati batch window.

RB-019: Kueri ad hoc melumpuhkan sistem

Pemicu. Satu job menyerap CPU atau menghasilkan penyimpanan sementara sangat besar.

Diagnosis

```
SELECT JOB_NAME, AUTHORIZATION_NAME, TEMPORARY_STORAGE,
       ELAPSED_CPU_PERCENTAGE, ELAPSED_TOTAL_DISK_IO_COUNT
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY TEMPORARY_STORAGE DESC
 FETCH FIRST 10 ROWS ONLY;

-- Statement yang sedang dijalankannya
SELECT JOB_NAME, SQL_STATEMENT_TEXT, STATEMENT_START_TIMESTAMP
  FROM QSYS2.ACTIVE_QUERY_INFO
 ORDER BY STATEMENT_START_TIMESTAMP;
```

Perbaikan bertingkat

1. Turunkan prioritas: `CHGJOB JOB(...) RUNPTY(60)`
2. Bila masih mengganggu, hubungi pemiliknya
3. Bila tidak ada respons dalam sepuluh menit atau dampaknya kritis, akhiri job

```
ENDJOB JOB(nomor/user/nama) OPTION(*CNTRLD) DELAY(30)
```

Pencegahan. `QUERY_TIME_LIMIT` di QAQQINI, exit program ODBC yang membatasi jam akses, dan pemindahan beban pelaporan ke partisi MIS.

📕 Bab 131. Runbook Aplikasi

RB-020: Program produksi berbeda dari sumbernya

Pemicu. Perilaku program tidak sesuai kode yang dibaca, atau pemeriksaan rutin menemukan ketidaksesuaian.

Diagnosis

```
SELECT PROGRAM_NAME, CREATION_TIMESTAMP,
       SOURCE_FILE, SOURCE_LIBRARY, SOURCE_FILE_MEMBER,
       SOURCE_STREAM_FILE_PATH, SOURCE_TIMESTAMP
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND SOURCE_TIMESTAMP > CREATION_TIMESTAMP;
```

Baris yang muncul berarti sumber lebih baru daripada objek — objek produksi bukan hasil kompilasi sumber terkini.

Perbaikan. Jangan langsung mengompilasi ulang di produksi. Telusuri dulu: apakah perubahan sumber memang belum dipromosikan, atau justru objek produksi yang dikompilasi dari sumber lain.

Eskalasi. Bila objek produksi tidak bisa dipetakan ke sumber mana pun, ini masalah tata kelola perubahan, bukan sekadar masalah teknis. Laporkan.

RB-021: Program hilang dari produksi

Pemicu. `CPF9801` pada program yang seharusnya ada.

Diagnosis

```
-- Siapa menghapusnya dan kapan
SELECT ENTRY_TIMESTAMP, USER_NAME, JOB_NAME,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'DO',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS))
 WHERE OBJECT_NAME = 'NAMAPROGRAM';
```

Perbaikan

```
/* Pulihkan dari salinan rollback bila ada */
RSTOBJ OBJ(NAMAPROGRAM) SAVLIB(APPPGM) DEV(*SAVF) +
       SAVF(ROLLBACK/BTIKET123) RSTLIB(APPPGM)

/* Atau dari backup */
RSTOBJ OBJ(NAMAPROGRAM) SAVLIB(APPPGM) DEV(TAPMLB01) +
       OBJTYPE(*PGM) RSTLIB(APPPGM)
```

Verifikasi. Bandingkan `CREATION_TIMESTAMP` objek yang dipulihkan dengan yang tercatat sebelum penghapusan. Pastikan versi yang dipulihkan adalah versi yang benar, bukan versi lama.

RB-022: Laporan regulator tidak terbentuk

Pemicu. Berkas laporan tidak ada di lokasi yang diharapkan pada pagi hari.

Dampak. Tenggat regulator, tidak bisa ditawar.

Diagnosis

```
-- Apakah fase pelaporan berjalan
SELECT NM_FASE, STATUS, JAM_MULAI, JAM_SELESAI, PESAN_ERROR
  FROM APPWRK.EOD_KONTROL
 WHERE TGL_BUKU = CURRENT DATE - 1 DAY
   AND KD_FASE LIKE 'LAPOR%';

-- Apakah berkas terbentuk tetapi di tempat lain
SELECT PATH_NAME, DATA_SIZE, CREATE_TIMESTAMP
  FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
         START_PATH_NAME => '/laporan',
         SUBTREE_DIRECTORIES => 'YES'))
 WHERE CREATE_TIMESTAMP > CURRENT DATE - 1 DAY;
```

Perbaikan. Laporan umumnya bisa dibentuk ulang dari data yang sudah ada, asalkan tanggal buku belum bergeser lebih jauh.

```
CALL PGM(APPPGM/BENTUKLAPOR) PARM('20260917' 'LAPKREDIT')
```

Verifikasi wajib sebelum kirim. Jalankan view validasi laporan. Jangan mengirim berkas yang dibentuk ulang tanpa rekonsiliasi ulang terhadap GL.

Eskalasi. Bila tenggat kurang dari dua jam, informasikan tim kepatuhan sekarang, bukan setelah gagal.

RB-023: Transaksi ganda terdeteksi

Pemicu. Nasabah melaporkan didebet dua kali, atau rekonsiliasi menemukan pasangan transaksi identik.

Diagnosis

```
-- Transaksi identik dalam jendela waktu pendek
SELECT NO_REK, NOMINAL, KD_TRX,
       COUNT(*) AS jml,
       MIN(WAKTU) AS pertama, MAX(WAKTU) AS terakhir,
       LISTAGG(NO_REF, ', ') AS referensi
  FROM DTALIB.TRXHIST
 WHERE TGL_TRX = CURRENT DATE
 GROUP BY NO_REK, NOMINAL, KD_TRX
HAVING COUNT(*) > 1
   AND TIMESTAMPDIFF(2, CHAR(MAX(WAKTU) - MIN(WAKTU))) < 60
 ORDER BY jml DESC;
```

Kemungkinan penyebab

Penyebab	Cara membedakan
Retry klien tanpa idempotency	Referensi berbeda, waktu berdekatan
Rerun fase posting	Banyak sekali, waktunya serentak
Nasabah memang bertransaksi dua kali	Waktu berjauhan, pola wajar
Timeout lalu kirim ulang	Satu berstatus gagal di sisi kanal

Perbaikan. Koreksi dilakukan dengan transaksi penyeimbang yang tercatat, bukan dengan menghapus atau mengubah record asli. Libatkan operasi dan akuntansi.

Pencegahan. Idempotency key pada seluruh jalur masuk transaksi. Lihat Bab 71.4.

📕 Bab 132. Runbook Infrastruktur

RB-024: Sertifikat kedaluwarsa

Pemicu. Layanan TLS berhenti bekerja, atau pemantauan melaporkan sertifikat mendekati kedaluwarsa.

Diagnosis

```
SELECT CERTIFICATE_LABEL, SUBJECT_COMMON_NAME,
       VALIDITY_START, VALIDITY_END,
       DAYS(VALIDITY_END) - DAYS(CURRENT DATE) AS sisa_hari
  FROM QSYS2.CERTIFICATE_INFO(
         CERTIFICATE_STORE_PASSWORD => '*NOPWD')
 ORDER BY VALIDITY_END;
```

Perbaikan. Pembaruan sertifikat dilakukan lewat Digital Certificate Manager. Setelah diperbarui, seluruh server yang memakainya harus dimulai ulang.

```
ENDTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
STRTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
```

PERINGATAN. Jangan menghapus sertifikat lama sebelum yang baru terverifikasi bekerja. Simpan keduanya sampai seluruh klien berhasil terhubung.

Pencegahan. Kueri di atas dijadwalkan mingguan dengan peringatan pada 120, 60, dan 30 hari.

RB-025: Waktu sistem tidak sinkron

Pemicu. Selisih waktu antara IBM i dan sistem lain, atau stempel waktu transaksi tidak konsisten.

Dampak. Rekonsiliasi lintas sistem gagal, jejak audit sulit dipadankan, sertifikat bisa ditolak.

Diagnosis

```
DSPSYSVAL SYSVAL(QTIME)
DSPSYSVAL SYSVAL(QUTCOFFSET)
DSPSYSVAL SYSVAL(QTIMZON)
```

```
SELECT CURRENT TIMESTAMP AS waktu_sistem,
       CURRENT TIMEZONE AS zona
  FROM SYSIBM.SYSDUMMY1;
```

Perbaikan. Aktifkan sinkronisasi waktu jaringan, jangan mengubah manual.

```
CHGNTPA RMTSYS('10.20.10.1') POLLITV(15) AUTOSTART(*YES)
STRTCPSVR SERVER(*NTP)
```

PERINGATAN. Mengubah waktu sistem mundur di sistem produksi berbahaya. Stempel waktu journal dan transaksi menjadi tidak berurutan, dan sebagian proses bisa berperilaku tak terduga. Bila koreksi besar dibutuhkan, lakukan dalam jendela pemeliharaan dengan sistem berhenti.

RB-026: Perangkat tape tidak terbaca

Pemicu. `CPF4257` atau kegagalan backup dengan pesan perangkat tidak siap.

Diagnosis

```
WRKCFGSTS CFGTYPE(*DEV) CFGD(TAPMLB01)
WRKMLBSTS MLB(TAPMLB01)
DSPLOG PERIOD((*AVAIL *CURRENT)) MSGID(CPF6700 CPF4200)
```

Perbaikan bertahap

```
/* 1. Matikan lalu nyalakan perangkat */
VRYCFG CFGOBJ(TAPMLB01) CFGTYPE(*DEV) STATUS(*OFF)
VRYCFG CFGOBJ(TAPMLB01) CFGTYPE(*DEV) STATUS(*ON)

/* 2. Bila library tape, periksa dan setel ulang */
CHKTAP DEV(TAPMLB01)
```

3. Periksa fisik: kabel, lampu status, apakah ada media tersangkut
4. Bersihkan drive bila indikator pembersihan menyala
5. Buka kasus ke vendor perangkat keras

Sementara perbaikan berlangsung. Alihkan backup ke save file di disk agar tidak ada malam tanpa titik pulih. Save file dipindahkan ke media setelah perangkat pulih.

```
CRTSAVF FILE(BACKUP/DARURAT01)
SAVLIB LIB(DTALIB) DEV(*SAVF) SAVF(BACKUP/DARURAT01) +
       SAVACT(*SYNCLIB)
```

RB-027: Partisi kehabisan memori

Pemicu. Faulting sangat tinggi di seluruh pool, sistem melambat menyeluruh.

Diagnosis

```
SELECT POOL_NAME, CURRENT_SIZE / 1024 AS mb,
       MAXIMUM_ACTIVE_THREADS,
       DATABASE_FAULTS, NON_DATABASE_FAULTS,
       ACTIVE_TO_INELIGIBLE
  FROM TABLE(QSYS2.MEMORY_POOL(RESET_STATISTICS => 'YES'));
```

Tunggu dua menit, jalankan lagi dengan `RESET_STATISTICS => 'NO'` untuk melihat laju sesungguhnya.

Perbaikan bertingkat

1. Kurangi beban serentak: turunkan `MAXACT` pada job queue batch
2. Pindahkan memori antar pool bila satu pool berlebih
3. Pindahkan memori dari partisi lain lewat HMC bila tersedia
4. Akhiri job yang memakan penyimpanan sementara besar

```
CHGSHRPOOL POOL(*SHRPOOL2) SIZE(*SAME) ACTLVL(8)
CHGJOBQE SBSD(QGPL/EODSBS) JOBQ(QGPL/EODJOBQ) MAXACT(2)
```

PERINGATAN. Menambah `ACTLVL` saat faulting tinggi memperburuk keadaan. Bila faulting tinggi, yang kurang adalah memori, bukan activity level.

RB-028: Jalur multipath ke storage hilang satu sisi

Pemicu. Pemeriksaan rutin VIOS, atau peringatan dari storage.

Dampak. Tidak terlihat sekarang, tetapi redundansi hilang. Kegagalan jalur kedua akan menghentikan sistem.

Diagnosis

```
# Dari VIOS
lspath
lsmap -all
errlog | head -40
```

Perbaikan. Identifikasi apakah masalahnya di adapter, kabel, switch fabric, atau storage. Libatkan tim storage dan vendor.

PERINGATAN. Selama hanya satu jalur aktif, hindari pekerjaan pemeliharaan apa pun yang menyentuh jalur itu. Sistem sedang berjalan tanpa jaring pengaman.

Eskalasi. Ini insiden tingkat tinggi meskipun tidak ada dampak yang terlihat. Perlakukan sesuai.

🔒 Bab 133. Runbook Keamanan

RB-029: Profil dengan wewenang berlebih ditemukan

Pemicu. Audit rutin, atau temuan auditor.

Diagnosis

```
SELECT AUTHORIZATION_NAME, STATUS, USER_CLASS_NAME,
       SPECIAL_AUTHORITIES, GROUP_PROFILE_NAME,
       PREVIOUS_SIGNON, TEXT_DESCRIPTION
  FROM QSYS2.USER_INFO
 WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%'
    OR SPECIAL_AUTHORITIES LIKE '%*SECADM%'
 ORDER BY AUTHORIZATION_NAME;

-- Apa yang sebenarnya dilakukan profil ini 90 hari terakhir
SELECT ENTRY_TYPE, COUNT(*) AS jml
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 90 DAYS))
 WHERE USER_NAME = 'PROFIL_DIPERIKSA'
 GROUP BY ENTRY_TYPE
 ORDER BY jml DESC;
```

Perbaikan. Jangan mencabut wewenang sekaligus. Urutannya:

1. Pahami apa yang dikerjakan profil itu dari data audit
2. Tentukan wewenang minimum yang benar-benar dibutuhkan
3. Siapkan wewenang pengganti, lewat authorization list atau adopsi program
4. Uji di lingkungan non-produksi
5. Terapkan dengan jendela dan rencana pengembalian
6. Pantau kegagalan wewenang selama dua minggu

```
-- Pantau kegagalan setelah pencabutan
SELECT ENTRY_TIMESTAMP, USER_NAME, OBJECT_LIBRARY,
       OBJECT_NAME, VIOLATION_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'AF',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 24 HOURS))
 WHERE USER_NAME = 'PROFIL_DIPERIKSA';
```

PERINGATAN. Mencabut `*ALLOBJ` dari profil yang dipakai proses batch bisa menghentikan EOD malam itu juga. Lakukan di awal minggu, bukan Jumat sore.

RB-030: Audit journal berhenti mencatat

Pemicu. Tidak ada entri baru di audit journal, atau `QAUDCTL` ditemukan tidak aktif.

Dampak. Kehilangan jejak audit. Ini kondisi serius yang punya kewajiban pelaporan tersendiri di banyak institusi.

Diagnosis

```
SELECT SYSTEM_VALUE_NAME, CURRENT_CHARACTER_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME IN ('QAUDCTL','QAUDLVL','QAUDLVL2','QAUDENDACN');

-- Entri terakhir yang tercatat
SELECT MAX(ENTRY_TIMESTAMP) AS entri_terakhir
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 7 DAYS));

-- Siapa mengubah QAUDCTL
SELECT ENTRY_TIMESTAMP, USER_NAME, SYSTEM_VALUE_NAME,
       OLD_VALUE, NEW_VALUE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'SV',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS))
 WHERE SYSTEM_VALUE_NAME LIKE 'QAUD%';
```

Kemungkinan penyebab

Penyebab	Penanganan
ASP journal penuh	Bersihkan, aktifkan kembali
Receiver tidak bisa dibuat	Periksa wewenang dan ruang
`QAUDCTL` diubah menjadi `*NONE`	Selidiki siapa dan kenapa
Journal rusak	Buat journal baru, laporkan

Perbaikan

```
/* Aktifkan kembali */
CHGSYSVAL SYSVAL(QAUDCTL) VALUE('*AUDLVL *OBJAUD')

/* Verifikasi receiver terpasang */
WRKJRNA JRN(QSYS/QAUDJRN)

/* Bila perlu receiver baru */
CHGJRN JRN(QSYS/QAUDJRN) JRNRCV(*GEN)
```

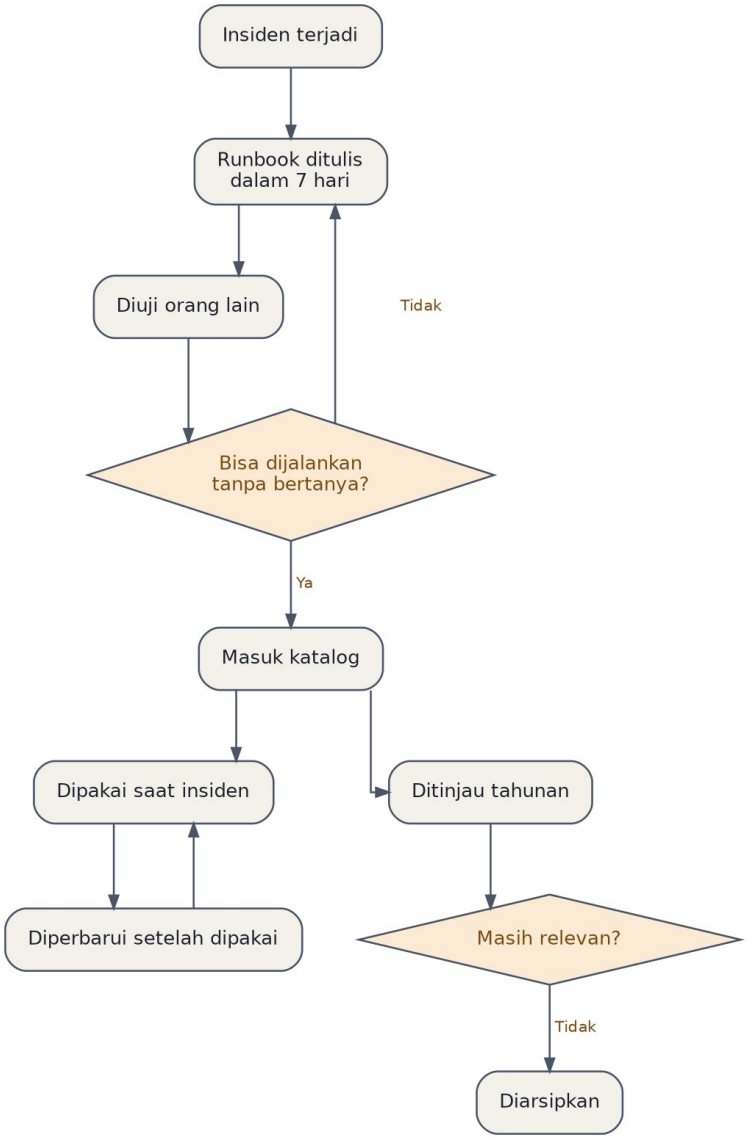
Eskalasi. Segera ke Security Officer dan manajer TI, tanpa menunggu. Periode tanpa jejak audit harus didokumentasikan: kapan mulai, kapan berakhir, dan apa yang terjadi selama itu sejauh bisa direkonstruksi dari sumber lain.

Pencegahan. Setel `QAUDENDACN` agar sistem berhenti bila audit tidak bisa dicatat, bukan melanjutkan tanpa audit. Tambahkan pemeriksaan keaktifan audit ke pemantauan harian.

```
-- Pemeriksaan harian: apakah audit masih mencatat
SELECT CASE WHEN MAX(ENTRY_TIMESTAMP) > CURRENT TIMESTAMP - 2 HOURS
            THEN 'NORMAL' ELSE 'AUDIT BERHENTI' END AS status
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 24 HOURS));
```

📕 Bab 134. Memelihara Katalog Runbook

134.1 Siklus hidup sebuah runbook



Gambar 104

134.2 Tanda runbook perlu diperbarui

Tanda	Artinya
Orang yang menjalankannya bertanya	Ada langkah yang tidak jelas
Perintah gagal karena nama objek berubah	Belum disesuaikan dengan perubahan
Langkah dilewati karena tidak relevan lagi	Sistem sudah berubah
Diperlukan improvisasi	Ada kasus yang belum tercakup
Tidak pernah dipakai dua tahun	Masalahnya mungkin sudah tercegah

Baris terakhir bukan alasan menghapus. Runbook untuk kegagalan katastrofik memang jarang dipakai, dan justru itu yang harus paling terpelihara.

134.3 Indeks runbook

Kode	Judul	Bab
RB-001	Disk ASP melewati ambang kritis	Bagian IX
RB-002	Job MSGW di tengah EOD	Bagian IX
RB-003	Record lock menahan job EOD	Bagian IX
RB-004	EOD gagal di fase akrual	Bagian IX
RB-005	GL tidak seimbang	Bagian IX
RB-006	Replikasi DR tertinggal	Bagian IX
RB-007	Kinerja menurun mendadak	Bagian IX
RB-008	Backup semalam gagal	Bagian IX
RB-009	Level check massal	Bagian IX
RB-010	Profil terkunci massal	Bagian IX
RB-011	IPL tidak selesai	Bagian IX
RB-012	Job tabel penuh	Bagian IX
RB-013	Akses ODBC mencurigakan	Bagian IX
RB-014	Tabel riwayat tumbuh tak terkendali	Bagian IX
RB-015	Kehilangan partisi produksi	Bagian IX
RB-016	Tabel produksi tidak dijurnal	Bab 130
RB-017	Constraint check pending	Bab 130
RB-018	Index rusak	Bab 130
RB-019	Kueri ad hoc melumpuhkan sistem	Bab 130
RB-020	Program berbeda dari sumbernya	Bab 131
RB-021	Program hilang dari produksi	Bab 131
RB-022	Laporan regulator tidak terbentuk	Bab 131
RB-023	Transaksi ganda	Bab 131
RB-024	Sertifikat kedaluwarsa	Bab 132
RB-025	Waktu sistem tidak sinkron	Bab 132
RB-026	Perangkat tape tidak terbaca	Bab 132
RB-027	Partisi kehabisan memori	Bab 132
RB-028	Multipath hilang satu sisi	Bab 132
RB-029	Profil wewenang berlebih	Bab 133
RB-030	Audit journal berhenti mencatat	Bab 133

Salin indeks ini ke satu halaman, cetak, dan tempel di ruang operasi. Pada pukul tiga pagi, mengetahui bahwa runbook-nya ada dan di mana mencarinya sudah setengah penyelesaian.

💾 Bagian XL — Pendalaman Prosedur Pemulihan Bencana

Bagian V memberi kerangka HA/DR. Bagian ini adalah prosedurnya, ditulis untuk dijalankan.

Bab 135. Prosedur Switchover Terencana

135.1 Persiapan, tujuh hari sebelum

- ☐ Jendela disepakati dan diumumkan ke seluruh pemangku kepentingan
- ☐ Pembekuan perubahan diberlakukan
- ☐ Seluruh mitra interface diberi tahu, termasuk jendela dan alamat yang akan dipakai
- ☐ Daftar periksa verifikasi disiapkan dan disetujui
- ☐ Peran ditetapkan dengan nama, termasuk cadangan
- ☐ Prosedur dibaca bersama oleh seluruh tim yang terlibat
- ☐ Angka kontrol terakhir dicatat sebagai pembanding

```
-- Angka kontrol yang dicatat sebelum switchover
CREATE TABLE APPWRK.KONTROL_DR (
    WAKTU       TIMESTAMP   NOT NULL,
    TAHAP       VARCHAR(20) NOT NULL,
    NAMA_UKURAN VARCHAR(40) NOT NULL,
    NILAI       DECIMAL(21,2),
    JUMLAH      BIGINT
);

INSERT INTO APPWRK.KONTROL_DR
SELECT CURRENT TIMESTAMP, 'SEBELUM', 'REKENING_AKTIF',
       SUM(SALDO), COUNT(*)
  FROM DTALIB.REKENING WHERE STATUS = 'A'
UNION ALL
SELECT CURRENT TIMESTAMP, 'SEBELUM', 'GL_HARI_INI',
       SUM(DEBET), COUNT(*)
  FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE
UNION ALL
SELECT CURRENT TIMESTAMP, 'SEBELUM', 'KREDIT_AKTIF',
       SUM(BAKI_DEBET), COUNT(*)
  FROM DTALIB.KREDIT WHERE STATUS = 'A'
UNION ALL
SELECT CURRENT TIMESTAMP, 'SEBELUM', 'TRX_HARI_INI',
       SUM(NOMINAL), COUNT(*)
  FROM DTALIB.TRXHIST WHERE TGL_TRX = CURRENT DATE;
```

135.2 Hari pelaksanaan, tahap demi tahap

Tahap 1 — Penutupan kanal

Langkah	Perintah atau tindakan	Verifikasi
Umumkan ke pengguna	`SNDBRKMSG` ke semua workstation	Pesan diterima
Tutup kanal digital	Koordinasi dengan tim kanal	Trafik masuk nol
Tutup cabang	Koordinasi dengan operasi	Tidak ada sesi aktif
Tunggu transaksi berjalan selesai	Pantau job aktif	Tidak ada transaksi pending

```
SNDBRKMSG MSG('Sistem akan dialihkan ke DR dalam 15 menit. +
               Selesaikan transaksi Anda.') TOMSGQ(*ALLWS)
```

```
-- Verifikasi tidak ada transaksi menggantung
SELECT COUNT(*) AS trx_pending
  FROM DTALIB.PESAN_KORELASI WHERE STATUS = 'K';

SELECT COUNT(*) AS commit_terbuka
  FROM QSYS2.DB_TRANSACTION_INFO WHERE PENDING_CHANGES > 0;

SELECT COUNT(*) AS sesi_interaktif
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_TYPE_FILTER => '*INTERACT'));
```

Ketiganya harus nol sebelum melanjutkan. Bila tidak nol, cari tahu penyebabnya — jangan memaksa.

Tahap 2 — Verifikasi replikasi

```
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA) DETAIL(*FULL)
WRKCLU
```

```
SELECT CLUSTER_RESOURCE_GROUP, CRG_STATUS,
       PRIMARY_NODE, RECOVERY_DOMAIN
  FROM QSYS2.CLUSTER_RESOURCE_GROUP_INFO;
```

Gerbang keras: lag replikasi harus nol. Bukan mendekati nol, bukan beberapa detik. Melanjutkan dengan lag berarti kehilangan transaksi sebanyak lag tersebut, dan di perbankan itu berarti transaksi nasabah hilang tanpa jejak.

Bila lag tidak nol setelah lima belas menit menunggu, batalkan switchover dan selidiki. Membatalkan switchover terencana adalah hasil yang dapat diterima; kehilangan data tidak.

Tahap 3 — Penghentian aplikasi

```
ENDSBS SBS(EODSBS) OPTION(*CNTRLD) DELAY(300)
ENDSBS SBS(QINTER) OPTION(*CNTRLD) DELAY(180)
ENDTCPSVR SERVER(*HTTP)
ENDTCPSVR SERVER(*DATABASE)
```

```
-- Verifikasi subsystem aplikasi benar-benar berhenti
SELECT SUBSYSTEM_DESCRIPTION, STATUS, CURRENT_ACTIVE_JOBS
  FROM QSYS2.SUBSYSTEM_INFO
 WHERE STATUS = 'ACTIVE';
```

Tahap 4 — Perpindahan peran

```
/* Catat waktu persis sebelum eksekusi */
CHGCRGPRI CLUSTER(CLSTR1) CRG(CRGDTA)

/* Pantau progres */
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA)
```

Perpindahan bisa memakan beberapa menit. Jangan menjalankan perintah lain selama berlangsung.

Tahap 5 — Aktivasi di node baru

```
/* Verifikasi iASP tersedia */
WRKCFGSTS CFGTYPE(*DEV) CFGD(IASP01)
VRYCFG CFGOBJ(IASP01) CFGTYPE(*DEV) STATUS(*ON)

/* Mulai subsystem aplikasi */
STRSBS SBSD(QGPL/EODSBS)
STRSBS SBSD(QSYS/QINTER)
STRTCPSVR SERVER(*DATABASE)
STRTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
```

Tahap 6 — Verifikasi teknis

```
-- Angka kontrol setelah switchover
INSERT INTO APPWRK.KONTROL_DR
SELECT CURRENT TIMESTAMP, 'SESUDAH', 'REKENING_AKTIF',
       SUM(SALDO), COUNT(*)
  FROM DTALIB.REKENING WHERE STATUS = 'A'
UNION ALL
SELECT CURRENT TIMESTAMP, 'SESUDAH', 'GL_HARI_INI',
       SUM(DEBET), COUNT(*)
  FROM DTALIB.GLTRX WHERE TGL_BUKU = CURRENT DATE
UNION ALL
SELECT CURRENT TIMESTAMP, 'SESUDAH', 'KREDIT_AKTIF',
       SUM(BAKI_DEBET), COUNT(*)
  FROM DTALIB.KREDIT WHERE STATUS = 'A'
UNION ALL
SELECT CURRENT TIMESTAMP, 'SESUDAH', 'TRX_HARI_INI',
       SUM(NOMINAL), COUNT(*)
  FROM DTALIB.TRXHIST WHERE TGL_TRX = CURRENT DATE;

-- Perbandingan; selisih harus nol
SELECT s.NAMA_UKURAN,
       s.NILAI AS nilai_sebelum, t.NILAI AS nilai_sesudah,
       s.NILAI - t.NILAI AS selisih_nilai,
       s.JUMLAH AS jml_sebelum, t.JUMLAH AS jml_sesudah,
       s.JUMLAH - t.JUMLAH AS selisih_jumlah
  FROM APPWRK.KONTROL_DR s
  JOIN APPWRK.KONTROL_DR t
    ON t.NAMA_UKURAN = s.NAMA_UKURAN AND t.TAHAP = 'SESUDAH'
 WHERE s.TAHAP = 'SEBELUM';
```

Daftar periksa verifikasi teknis:

- ☐ Seluruh angka kontrol cocok, selisih nol
- ☐ Seluruh subsystem yang seharusnya aktif sudah aktif
- ☐ Seluruh interface TCP/IP naik dengan alamat yang benar
- ☐ Journaling aktif untuk seluruh tabel produksi
- ☐ Job scheduler entry ada dan tidak ada yang terlewat
- ☐ Tidak ada pesan gagal di QSYSOPR
- ☐ Server database menerima koneksi

Tahap 7 — Verifikasi kanal

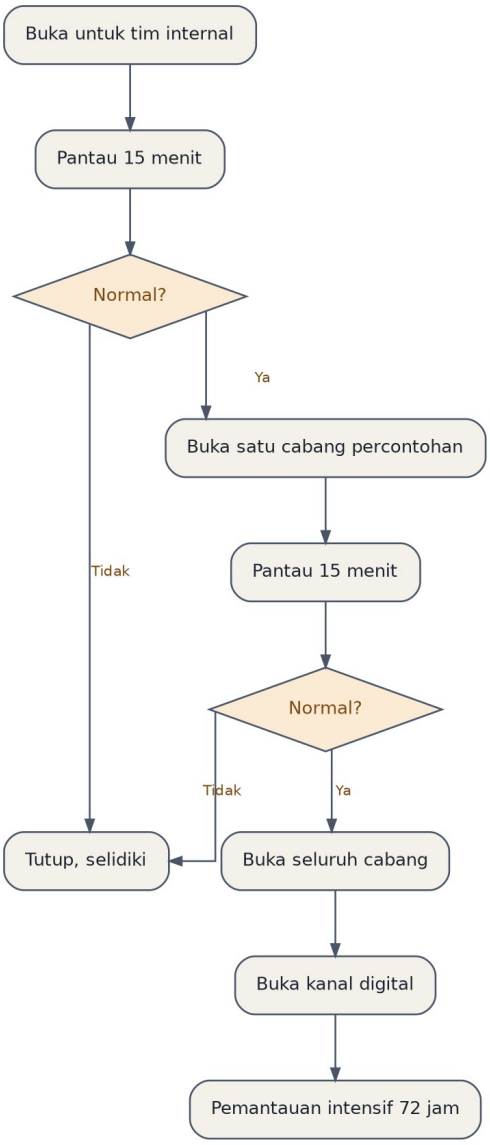
Setiap kanal diuji dengan satu transaksi nyata, bukan sekadar uji koneksi.

Kanal	Uji	Penanggung jawab
Teller cabang	Satu setoran dan satu penarikan	Operasi cabang
ATM	Satu tarik tunai dan satu cek saldo	Tim kanal

Kanal	Uji	Penanggung jawab
Mobile banking	Satu transfer antar rekening	Tim digital
Internet banking	Satu pembayaran tagihan	Tim digital
Sistem pembayaran	Satu transfer antar bank	Tim pembayaran
Kartu	Satu otorisasi	Tim kartu

Gerbang keras: seluruh kanal harus lulus sebelum layanan dibuka umum. Kasus 19 di Bab 91 terjadi karena tahap ini dilewati.

Tahap 8 — Pembukaan bertahap



Gambar 105

Pembukaan bertahap membatasi jumlah nasabah terdampak bila ada yang terlewat dalam verifikasi.

135.3 Kriteria pembatalan

Tetapkan sebelum hari H, dan patuhi meskipun sudah larut.

Kondisi	Tindakan
Lag replikasi tidak nol setelah 15 menit	Batalkan
Angka kontrol tidak cocok	Batalkan, kembali ke primer
Satu kanal gagal setelah 30 menit upaya	Batalkan atau tunda pembukaan
Melewati batas waktu yang disepakati	Batalkan, jadwalkan ulang
Personel kunci tidak tersedia	Batalkan sebelum mulai

Pembatalan bukan kegagalan. Switchover yang dibatalkan dengan tertib jauh lebih baik daripada switchover yang dipaksakan lalu harus dipulihkan pada pukul lima pagi.

Bab 136. Prosedur Failover Tak Terencana

136.1 Perbedaan mendasar dari switchover

Aspek	Switchover	Failover
Sistem primer	Sehat	Hilang atau rusak
Lag replikasi	Bisa dinolkan	Apa adanya
Kehilangan data	Nol	Sebesar lag
Waktu persiapan	Berhari-hari	Tidak ada
Keputusan	Terencana	Di bawah tekanan
Pengembalian	Mudah	Rumit

136.2 Urutan tindakan

Menit 0 sampai 5 — Konfirmasi

1. Konfirmasi primer benar-benar tidak dapat dipulihkan cepat
2. Periksa apakah ini kegagalan sistem atau kegagalan jaringan menuju sistem
3. Catat waktu dan gejala

Membedakan keduanya sangat penting. Bila yang putus adalah jaringan, sistem primer masih hidup dan masih menerima transaksi dari jalur lain. Melakukan failover dalam kondisi itu menghasilkan dua sistem aktif dan data yang menyimpang.

```
# Dari lokasi ketiga, uji apakah primer benar-benar mati
ping 10.20.30.50
telnet 10.20.30.50 23
```

Menit 5 sampai 15 — Deklarasi

1. Eskalasi ke pejabat berwenang
2. Deklarasi bencana, dicatat waktunya
3. Aktifkan tim tanggap sesuai daftar kontak
4. Tetapkan komandan insiden

Menit 15 sampai 30 — Penilaian kehilangan data

```
/* Di node DR, periksa sampai mana replikasi sampai */
DSPCRGINF CLUSTER(CLSTR1) CRG(CRGDTA) DETAIL(*FULL)
WRKJRNA JRN(JRNLIB/JRNDTA)
```

```
-- Transaksi terakhir yang tercatat di DR
SELECT MAX(WAKTU) AS trx_terakhir, COUNT(*) AS jml_hari_ini
  FROM DTALIB.TRXHIST WHERE TGL_TRX = CURRENT DATE;
```

Bandingkan dengan catatan dari sistem kanal: berapa transaksi yang mereka kirim hari ini. Selisihnya adalah perkiraan kehilangan.

Menit 30 — Keputusan failover

Keputusan ini melibatkan manajemen bisnis karena ada potensi kehilangan data. Sajikan:

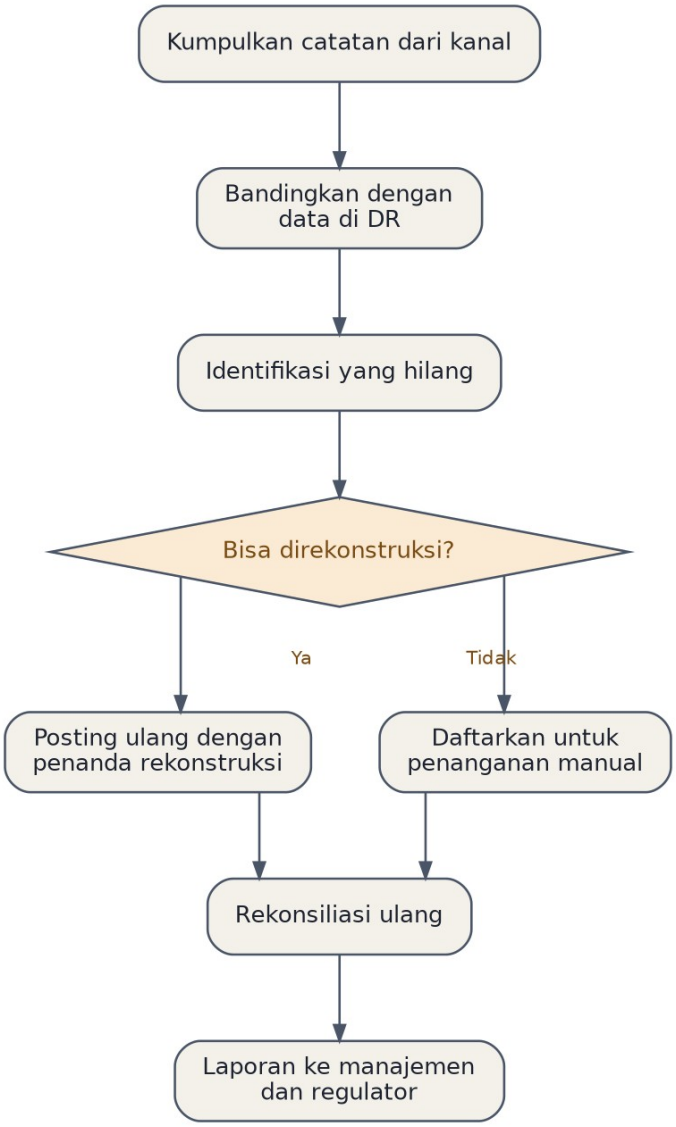
- Perkiraan jumlah transaksi yang hilang
- Perkiraan waktu pemulihan bila menunggu primer
- Perkiraan waktu buka layanan bila failover sekarang
- Konsekuensi masing-masing

Setelah keputusan — Eksekusi

```
/* Failover; sintaks bergantung konfigurasi cluster Anda */
CHGCRGPRI CLUSTER(CLSTR1) CRG(CRGDTA)

/* Bila node primer tidak responsif, mungkin perlu */
CHGCLUNODE CLUSTER(CLSTR1) NODE(NODEPRIMER) OPTION(*CHGSTS)

VRYCFG CFGOBJ(IASP01) CFGTYPE(*DEV) STATUS(*ON)
STRSBS SBSD(QGPL/EODSBS)
STRTCP
```

136.3 Rekonstruksi transaksi yang hilang

Ini pekerjaan yang paling menuntut ketelitian setelah sistem naik.



Gambar 106

```
-- Struktur untuk mencatat rekonstruksi
CREATE TABLE APPWRK.REKONSTRUKSI (
    ID          BIGINT GENERATED ALWAYS AS IDENTITY,
    WAKTU_PROSES TIMESTAMP NOT NULL,
    SUMBER      VARCHAR(30) NOT NULL,
    NO_REF_ASLI VARCHAR(40),
    NO_REK      CHAR(16),
    NOMINAL     DECIMAL(17,2),
    STATUS      CHAR(1) NOT NULL,
    KETERANGAN  VARCHAR(300),
    DISETUJUI_OLEH VARCHAR(30),
    PRIMARY KEY (ID)
);
```

Prinsip yang harus dipegang. Setiap transaksi yang diposting ulang harus tercatat sebagai rekonstruksi, dengan sumber datanya dan siapa yang menyetujui. Jangan pernah memasukkan transaksi ke sistem

produksi tanpa jejak yang bisa diaudit, betapa pun mendesaknya.

136.4 Pengembalian ke primer

Setelah sistem primer pulih, pengembalian bukan kebalikan sederhana dari failover.

Langkah	Alasan
Periksa penyebab kegagalan sudah teratasi	Jangan kembali ke sistem yang masih rusak
Sinkronkan data dari DR ke primer	Primer tertinggal sejak failover
Verifikasi sinkronisasi lengkap	Sebelum apa pun dialihkan
Jadwalkan jendela terencana	Pengembalian adalah switchover, bukan darurat
Jalankan prosedur switchover normal	Termasuk seluruh verifikasi

Jangan terburu-buru kembali. Sistem yang berjalan stabil di DR lebih baik daripada pengembalian tergesa yang menimbulkan insiden kedua. Banyak institusi berjalan di DR selama berminggu-minggu sampai jendela yang tepat tersedia.

💾 Bab 137. Dokumen Pemulihan yang Harus Dicetak

Satu berkas, dicetak, salinannya di lokasi cadangan dan di rumah personel kunci. Isinya:

```
DOKUMEN PEMULIHAN DARURAT
Versi: [   ]   Tanggal: [   ]   Halaman: [   ] dari [   ]

1. KONTAK
   Peran                  Nama        Telepon      Cadangan
   -------------------------------------------------------
   Komandan insiden
   System Engineer
   Application Support
   Manajer TI
   Direktur
   IBM support (nomor kontrak:        )
   Vendor aplikasi (nomor kontrak:    )
   Penyedia jaringan

2. IDENTITAS SISTEM
   Sistem      Serial    Model    Rilis    Lokasi
   ----------------------------------------------
   Primer
   DR

3. AKSES DARURAT
   Cara memperoleh kredensial darurat: [prosedur, BUKAN kata sandinya]
   Lokasi fisik konsol:
   Akses HMC:

4. KONFIGURASI JARINGAN
   Sistem   Alamat IP   Subnet   Gateway   VLAN
   ---------------------------------------------
   Primer
   DR
   Virtual IP layanan

5. URUTAN FAILOVER
   [langkah demi langkah, dengan perintah persis]
```

```
6. ANGKA KONTROL YANG DIVERIFIKASI
   [daftar ukuran dan cara menghitungnya]

7. DAFTAR KANAL YANG DIUJI
   [kanal, cara uji, penanggung jawab, nomor telepon]

8. MITRA YANG DIBERI TAHU
   [nama mitra, kontak, apa yang perlu disampaikan]

9. KEWAJIBAN PELAPORAN
   [regulator, tenggat, siapa yang melapor]

10. RIWAYAT REVISI
```

Dokumen ini diperbarui setiap kali konfigurasi berubah, dan diverifikasi kelengkapannya setiap latihan DR. Dokumen yang hanya ada di dalam sistem yang sedang mati tidak membantu siapa pun — itulah alasan keberadaan versi cetaknya, betapa pun kunonya terdengar.

🧮 Bagian XLI — Algoritma Operasional

Bab-bab sebelumnya menyajikan potongan kode dalam bahasa tertentu. Bagian ini menyajikan logikanya sebagai algoritma, terlepas dari bahasa, sehingga bisa diterapkan di RPG, CL, Python, atau apa pun yang Anda pakai. Bentuk ini juga memudahkan peninjauan: seorang auditor atau rekan yang tidak menguasai RPG tetap bisa memeriksa apakah logikanya benar.

🌙 Bab 138. Algoritma Pemrosesan Batch

138.1 Kenapa bentuk algoritma penting

Ada jarak yang sering diabaikan antara memahami sebuah prosedur dan mampu menuliskannya dengan benar. Seseorang bisa menjelaskan dengan lancar bahwa fase akrual tidak boleh dijalankan dua kali, lalu menulis program yang tidak punya satu pun mekanisme untuk mencegahnya. Jarak itu muncul karena pemahaman tersimpan sebagai narasi, sementara program menuntut keputusan eksplisit pada setiap percabangan: apa yang terjadi bila penanda tidak ditemukan, apa yang terjadi bila penanda ditemukan tetapi statusnya tidak dikenali, apa yang terjadi bila proses berhenti persis di antara penulisan penanda dan commit.

Menuliskan logika sebagai algoritma memaksa jarak itu tertutup. Setiap cabang harus punya tujuan, setiap kondisi harus punya penanganan, dan setiap asumsi yang tidak divalidasi akan terlihat sebagai lubang di dalam struktur. Inilah alasan bagian ini ada: bukan untuk mengajari cara menulis kode, melainkan untuk menyediakan kerangka yang bisa diperiksa sebelum kode ditulis.

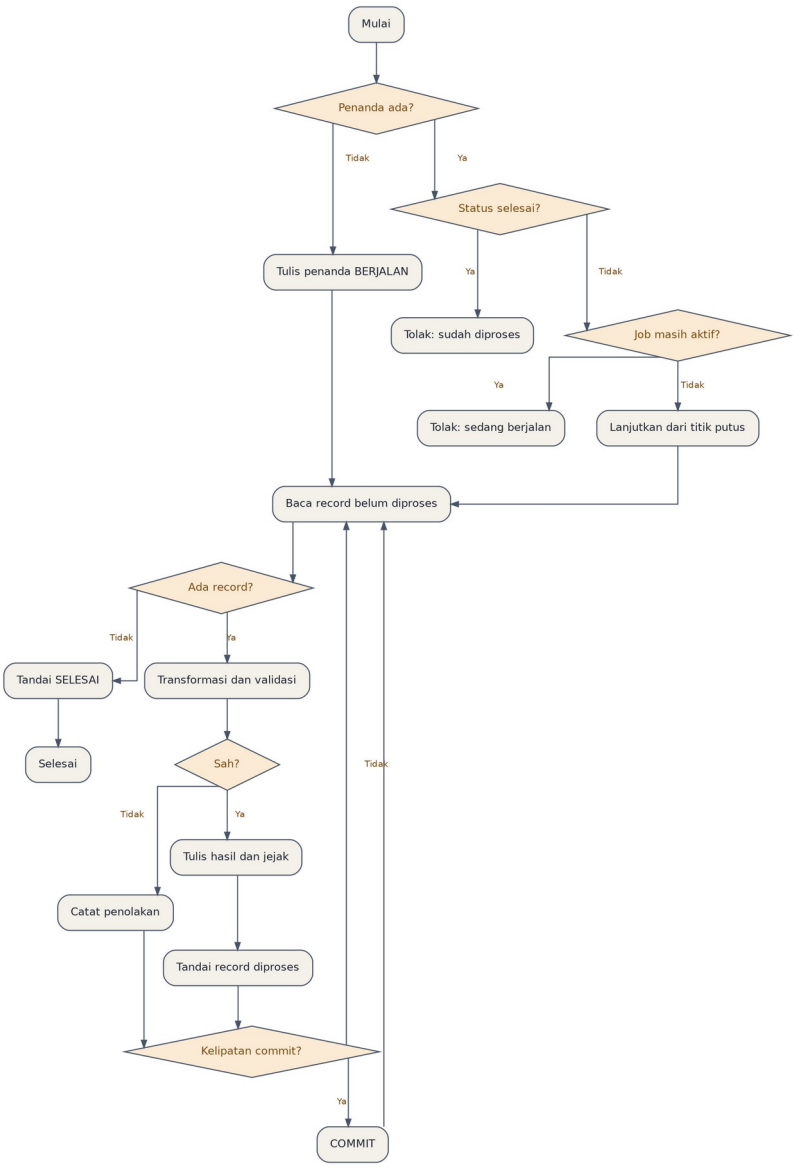
138.2 Algoritma: pemrosesan batch idempoten

```
ALGORITMA ProsesBatchIdempoten
MASUKAN   kdBatch, tglBuku, ukuranCommit
KELUARAN  jmlProses, jmlLolos, jmlTolak

1.  MULAI
2.  penanda <- BacaPenanda(kdBatch, tglBuku)
```

```
3.  JIKA penanda ADA DAN penanda.status = SELESAI MAKA
4.      LEMPAR Kesalahan('Batch sudah diproses')
5.  AKHIR JIKA
6.  JIKA penanda ADA DAN penanda.status = BERJALAN MAKA
7.      JIKA penanda.job MASIH AKTIF MAKA
8.          LEMPAR Kesalahan('Batch sedang diproses job lain')
9.      LAIN
10.         CATAT('Melanjutkan batch yang terputus')
11.     AKHIR JIKA
12. LAIN
13.     TulisPenanda(kdBatch, tglBuku, BERJALAN)
14.     COMMIT
15. AKHIR JIKA
16.
17. jmlProses <- 0 ; jmlLolos <- 0 ; jmlTolak <- 0
18. BUKA kursor ATAS sumber DENGAN FILTER belumDiproses
19. SELAMA ADA record BERIKUTNYA LAKUKAN
20.     record <- AmbilBerikutnya()
21.     jmlProses <- jmlProses + 1
22.     COBA
23.         hasil <- Transformasi(record)
24.         Validasi(hasil)
25.         Tulis(hasil)
26.         TulisJejak(record.kunci, hasil.kunci)
27.         TandaiDiproses(record.kunci)
28.         jmlLolos <- jmlLolos + 1
29.     TANGKAP kesalahan
30.         ROLLBACK KE titikAmanTerakhir
31.         TulisPenolakan(record.kunci, kesalahan)
32.         jmlTolak <- jmlTolak + 1
33.     AKHIR COBA
34.     JIKA jmlProses MOD ukuranCommit = 0 MAKA
35.         COMMIT
36.     AKHIR JIKA
37. AKHIR SELAMA
38. TUTUP kursor
39. COMMIT
40.
41. PerbaruiPenanda(kdBatch, SELESAI, jmlProses, jmlLolos, jmlTolak)
42. COMMIT
43. KEMBALIKAN jmlProses, jmlLolos, jmlTolak
44. SELESAI
```

Baris yang menentukan keselamatannya. Baris 6 sampai 11 menangani kasus yang paling sering diabaikan: batch yang terputus di tengah, meninggalkan penanda berstatus berjalan tanpa ada job yang benar-benar berjalan. Program yang hanya memeriksa status selesai akan menolak melanjutkan dan memaksa intervensi manual pada pukul tiga pagi. Program yang tidak memeriksa apa pun akan berjalan bersamaan dengan job yang masih hidup, menghasilkan pemrosesan ganda.

Baris 27 dan 34 bersama-sama menentukan apakah proses bisa dilanjutkan setelah gagal. Penandaan per record memungkinkan filter di baris 18 melewati yang sudah selesai, dan commit berkala membatasi berapa banyak pekerjaan yang hilang bila terjadi kegagalan. Menghapus salah satunya membuat proses harus diulang dari nol setiap kali.



Gambar 107

138.3 Algoritma: pemecahan kerja paralel

```
ALGORITMA JalankanParalel
MASUKAN   namaProgram, jumlahJob, antrean
KELUARAN  status

1.  MULAI
2.  JIKA jumlahJob < 1 ATAU jumlahJob > batasAman MAKA
3.      LEMPAR Kesalahan('Jumlah job di luar batas')
4.  AKHIR JIKA
5.  BersihkanPenandaSelesai()
6.  UNTUK i DARI 1 SAMPAI jumlahJob LAKUKAN
7.      KirimJob(namaProgram, parameter=(i, jumlahJob), antrean)
8.  AKHIR UNTUK
9.
10. waktuMulai <- Sekarang()
11. SELAMA BENAR LAKUKAN
12.     selesai <- HitungPenandaSelesai()
13.     gagal   <- HitungPenandaGagal()
```

```
14.     JIKA gagal > 0 MAKA
15.         HentikanSeluruhJob()
16.         LEMPAR Kesalahan('Satu atau lebih job paralel gagal')
17.     AKHIR JIKA
18.     JIKA selesai = jumlahJob MAKA
19.         KELUAR DARI SELAMA
20.     AKHIR JIKA
21.     JIKA Sekarang() - waktuMulai > batasWaktu MAKA
22.         HentikanSeluruhJob()
23.         LEMPAR Kesalahan('Timeout menunggu job paralel')
24.     AKHIR JIKA
25.     Tidur(15 detik)
26. AKHIR SELAMA
27. KEMBALIKAN SUKSES
28. SELESAI
```

Tiga pengaman yang membedakannya dari perulangan tunggu biasa. Pertama, baris 14 sampai 17 mendeteksi kegagalan salah satu job dan menghentikan sisanya, alih-alih menunggu selamanya job yang tidak akan pernah selesai. Kedua, baris 21 sampai 24 memberi batas waktu absolut. Ketiga, baris 5 membersihkan penanda dari eksekusi sebelumnya — tanpa itu, penanda lama membuat program mengira pekerjaan sudah selesai padahal belum dimulai.

Saya menempatkan penekanan pada baris 5 karena kesalahan ini punya sifat yang khas: ia tidak pernah muncul pada eksekusi pertama, hanya pada eksekusi kedua setelah kegagalan. Artinya ia lolos dari pengujian normal dan baru muncul di produksi, tepat pada malam ketika Anda sedang menangani kegagalan lain.

138.4 Algoritma: penentuan tindakan setelah fase gagal

```
ALGORITMA TentukanTindakan
MASUKAN   fase, statusHasil, adaPenanda, adaHasilTersimpan
KELUARAN  tindakan, perluPersetujuanGanda

1.  MULAI
2.  JIKA fase.sifat = BACA_SAJA MAKA
3.      KEMBALIKAN (RERUN, TIDAK)
4.  AKHIR JIKA
5.
6.  JIKA fase.idempoten = BENAR MAKA
7.      KEMBALIKAN (RERUN, TIDAK)
8.  AKHIR JIKA
9.
10. JIKA adaPenanda = BENAR MAKA
11.     JIKA statusHasil = SEBAGIAN_TERPOSTING MAKA
12.         JIKA VerifikasiNilai(sampel) = BENAR MAKA
13.             KEMBALIKAN (LANJUT_DARI_PENANDA, YA)
14.         LAIN
15.             KEMBALIKAN (REVERSAL_LALU_ULANG, YA)
16.         AKHIR JIKA
17.     LAIN
18.         KEMBALIKAN (RERUN, YA)
19.     AKHIR JIKA
20. AKHIR JIKA
21.
22. JIKA adaHasilTersimpan = BENAR MAKA
23.     KEMBALIKAN (REVERSAL_LALU_ULANG, YA)
24. AKHIR JIKA
25.
26. KEMBALIKAN (RESTORE, YA)
```

```
27. SELESAI
```

Algoritma ini adalah bentuk formal dari matriks keputusan di Bab 125.3. Menuliskannya begini memperlihatkan sesuatu yang tidak langsung terlihat dari tabel: **setiap jalur yang berakhir di restore adalah akibat dari ketiadaan sesuatu** — tidak ada penanda, tidak ada hasil tersimpan. Restore bukan pilihan yang dipilih, melainkan yang tersisa ketika pilihan lain telah ditutup oleh keputusan desain yang dibuat bertahun-tahun sebelumnya.

Konsekuensinya praktis dan langsung. Setiap kali Anda meninjau program fase akrual dan menemukan bahwa hasil perhitungannya tidak disimpan di mana pun, Anda sedang melihat keputusan yang kelak memaksa restore pada malam ketika sesuatu gagal. Memperbaikinya sekarang, saat tidak ada tekanan, jauh lebih murah daripada menghadapi konsekuensinya nanti.

🧮 Bab 139. Algoritma Diagnosis

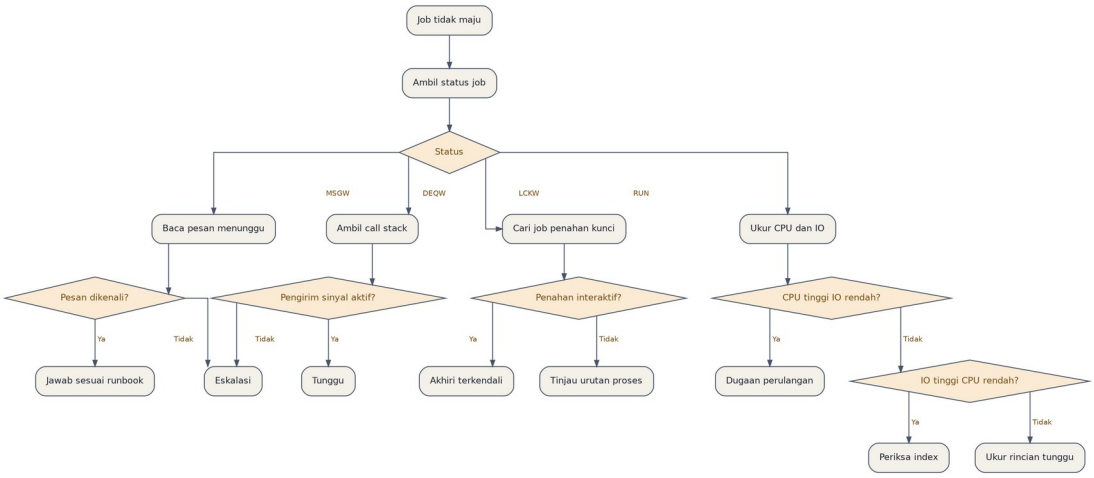
139.1 Algoritma: diagnosis job yang tidak maju

```
ALGORITMA DiagnosisJobMacet
MASUKAN   namaJob
KELUARAN  penyebab, tindakan

1.  MULAI
2.  status <- AmbilStatusJob(namaJob)
3.
4.  KASUS status DARI
5.    MSGW:
6.      pesan <- AmbilPesanMenunggu(namaJob)
7.      JIKA pesan.jenis DIKENALI MAKA
8.          KEMBALIKAN (pesan.id, JawabSesuaiRunbook(pesan.id))
9.      LAIN
10.         KEMBALIKAN (pesan.id, ESKALASI)
11.     AKHIR JIKA
12.
13.   LCKW:
14.     penahan <- CariJobPenahanKunci(namaJob)
15.     JIKA penahan.jenis = INTERAKTIF MAKA
16.         KEMBALIKAN ('Sesi tertinggal', AKHIRI_TERKENDALI(penahan))
17.     LAIN
18.         KEMBALIKAN ('Kontensi antar batch', TINJAU_URUTAN_PROSES)
19.     AKHIR JIKA
20.
21.   DEQW:
22.     tumpukan <- AmbilCallStack(namaJob)
23.     JIKA tumpukan MENGANDUNG TerimaAntrean MAKA
24.         pengirim <- CariJobPengirim()
25.         JIKA pengirim TIDAK AKTIF MAKA
26.             KEMBALIKAN ('Pengirim sinyal mati', ESKALASI)
27.         LAIN
28.             KEMBALIKAN ('Menunggu wajar', TUNGGU)
29.         AKHIR JIKA
30.     AKHIR JIKA
31.
32.   RUN:
33.     cpu <- AmbilPemakaianCPU(namaJob)
34.     io  <- AmbilJumlahIO(namaJob)
35.     JIKA cpu TINGGI DAN io RENDAH MAKA
```

```
36.         KEMBALIKAN ('Kemungkinan perulangan', TINJAU_PROGRAM)
37.     AKHIR JIKA
38.     JIKA io TINGGI DAN cpu RENDAH MAKA
39.         KEMBALIKAN ('Kemungkinan table scan', PERIKSA_INDEX)
40.     AKHIR JIKA
41.     KEMBALIKAN ('Berjalan normal tapi lambat', UKUR_TUNGGU)
42. AKHIR KASUS
43. SELESAI
```

Nilai algoritma ini bukan pada kecanggihannya, melainkan pada keterurutannya. Di bawah tekanan, orang cenderung melompat ke dugaan favoritnya — biasanya dugaan yang terakhir kali terbukti benar. Struktur berbasis status memaksa langkah pertama menjadi pengamatan, bukan hipotesis. Dan pengamatan itu hanya menuntut satu kueri yang bisa dijalankan dalam lima detik.

Saya sarankan mencetak algoritma ini dan menempelkannya di dekat konsol operasi. Pada pukul tiga pagi, empat cabang sederhana yang dibaca dari kertas lebih berguna daripada pengetahuan mendalam yang sulit diakses karena kantuk dan tekanan.



Gambar 108

139.2 Algoritma: menelusuri nilai yang salah

```
ALGORITMA TelusuriNilaiSalah
MASUKAN   tabel, kunci, kolom, nilaiSalah
KELUARAN  programPenyebab, waktuKejadian

1.  MULAI
2.  entri <- BacaJournal(tabel, kunci, rentangWaktu)
3.  JIKA entri KOSONG MAKA
4.      JIKA TabelDijurnal(tabel) = SALAH MAKA
5.          KEMBALIKAN ('Tidak dapat ditelusuri: tabel tidak dijurnal', NULL)
6.      LAIN
7.          KEMBALIKAN ('Nilai tidak pernah berubah di rentang ini', NULL)
8.      AKHIR JIKA
9.  AKHIR JIKA
10.
11. UNTUK SETIAP e DALAM entri DIURUTKAN MENURUT waktu LAKUKAN
12.     JIKA e.nilaiSesudah[kolom] = nilaiSalah MAKA
13.         KEMBALIKAN (e.program, e.waktu)
14.     AKHIR JIKA
15. AKHIR UNTUK
16.
17. KEMBALIKAN ('Berasal dari penulisan awal record', entri[1].waktu)
18. SELESAI
```

Baris 4 dan 5 layak diperhatikan karena menjawab pertanyaan sebelum pertanyaan itu sempat membuang waktu. Ketika penelusuran journal tidak menghasilkan apa-apa, ada dua kemungkinan yang sangat berbeda: perubahan memang tidak pernah terjadi, atau perubahan terjadi tetapi tidak tercatat. Membedakan keduanya lebih dulu mencegah kesimpulan yang keliru, dan sekaligus mengungkap masalah yang lebih besar — tabel produksi tanpa journaling.

139.3 Algoritma: evaluasi ambang dengan penekanan berulang

```
ALGORITMA EvaluasiAmbang
MASUKAN   metrik, nilai, definisiAmbang, riwayatPeringatan
KELUARAN  perluKirim, tingkat

1.  MULAI
2.  JIKA definisiAmbang TIDAK ADA MAKA KEMBALIKAN (TIDAK, INFO)
3.  JIKA definisiAmbang.aktif = SALAH MAKA KEMBALIKAN (TIDAK, INFO)
```

```
4.  JIKA Sekarang() DI LUAR definisiAmbang.jendelaJam MAKA
5.      KEMBALIKAN (TIDAK, INFO)
6.  AKHIR JIKA
7.
8.  JIKA definisiAmbang.arah = NAIK_BURUK MAKA
9.      JIKA nilai >= definisiAmbang.kritis MAKA tingkat <- KRITIS
10.     LAIN JIKA nilai >= definisiAmbang.perhatian MAKA tingkat <- PERHATIAN
11.     LAIN KEMBALIKAN (TIDAK, NORMAL)
12.     AKHIR JIKA
13. LAIN
14.     JIKA nilai <= definisiAmbang.kritis MAKA tingkat <- KRITIS
15.     LAIN JIKA nilai <= definisiAmbang.perhatian MAKA tingkat <- PERHATIAN
16.     LAIN KEMBALIKAN (TIDAK, NORMAL)
17.     AKHIR JIKA
18. AKHIR JIKA
19.
20. terakhir <- PeringatanTerakhir(metrik, riwayatPeringatan)
21. JIKA terakhir TIDAK ADA MAKA KEMBALIKAN (YA, tingkat)
22. JIKA tingkat LEBIH TINGGI DARI terakhir.tingkat MAKA
23.     KEMBALIKAN (YA, tingkat)
24. AKHIR JIKA
25. JIKA Sekarang() - terakhir.waktu > jedaPenekanan MAKA
26.     KEMBALIKAN (YA, tingkat)
27. AKHIR JIKA
28. KEMBALIKAN (TIDAK, tingkat)
29. SELESAI
```

Baris 22 dan 23 adalah pengecualian yang menyelamatkan sistem pemantauan dari kelemahan terbesarnya sendiri. Penekanan peringatan berulang dibuat agar tim tidak dibanjiri, tetapi penekanan yang diterapkan tanpa pengecualian akan menyembunyikan perburukan. Sebuah metrik yang naik dari perhatian menjadi kritis pada menit keempat puluh adalah informasi baru yang mendesak, dan ia tidak boleh ditelan oleh aturan yang dirancang untuk menyaring pengulangan.

🧮 Bab 140. Algoritma Verifikasi

140.1 Algoritma: rekonsiliasi setelah pemulihan

```
ALGORITMA VerifikasiPemulihan
MASUKAN   daftarUkuran, nilaiSebelum
KELUARAN  lulus, daftarSelisih

1.  MULAI
2.  daftarSelisih <- KOSONG
3.  UNTUK SETIAP u DALAM daftarUkuran LAKUKAN
4.      sesudah <- HitungUkuran(u)
5.      selisihNilai  <- nilaiSebelum[u].nilai  - sesudah.nilai
6.      selisihJumlah <- nilaiSebelum[u].jumlah - sesudah.jumlah
7.      JIKA u.jenis = KEUANGAN MAKA
8.          toleransi <- 0
9.      LAIN
10.         toleransi <- u.toleransi
11.     AKHIR JIKA
12.     JIKA ABS(selisihNilai) > toleransi ATAU selisihJumlah <> 0 MAKA
13.         TAMBAHKAN (u, selisihNilai, selisihJumlah) KE daftarSelisih
14.     AKHIR JIKA
15. AKHIR UNTUK
16.
17. JIKA daftarSelisih KOSONG MAKA
```

```
18.     CatatBeritaAcara(LULUS, daftarUkuran)
19.     KEMBALIKAN (BENAR, KOSONG)
20. LAIN
21.     CatatBeritaAcara(GAGAL, daftarSelisih)
22.     KEMBALIKAN (SALAH, daftarSelisih)
23. AKHIR JIKA
24. SELESAI
```

Baris 7 sampai 11 memisahkan ukuran keuangan dari ukuran lain, dan menetapkan toleransi nol untuk yang pertama. Pemisahan ini bukan kehati-hatian berlebihan. Selisih pada jumlah record laporan bisa punya penjelasan yang sah — misalnya baris yang memang tidak direstore karena berada di luar cakupan. Selisih pada total saldo tidak pernah punya penjelasan sah. Uang tidak berkurang karena proses pemulihan; kalau angkanya berbeda, ada yang salah dan ia harus ditemukan sebelum sistem dinyatakan pulih.

140.2 Algoritma: gerbang kesiapan

```
ALGORITMA GerbangKesiapan
MASUKAN   daftarPemeriksaan
KELUARAN  bolehLanjut, daftarGagal

1.  MULAI
2.  daftarGagal <- KOSONG
3.  UNTUK SETIAP p DALAM daftarPemeriksaan LAKUKAN
4.      COBA
5.          hasil <- p.jalankan()
6.      TANGKAP kesalahan
7.          hasil <- GAGAL
8.      AKHIR COBA
9.      JIKA hasil <> LULUS MAKA
10.         TAMBAHKAN p KE daftarGagal
11.     AKHIR JIKA
12. AKHIR UNTUK
13.
14. JIKA daftarGagal KOSONG MAKA
15.     KEMBALIKAN (BENAR, KOSONG)
16. LAIN
17.     CatatPenolakan(daftarGagal)
18.     BeritahuOperator(daftarGagal)
19.     KEMBALIKAN (SALAH, daftarGagal)
20. AKHIR JIKA
21. SELESAI
```

Baris 6 sampai 8 menetapkan prinsip yang layak diterapkan di seluruh sistem Anda: **pemeriksaan yang gagal dijalankan diperlakukan sebagai pemeriksaan yang tidak lulus.** Pilihan sebaliknya — menganggap pemeriksaan yang bermasalah sebagai lulus agar proses tetap berjalan — terasa pragmatis pada saat ditulis, dan akan membuka gerbang tepat pada malam ketika sesuatu benar-benar rusak sampai pemeriksaannya pun ikut gagal.

Pola pikir ini punya nama di dunia rekayasa keandalan: gagal dalam keadaan aman. Di lingkungan perbankan, keadaan aman hampir selalu berarti berhenti dan meminta manusia memutuskan, bukan melanjutkan dengan asumsi optimis.

140.3 Algoritma: deteksi degradasi bertahap

```
ALGORITMA DeteksiDegradasi
```

```
MASUKAN   metrik, jendelaBaseline, ambangRasio, minimalSampel
KELUARAN  terdegradasi, rasio

1.  MULAI
2.  sampel <- AmbilRiwayat(metrik, jendelaBaseline, jamYangSama)
3.  JIKA JUMLAH(sampel) < minimalSampel MAKA
4.      KEMBALIKAN (TIDAK_DAPAT_DINILAI, NULL)
5.  AKHIR JIKA
6.
7.  rata      <- RataRata(sampel)
8.  simpangan <- SimpanganBaku(sampel)
9.  kini      <- NilaiTerkini(metrik)
10.
11. JIKA rata = 0 MAKA KEMBALIKAN (TIDAK_DAPAT_DINILAI, NULL)
12. rasio <- kini / rata
13.
14. JIKA simpangan > 0 MAKA
15.     skor <- (kini - rata) / simpangan
16.     JIKA skor > 3 DAN rasio > ambangRasio MAKA
17.         KEMBALIKAN (BENAR, rasio)
18.     AKHIR JIKA
19. LAIN
20.     JIKA rasio > ambangRasio MAKA KEMBALIKAN (BENAR, rasio)
21. AKHIR JIKA
22.
23. KEMBALIKAN (SALAH, rasio)
24. SELESAI
```

Algoritma ini menjawab kelas masalah yang tidak tertangkap oleh ambang tetap, dan yang menurut pengalaman lapangan paling sering berkembang menjadi insiden besar: perburukan bertahap. Durasi fase EOD yang naik satu menit setiap minggu tidak akan pernah memicu alarm apa pun sampai ia melewati batch window, dan pada saat itu perbaikannya sudah mendesak.

Baris 2 memuat detail yang mudah terlewat tetapi menentukan: baseline diambil dari jam yang sama pada hari-hari sebelumnya, bukan dari seluruh hari. Beban sistem perbankan sangat berbeda antara pukul sepuluh pagi dan pukul dua pagi. Baseline yang mencampur keduanya menghasilkan rata-rata yang tidak menggambarkan kondisi mana pun, dan simpangan baku yang begitu besar sehingga hampir tidak ada penyimpangan yang terdeteksi.

Baris 3 dan 4 menetapkan bahwa data yang tidak cukup menghasilkan jawaban tidak dapat dinilai, bukan jawaban normal. Perbedaannya penting: yang pertama memberi tahu Anda bahwa Anda belum tahu, yang kedua memberi tahu Anda bahwa semuanya baik-baik saja. Sistem pemantauan yang mengaburkan keduanya memberi rasa aman yang tidak berdasar, dan itu justru kondisi yang paling ingin Anda hindari.

💻 Bagian XLII — Pemrograman RPG dan COBOL

Bagian XI memberi primer RPG dari sisi cara membacanya. Bagian ini melengkapinya dengan referensi bahasa yang lebih padat, lalu menambahkan COBOL — bahasa yang di banyak bank Indonesia masih memegang modul-modul tertua dan paling jarang disentuh, justru karena orang yang memahaminya semakin sedikit.

Keduanya ditempatkan dalam satu bagian karena di lingkungan nyata keduanya hidup berdampingan dalam satu aplikasi. Sebuah transaksi bisa dimulai dari program CL, memanggil modul RPG untuk validasi, lalu

memanggil program COBOL warisan untuk posting ke buku besar. System Engineer yang hanya menguasai satu di antaranya akan berhenti tepat di tengah rantai pemanggilan.

💻 Bab 141. Referensi Bahasa RPG

141.1 Operasi dan padanan modernnya

Banyak operasi RPG lama masih ditemui di kode produksi, sementara padanan modernnya lebih jelas dibaca. Tabel ini membantu ketika Anda menemui yang lama dan ingin memahaminya, atau menulis yang baru.

Operasi lama	Bentuk modern	Fungsi
`MOVE`	`eval` atau `%subst`	Pemindahan nilai, aturannya rumit
`MOVEL`	`eval`	Pemindahan rata kiri
`Z-ADD`	`eval`	Set nilai numerik
`ADD` / `SUB`	`+=` / `-=`	Aritmetika
`MULT` / `DIV`	`*` / `/`	Aritmetika
`MVR`	`%rem()`	Sisa pembagian
`COMP`	`if`	Perbandingan
`CABxx`	`if` dan `leave`	Bandingkan lalu lompat
`GOTO` / `TAG`	struktur blok	Lompatan, hindari di kode baru
`DO` / `ENDDO`	`for`	Perulangan berhitung
`DOW` / `DOU`	`dow` / `dou`	Perulangan bersyarat
`SETON` / `SETOFF`	`*inxx = *on`	Indikator
`EXSR`	panggilan prosedur	Subrutin
`CALL` / `PARM`	prototipe dan `callp`	Pemanggilan program
`KLIST` / `KFLD`	daftar kunci inline	Kunci gabungan
`LOKUP`	`%lookup()`	Pencarian dalam array
`TESTN`	`%check()`	Uji isi numerik

Catatan tentang `MOVE`. Operasi ini punya aturan konversi implisit yang berbeda-beda menurut tipe operand, dan itulah sumber banyak bug warisan yang sulit dilacak. `MOVE` dari numerik ke karakter, dari karakter panjang ke pendek, atau antara tanggal dan karakter, semuanya berperilaku berbeda. Ketika Anda menemui `MOVE` di kode yang sedang diselidiki, jangan mengasumsikan artinya — periksa tipe kedua operandnya lebih dulu.

141.2 Built-in function yang paling sering dipakai

Fungsi	Kegunaan
`%trim` `%triml` `%trimr`	Hapus spasi
`%subst`	Ambil sebagian teks
`%len`	Panjang
`%size`	Ukuran definisi
`%scan`	Cari posisi
`%scanrpl`	Cari dan ganti
`%replace`	Ganti sebagian
`%char` `%dec` `%int` `%float`	Konversi tipe

Fungsi	Kegunaan
`%dech`	Konversi dengan pembulatan setengah ke atas
`%editc` `%editw`	Format keluaran
`%date` `%time` `%timestamp`	Konversi waktu
`%days` `%months` `%years`	Durasi
`%diff`	Selisih dua tanggal
`%found` `%eof` `%equal`	Hasil operasi file
`%error` `%status`	Status kesalahan
`%open`	Apakah file terbuka
`%parms`	Jumlah parameter dikirim
`%addr` `%paddr`	Penunjuk
`%elem`	Jumlah elemen array
`%lookup`	Cari dalam array
`%xlate`	Terjemahkan karakter
`%check` `%checkr`	Validasi isi
`%null`	Uji nilai null

`%dech` layak diperhatikan khusus di konteks keuangan. Pembagian biasa di RPG memotong, bukan membulatkan. Untuk perhitungan bunga dan biaya, pemotongan yang berulang jutaan kali menghasilkan selisih yang terakumulasi — persis kasus 15 di Bab 122.2.

```
**free
dcl-s wHasil packed(17:2);
dcl-s wPokok packed(17:2) inz(1000000);
dcl-s wRate  packed(9:6)  inz(0.055000);

// Memotong: 150.68
wHasil = wPokok * wRate / 365;

// Membulatkan: 150.68 atau 150.69 sesuai pecahannya
wHasil = %dech(wPokok * wRate / 365: 17: 2);
```

141.3 Pola penanganan file yang aman

```
**free
ctl-opt dftactgrp(*no) actgrp('APPGRP');
dcl-f REKENING usage(*update) keyed;

dcl-proc perbaruiSaldo;
  dcl-pi *n ind;
    pNoRek  char(16)      const;
    pJumlah packed(17:2)  const;
  end-pi;

  chain(e) pNoRek REKENING;

  if %error();
     select;
       when %status() = 1218;
         kirimPesan('Record dipakai job lain: ' + pNoRek);
       when %status() = 1222;
         kirimPesan('Timeout menunggu record: ' + pNoRek);
       other;
```

```
        kirimEscape('Kesalahan baca: ' + %char(%status()));
     endsl;
     return *off;
  endif;

  if not %found(REKENING);
     return *off;
  endif;

  if SALDO + pJumlah < 0;
     unlock REKENING;          // WAJIB: lepaskan kunci sebelum keluar
     return *off;
  endif;

  SALDO += pJumlah;
  update(e) REKENINGR;

  if %error();
     kirimEscape('Kesalahan tulis: ' + %char(%status()));
  endif;

  return *on;
end-proc;
```

Baris `unlock` di jalur penolakan adalah baris yang paling sering hilang dari kode nyata. Program mengembalikan penolakan, pemanggil melanjutkan ke record berikutnya, dan record yang ditolak tetap terkunci sampai operasi file berikutnya. Di fase posting yang menolak ribuan record, akibatnya adalah kontensi yang tampak tanpa sebab.

141.4 Kompilasi

```
/* RPG tanpa SQL */
CRTBNDRPG PGM(APPPGM/HITBUNGA) +
          SRCSTMF('/src/rpgle/hitbunga.rpgle') +
          DBGVIEW(*SOURCE) OPTION(*EVENTF) +
          DFTACTGRP(*NO) ACTGRP('APPGRP') +
          BNDDIR(APPPGM/APPBND) REPLACE(*YES)

/* RPG dengan embedded SQL */
CRTSQLRPGI OBJ(APPPGM/CEKSALDO) +
           SRCSTMF('/src/sqlrpgle/ceksaldo.sqlrpgle') +
           OBJTYPE(*PGM) +
           DBGVIEW(*SOURCE) +
           COMMIT(*CHG) +
           CLOSQLCSR(*ENDMOD) +
           RPGPPOPT(*LVL2) +
           REPLACE(*YES)
```

Parameter	Arti	Nilai yang disarankan
`COMMIT`	Tingkat penguncian SQL	`*CHG` untuk transaksional
`CLOSQLCSR`	Kapan kursor ditutup	`*ENDMOD` untuk modul ILE
`RPGPPOPT`	Praproses `/copy` untuk SQL	`*LVL2` bila copybook memuat SQL
`DBGVIEW`	Tampilan debug	`*SOURCE`, atau `*LIST` bila banyak `/copy`
`OPTION(*EVENTF)`	Berkas event kesalahan	Selalu, untuk alat editor modern

`CLOSQLCSR(*ENDMOD)` sering salah disetel. Nilai bawaannya menutup kursor saat activation group

berakhir, bukan saat modul berakhir. Pada program yang dipanggil berulang dalam satu job, kursor yang tidak tertutup menumpuk dan akhirnya menghabiskan batas.

💻 Bab 142. ILE COBOL untuk IBM i

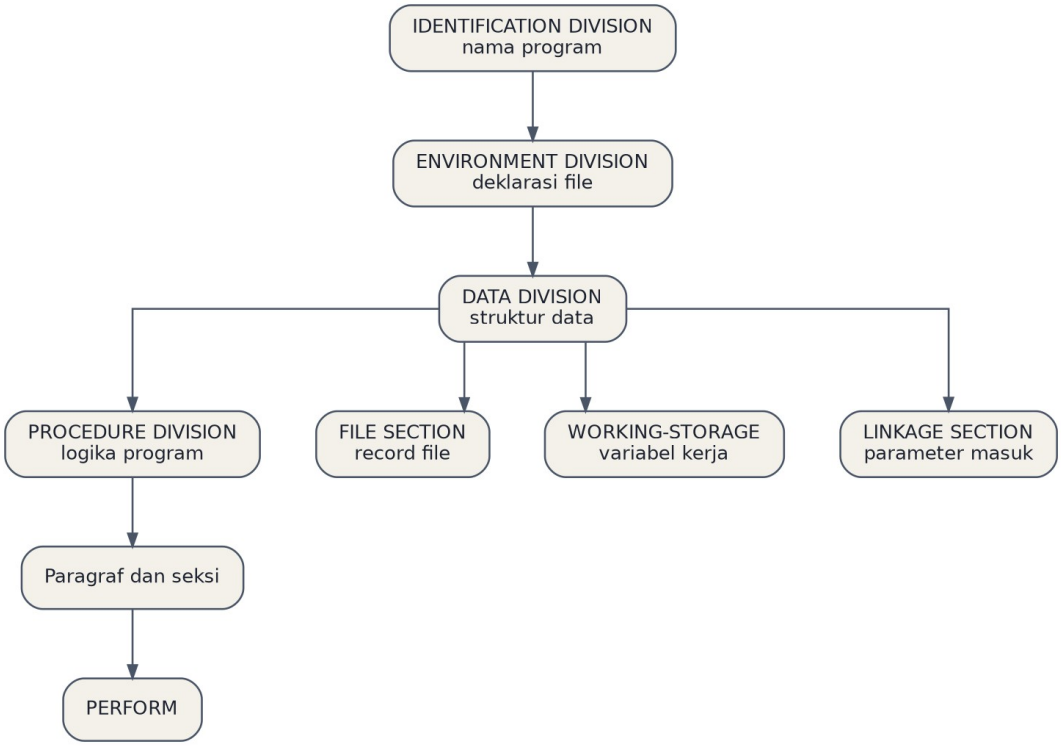
142.1 Kenapa COBOL masih penting

Di banyak bank, modul yang menangani buku besar, perhitungan bunga historis, atau antarmuka ke sistem pembayaran nasional ditulis dalam COBOL pada 1990-an dan belum pernah ditulis ulang. Alasannya sederhana dan masuk akal: modul itu bekerja, jarang berubah, dan risiko menyentuhnya jauh melebihi manfaat kerapian.

Konsekuensinya bagi System Engineer adalah kenyataan yang tidak bisa dihindari. Ketika modul itu gagal, tidak ada gunanya berargumen bahwa seharusnya sudah dimodernisasi. Anda perlu bisa membaca kodenya, memahami alur datanya, dan menelusuri di mana nilai yang salah berasal. Bab ini menyediakan bekal minimum untuk itu.

142.2 Struktur program COBOL

COBOL disusun dalam empat divisi yang urutannya mengikat.



Gambar 109

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HITBUNGA.
       AUTHOR. TIM APLIKASI.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT REKENING-FILE
        ASSIGN TO DATABASE-REKENING
        ORGANIZATION IS INDEXED
        ACCESS MODE IS DYNAMIC
        RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
        FILE STATUS IS WS-FILE-STATUS.

DATA DIVISION.
FILE SECTION.
FD  REKENING-FILE.
01  REKENING-REC.
    COPY DDS-ALL-FORMATS OF REKENING.

WORKING-STORAGE SECTION.
01  WS-FILE-STATUS      PIC X(2).
01  WS-SALDO            PIC S9(15)V99 PACKED-DECIMAL.
01  WS-RATE             PIC S9(3)V9(6) PACKED-DECIMAL.
01  WS-BUNGA            PIC S9(15)V99 PACKED-DECIMAL.
01  WS-HITUNG           PIC S9(20)V9(10) PACKED-DECIMAL.
01  WS-HARI-SETAHUN     PIC S9(3) VALUE 365.
01  WS-JML-PROSES       PIC S9(9) PACKED-DECIMAL VALUE ZERO.
01  WS-SELESAI          PIC X VALUE 'N'.
    88  SUDAH-SELESAI   VALUE 'Y'.

LINKAGE SECTION.
01  LK-TGL-BUKU         PIC X(10).
01  LK-KODE-HASIL       PIC X(4).

PROCEDURE DIVISION USING LK-TGL-BUKU LK-KODE-HASIL.

MAIN-PROCESS.
    PERFORM BUKA-FILE
    PERFORM PROSES-SELURUH-REKENING
    PERFORM TUTUP-FILE
    MOVE '0000' TO LK-KODE-HASIL
    GOBACK.

BUKA-FILE.
    OPEN I-O REKENING-FILE
    IF WS-FILE-STATUS NOT = '00'
        MOVE 'E001' TO LK-KODE-HASIL
        PERFORM LAPOR-KESALAHAN
        GOBACK
    END-IF.

PROSES-SELURUH-REKENING.
    PERFORM UNTIL SUDAH-SELESAI
        READ REKENING-FILE NEXT RECORD
            AT END
                SET SUDAH-SELESAI TO TRUE
            NOT AT END
                PERFORM PROSES-SATU-REKENING
        END-READ
```

```
        END-PERFORM.

    PROSES-SATU-REKENING.
        IF STATUS OF REKENING-REC = 'A'
            MOVE SALDO OF REKENING-REC TO WS-SALDO
            PERFORM AMBIL-RATE
            IF WS-RATE = ZERO
                MOVE 'E002' TO LK-KODE-HASIL
                PERFORM LAPOR-KESALAHAN
                GOBACK
            END-IF
            COMPUTE WS-HITUNG = WS-SALDO * WS-RATE
            COMPUTE WS-BUNGA ROUNDED =
                    WS-HITUNG / WS-HARI-SETAHUN
            ADD WS-BUNGA TO SALDO OF REKENING-REC
            REWRITE REKENING-REC
            IF WS-FILE-STATUS NOT = '00'
                MOVE 'E003' TO LK-KODE-HASIL
                PERFORM LAPOR-KESALAHAN
                GOBACK
            END-IF
            ADD 1 TO WS-JML-PROSES
        END-IF.

    TUTUP-FILE.
        CLOSE REKENING-FILE.
```

142.3 Yang perlu diperhatikan saat membaca COBOL

Unsur	Arti	Kenapa penting
`PIC S9(15)V99`	Numerik bertanda, 15 digit, 2 desimal	Batas kapasitas field
`PACKED-DECIMAL`	Penyimpanan packed	Padanan `packed` di RPG
`COMP-3`	Sinonim packed-decimal	Bentuk lama yang sama
`V` dalam PIC	Titik desimal tersirat	Tidak disimpan, hanya posisi
`88` level	Nama kondisi	Seperti konstanta boolean
`FILE STATUS`	Kode hasil operasi file	Wajib diperiksa tiap operasi
`ROUNDED`	Pembulatan pada COMPUTE	Tanpa ini, hasilnya dipotong
`GOBACK`	Kembali ke pemanggil	Berbeda dari `STOP RUN`
`COPY DDS-ALL-FORMATS`	Ambil definisi dari tabel	Setara `likerec` di RPG
`EXTERNALLY-DESCRIBED-KEY`	Kunci dari definisi file	Tidak ditulis ulang

`ROUNDED` adalah kata kunci yang menentukan kebenaran finansial. Tanpa `ROUNDED`, `COMPUTE` memotong hasil. Ketika meninjau program COBOL yang menghitung uang, keberadaan atau ketiadaan kata ini pada setiap `COMPUTE` adalah salah satu hal pertama yang harus diperiksa.

`STOP RUN` versus `GOBACK`. `STOP RUN` mengakhiri seluruh unit eksekusi, menutup activation group, dan bisa menjatuhkan pemanggilnya. `GOBACK` hanya kembali ke pemanggil. Program yang dimaksudkan sebagai subprogram tetapi memakai `STOP RUN` akan menghentikan seluruh rantai pemanggilan — masalah yang muncul hanya ketika program itu dipanggil dari program lain, bukan saat diuji sendiri.

142.4 Kode FILE STATUS

Kode	Arti
00	Berhasil
02	Berhasil, kunci duplikat
10	Akhir file
21	Urutan kunci salah
22	Kunci duplikat, ditolak
23	Record tidak ditemukan
30	Kesalahan permanen
34	Batas ruang terlampaui
35	File tidak ditemukan saat open
37	Mode open tidak sah
41	File sudah terbuka
42	Close pada file tertutup
43	Rewrite tanpa read sebelumnya
47	Read pada file tidak dibuka input
48	Write pada file tidak dibuka output
9x	Kesalahan khusus implementasi

Kode 43 adalah padanan CPF1221 di RPG: percobaan memperbarui record tanpa membacanya lebih dulu. Kode 9x menuntut pemeriksaan pesan sistem, karena artinya bergantung pada situasi.

142.5 Embedded SQL dalam COBOL

```
      WORKING-STORAGE SECTION.
          EXEC SQL INCLUDE SQLCA END-EXEC.

      01  WS-NO-REK           PIC X(16).
      01  WS-SALDO            PIC S9(15)V99 PACKED-DECIMAL.
      01  WS-STATUS-REK       PIC X(1).

      PROCEDURE DIVISION.

      AMBIL-SALDO.
          EXEC SQL
              SELECT SALDO, STATUS
                INTO :WS-SALDO, :WS-STATUS-REK
                FROM DTALIB.REKENING
               WHERE NO_REK = :WS-NO-REK
          END-EXEC

          EVALUATE SQLCODE
              WHEN 0
                  CONTINUE
              WHEN 100
                  MOVE 'E404' TO LK-KODE-HASIL
              WHEN OTHER
                  MOVE 'E500' TO LK-KODE-HASIL
                  PERFORM LAPOR-SQL-ERROR
          END-EVALUATE.

      PROSES-DENGAN-KURSOR.
          EXEC SQL
```

```
            DECLARE CUR-REK CURSOR FOR
                SELECT NO_REK, SALDO, KD_PRODUK
                  FROM DTALIB.REKENING
                 WHERE STATUS = 'A'
                 FOR UPDATE OF SALDO
        END-EXEC

        EXEC SQL OPEN CUR-REK END-EXEC

        PERFORM UNTIL SQLCODE NOT = 0
            EXEC SQL
                FETCH CUR-REK
                 INTO :WS-NO-REK, :WS-SALDO, :WS-KD-PRODUK
            END-EXEC
            IF SQLCODE = 0
                PERFORM HITUNG-DAN-POSTING
                ADD 1 TO WS-JML-PROSES
                IF FUNCTION MOD(WS-JML-PROSES, 5000) = 0
                    EXEC SQL COMMIT END-EXEC
                END-IF
            END-IF
        END-PERFORM

        EXEC SQL CLOSE CUR-REK END-EXEC
        EXEC SQL COMMIT END-EXEC.
```

142.6 Kompilasi COBOL

```
/* COBOL tanpa SQL, langsung jadi program */
CRTBNDCBL PGM(APPPGM/HITBUNGA) +
          SRCSTMF('/src/cobol/hitbunga.cbl') +
          DBGVIEW(*SOURCE) +
          OPTION(*SRCDBG *EVENTF) +
          DFTACTGRP(*NO) ACTGRP('APPGRP') +
          REPLACE(*YES)

/* COBOL jadi modul untuk di-bind */
CRTCBLMOD MODULE(APPPGM/HITBUNGA) +
          SRCSTMF('/src/cobol/hitbunga.cbl') +
          DBGVIEW(*ALL)

/* COBOL dengan embedded SQL */
CRTSQLCBLI OBJ(APPPGM/POSTINGGL) +
           SRCSTMF('/src/cobol/postinggl.sqlcbl') +
           OBJTYPE(*PGM) +
           DBGVIEW(*SOURCE) +
           COMMIT(*CHG) +
           CLOSQLCSR(*ENDMOD) +
           REPLACE(*YES)

/* Melihat informasi program COBOL yang sudah ada */
DSPPGM PGM(APPPGM/HITBUNGA) DETAIL(*MODULE)
```

Perintah	Untuk
`CRTBNDCBL`	COBOL ILE, langsung program
`CRTCBLMOD`	COBOL ILE, jadi modul
`CRTSQLCBLI`	COBOL ILE dengan embedded SQL
`CRTCBLPGM`	COBOL OPM warisan, hindari untuk kode baru
`CRTSQLCBL`	COBOL OPM dengan SQL, warisan

Perhatikan pembedaan ILE dan OPM. Program yang dikompilasi dengan `CRTCBLPGM` berjalan di model program lama, tidak bisa memakai activation group bernama, dan tidak bisa di-bind dengan service program ILE. Bila Anda menemukan program semacam ini di produksi, ia adalah kandidat konversi — tetapi konversi itu mengubah perilaku activation group dan commitment scope, jadi perlu diuji menyeluruh.

```
-- Menemukan program OPM yang masih ada
SELECT PROGRAM_NAME, PROGRAM_LIBRARY, PROGRAM_TYPE,
       PROGRAM_CREATION_DATE, TEXT_DESCRIPTION
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND PROGRAM_TYPE = 'OPM'
 ORDER BY PROGRAM_NAME;
```

142.7 Debug program COBOL

Mekanismenya sama dengan RPG, tetapi perintah dalam sesi debug merujuk nama COBOL.

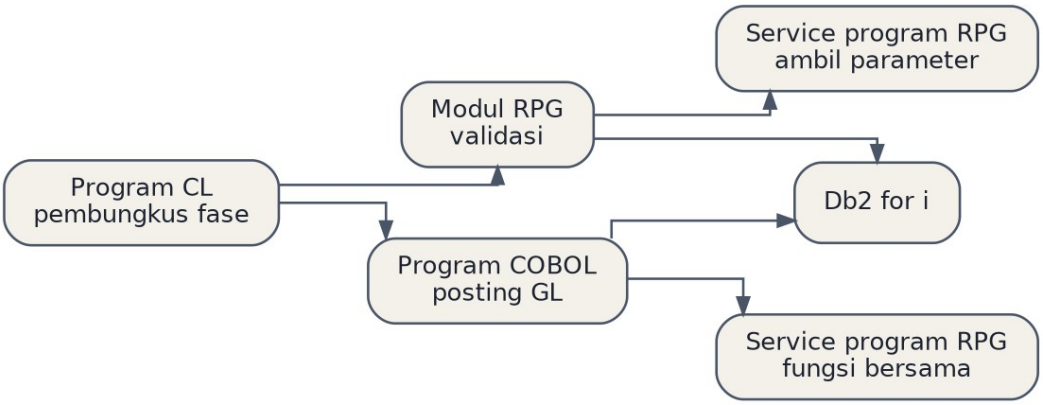
```
STRSRVJOB JOB(123456/EODUSER/EODGL)
STRDBG PGM(APPPGM/POSTINGGL) UPDPROD(*NO)
```

Perintah debug	Contoh untuk COBOL
Tampilkan variabel	`EVAL WS-SALDO`
Isi sebagian struktur	`EVAL REKENING-REC.SALDO`
Kondisi bernama level 88	`EVAL SUDAH-SELESAI`
Elemen tabel	`EVAL WS-TABEL(3)`
Sebagian teks	`EVAL WS-NAMA(1:20)`
Breakpoint bersyarat	`BREAK 145 WHEN WS-NO-REK = '1234567890123456'`
Pantau perubahan	`WATCH WS-BUNGA`

Perhatikan bahwa nama variabel COBOL memakai tanda hubung, bukan garis bawah, dan bahwa referensi sebagian teks memakai notasi `(awal:panjang)` — berbeda dari RPG yang memakai `%subst`.

Bab 143. Interoperabilitas Antar Bahasa

143.1 Rantai pemanggilan yang khas



Gambar 110

Rantai semacam ini lazim di aplikasi yang tumbuh selama dua dekade. Masalahnya muncul di batas antar bahasa: perbedaan tipe data, cara melewatkan parameter, dan cakupan commitment control.

143.2 Pemetaan tipe data

RPG	COBOL	CL	Catatan
`char(n)`	`PIC X(n)`	`*CHAR n`	Setara langsung
`packed(m:n)`	`PIC S9(m-n)V9(n) PACKED-DECIMAL`	`*DEC (m n)`	Perhatikan total digit
`zoned(m:n)`	`PIC S9(m-n)V9(n) DISPLAY`	`*DEC`	Penyimpanan berbeda
`int(10)`	`PIC S9(9) BINARY`	`*INT 4`	Ukuran harus cocok
`int(5)`	`PIC S9(4) BINARY`	`*INT 2`	
`ind`	`PIC X(1)` nilai '1'/'0'	`*LGL`	Bukan boolean sejati
`date`	`PIC X(10)` atau tipe tanggal	`*CHAR 10`	Format harus disepakati
`varchar(n)`	Group dengan panjang	—	Perlu penanganan khusus

Kesalahan pemetaan yang paling merusak adalah ketidaksesuaian pada tipe packed. `packed(17:2)` di RPG berarti 17 digit total dengan 2 desimal, yang menempati 9 byte. Padanannya di COBOL adalah `PIC S9(15)V99 PACKED-DECIMAL` — lima belas digit sebelum desimal, bukan tujuh belas. Menuliskan `PIC S9(17)V99` menghasilkan field 19 digit, ukurannya 10 byte, dan pemanggilan akan membaca memori yang salah.

Gejalanya khas dan membingungkan: nilai yang diterima program kedua tampak acak atau tergeser, tanpa pesan kesalahan apa pun. Ketika Anda menemui gejala itu di batas antar bahasa, periksa ukuran byte kedua definisi lebih dulu.

```
**free
// RPG: prototipe untuk memanggil program COBOL
dcl-pr postingGL extpgm('POSTINGGL');
  pTglBuku   char(10)     const;
  pNoRef     char(40)     const;
  pKdGL      char(10)     const;
  pDebet     packed(17:2) const;
  pKredit    packed(17:2) const;
  pKodeHasil char(4);
end-pr;

dcl-s wKodeHasil char(4);

postingGL('2026-09-18': wNoRef: 'KAS001':
          wJumlah: 0: wKodeHasil);

if wKodeHasil <> '0000';
   kirimEscape('Posting GL gagal: ' + wKodeHasil);
endif;
```

```
      * COBOL: sisi penerima
       LINKAGE SECTION.
       01  LK-TGL-BUKU     PIC X(10).
       01  LK-NO-REF       PIC X(40).
       01  LK-KD-GL        PIC X(10).
       01  LK-DEBET        PIC S9(15)V99 PACKED-DECIMAL.
       01  LK-KREDIT       PIC S9(15)V99 PACKED-DECIMAL.
       01  LK-KODE-HASIL   PIC X(4).

       PROCEDURE DIVISION USING LK-TGL-BUKU LK-NO-REF
                                LK-KD-GL LK-DEBET
                                LK-KREDIT LK-KODE-HASIL.
```

143.3 Memanggil COBOL dari CL

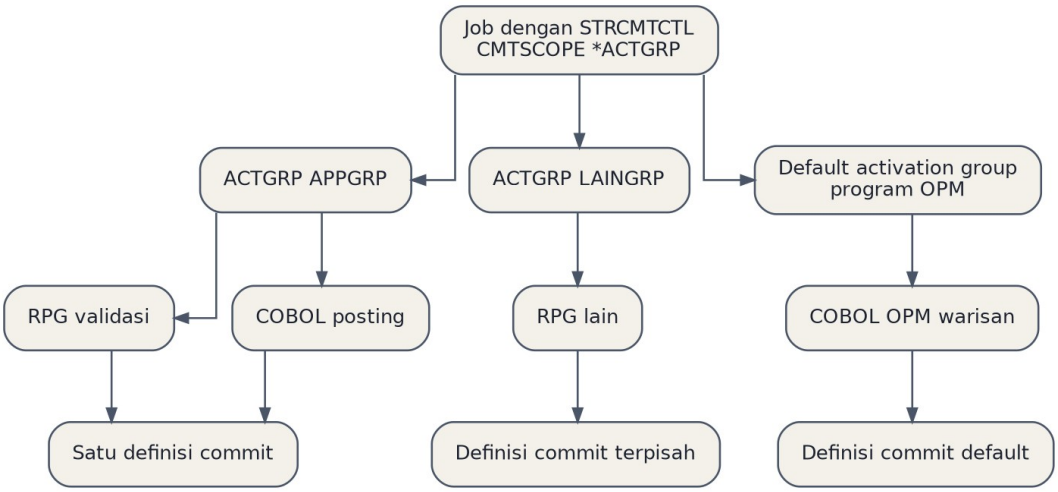
```
PGM
   DCL VAR(&TGLBUKU)  TYPE(*CHAR) LEN(10)
   DCL VAR(&NOREF)    TYPE(*CHAR) LEN(40)
   DCL VAR(&KDGL)     TYPE(*CHAR) LEN(10)
   DCL VAR(&DEBET)    TYPE(*DEC)  LEN(17 2)
   DCL VAR(&KREDIT)   TYPE(*DEC)  LEN(17 2)
   DCL VAR(&KODEHASIL) TYPE(*CHAR) LEN(4)

   CHGVAR VAR(&TGLBUKU) VALUE('2026-09-18')
   CHGVAR VAR(&DEBET)   VALUE(1500000.00)
   CHGVAR VAR(&KREDIT)  VALUE(0)

   CALL PGM(APPPGM/POSTINGGL) +
        PARM(&TGLBUKU &NOREF &KDGL &DEBET &KREDIT &KODEHASIL)

   IF COND(&KODEHASIL *NE '0000') THEN(DO)
      SNDPGMMSG MSG('Posting GL gagal: ' *CAT &KODEHASIL) +
                MSGTYPE(*ESCAPE)
   ENDDO
ENDPGM
```

Catatan tentang `*DEC` di CL. Panjang maksimum variabel `*DEC` di CL adalah 15 digit, bukan 17. Bila program COBOL atau RPG mengharapkan `packed(17:2)`, CL tidak bisa melewatkannya langsung. Pola yang dipakai: lewatkan sebagai karakter dan konversi di sisi penerima, atau pecah nilainya. Batas ini sering ditemukan

terlambat, saat nilai transaksi besar mulai muncul.

143.4 Commitment control lintas bahasa

Inilah tempat masalah paling halus muncul. Cakupan commit ditentukan oleh activation group, bukan oleh bahasa.



Gambar 111

Tiga definisi commit yang terpisah berarti tiga transaksi yang bisa berhasil dan gagal secara mandiri. Program RPG yang commit dan program COBOL yang rollback dalam satu rangkaian logis menghasilkan transaksi setengah jadi — tanpa satu pun pesan kesalahan, karena masing-masing program berperilaku benar menurut cakupannya sendiri.

Cara memeriksanya.

```
SELECT JOB_NAME, COMMITMENT_DEFINITION, STATE,
       LOCK_LEVEL, PENDING_CHANGES
  FROM QSYS2.DB_TRANSACTION_INFO
 WHERE JOB_NAME = '123456/EODUSER/EODPOST';
```

Lebih dari satu baris untuk satu job adalah tanda bahaya. Periksa activation group setiap program dalam rantai.

```
SELECT PROGRAM_NAME, PROGRAM_LIBRARY, PROGRAM_TYPE,
       ACTIVATION_GROUP, ACTIVATION_GROUP_ATTRIBUTE
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND PROGRAM_NAME IN ('EODPOST','HITBUNGA','POSTINGGL')
 ORDER BY PROGRAM_NAME;
```

Seluruh program dalam satu rangkaian transaksional harus berada di activation group yang sama. Bila salah satunya OPM, ia tidak bisa — dan itu adalah alasan teknis yang sah untuk memprioritaskan konversinya.

Bab 144. Memilih dan Memelihara

144.1 Untuk kode baru

Kebutuhan	Pilihan yang disarankan
Logika transaksional baru	RPG bebas format dengan embedded SQL
Otomasi operasional	CL, atau Python di PASE
Layanan yang diekspos sebagai API	RPG service program di belakang IWS
Pemrosesan batch besar	RPG dengan SQL blok, atau SQL murni
Pelaporan	SQL, view, dan fungsi tabel
COBOL baru	Hampir tidak pernah dibenarkan

Menulis COBOL baru masuk akal hanya dalam satu situasi: perubahan kecil pada modul COBOL yang sudah ada, di mana menulis ulang dalam bahasa lain justru menambah risiko. Di luar itu, tidak ada alasan teknis memilih COBOL untuk kode baru di IBM i.

144.2 Memelihara kode COBOL warisan

Pendekatan yang saya sarankan, diurutkan berdasarkan rasio manfaat terhadap risiko:

1. **Dokumentasikan sebelum menyentuh.** Petakan file yang dibaca dan ditulis, parameter, dan program yang memanggilnya. `DSPPGMREF` menghasilkan setengah dari peta itu secara otomatis.
2. **Bungkus, jangan tulis ulang.** Bila modul COBOL perlu diakses dari kanal baru, buat service program RPG yang memanggilnya dan mengekspos antarmuka yang bersih. Isinya tidak perlu disentuh.
3. **Konversi OPM ke ILE lebih dulu.** Ini perubahan teknis yang tidak mengubah logika bisnis, dan membuka kemungkinan activation group dan commitment scope yang benar.
4. **Tambahkan uji sebelum mengubah.** Sebuah program yang tidak punya uji dan akan diubah adalah risiko yang tidak perlu. Bangun uji dari perilaku yang ada, bukan dari spesifikasi yang mungkin sudah tidak akurat.
5. **Tulis ulang hanya bila ada alasan bisnis.** Kerapian bukan alasan bisnis.

144.3 Inventarisasi bahasa di sistem Anda

```
-- Komposisi bahasa dan model program
SELECT PROGRAM_TYPE,
       COUNT(*) AS jumlah,
       SUM(CASE WHEN SOURCE_FILE_MEMBER IS NOT NULL
                THEN 1 ELSE 0 END) AS dari_member,
       SUM(CASE WHEN SOURCE_STREAM_FILE_PATH IS NOT NULL
                THEN 1 ELSE 0 END) AS dari_ifs,
       MIN(PROGRAM_CREATION_DATE) AS tertua,
       MAX(PROGRAM_CREATION_DATE) AS terbaru
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
 GROUP BY PROGRAM_TYPE;

-- Program yang sumbernya tidak diketahui sama sekali
SELECT PROGRAM_NAME, PROGRAM_TYPE, PROGRAM_CREATION_DATE,
       TEXT_DESCRIPTION
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND SOURCE_FILE_MEMBER IS NULL
   AND SOURCE_STREAM_FILE_PATH IS NULL
 ORDER BY PROGRAM_CREATION_DATE;
```

Kueri kedua menghasilkan daftar yang layak dibawa ke rapat manajemen risiko. Program produksi yang sumbernya tidak diketahui tidak bisa dikompilasi ulang, tidak bisa diperbaiki, dan akan berhenti berfungsi pada upgrade rilis yang menuntut konversi objek. Setiap baris di daftar itu adalah risiko yang sudah ada, bukan risiko yang mungkin muncul.

144.4 Daftar periksa tinjauan program COBOL

- ☐ Setiap `COMPUTE` yang menghitung uang memakai `ROUNDED`
- ☐ Setiap operasi file diikuti pemeriksaan `FILE STATUS`
- ☐ Subprogram memakai `GOBACK`, bukan `STOP RUN`
- ☐ Tidak ada `GO TO` yang melompat keluar dari paragraf
- ☐ Ukuran `PIC` cocok dengan definisi pemanggil, dihitung dalam byte
- ☐ Activation group sama dengan program lain dalam rantai transaksi
- ☐ Kursor SQL ditutup pada setiap jalur keluar
- ☐ Commit dilakukan berkala pada pemrosesan besar
- ☐ `LINKAGE SECTION` sesuai jumlah dan urutan parameter pemanggil
- ☐ Dikompilasi dengan `DBGVIEW(*SOURCE)` atau lebih

Butir kelima adalah yang paling sering menjadi akar masalah lintas bahasa, dan yang paling mudah diperiksa: hitung ukuran byte kedua definisi, bandingkan, selesai.

🧪 Bagian XLIII — Praktikum

Bagian ini dirancang untuk dikerjakan, bukan dibaca. Setiap praktikum punya tujuan, prasyarat, langkah, hasil yang diharapkan, dan cara memverifikasi bahwa Anda benar-benar berhasil.

Peringatan yang berlaku untuk seluruh bagian ini. Kerjakan di lingkungan pengembangan atau uji, tidak di

produksi. Beberapa praktikum sengaja merusak sesuatu agar Anda belajar memperbaikinya, dan itu tidak boleh dilakukan pada sistem yang melayani nasabah.

🧪 Bab 145. Praktikum Dasar

P-01: Memetakan sistem Anda sendiri

Tujuan. Membangun gambaran menyeluruh tentang sistem yang Anda kelola, dan menemukan hal yang belum Anda ketahui.

Waktu. 2 jam.

Langkah

1. Kumpulkan informasi dasar sistem

```
SELECT * FROM TABLE(QSYS2.SYSTEM_STATUS());

SELECT SYSTEM_VALUE_NAME, CURRENT_CHARACTER_VALUE, CURRENT_NUMERIC_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME IN ('QSECURITY','QMAXSIGN','QAUDCTL',
                             'QPWDLVL','QCRTAUT','QALWOBJRST');
```

2. Petakan library dan ukurannya

```
SELECT OBJLONGSCHEMA AS library, COUNT(*) AS jml_objek,
       SUM(OBJSIZE)/1048576 AS mb
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR','*ALL'))
 GROUP BY OBJLONGSCHEMA ORDER BY mb DESC;
```

3. Daftar subsystem, job queue, dan memory pool yang aktif
4. Daftar interface TCP/IP dan port yang mendengarkan
5. Daftar profil dengan wewenang khusus
6. Daftar tabel produksi yang belum dijurnal

Hasil yang diharapkan. Satu dokumen berisi enam daftar di atas.

Verifikasi. Tunjukkan dokumen itu kepada rekan senior dan minta ia menyebut satu hal yang tidak ada di daftar Anda. Yang ia sebut adalah bagian sistem yang belum Anda ketahui.

Pertanyaan renungan. Berapa banyak dari enam daftar itu yang isinya mengejutkan Anda? Bagian yang mengejutkan adalah bagian yang paling berisiko.

P-02: Membaca job log sampai akar masalah

Tujuan. Melatih kemampuan menelusuri kegagalan dari gejala sampai penyebab.

Waktu. 1 jam.

Persiapan. Buat program CL yang sengaja gagal.

```
PGM
   DCL VAR(&HASIL) TYPE(*DEC) LEN(5 0)
   DCL VAR(&NOL)   TYPE(*DEC) LEN(5 0) VALUE(0)

   SNDPGMMSG MSG('Mulai proses') TOPGMQ(*EXT) MSGTYPE(*INFO)
   CHGVAR VAR(&HASIL) VALUE(100 / &NOL)
   SNDPGMMSG MSG('Selesai') TOPGMQ(*EXT) MSGTYPE(*INFO)
ENDPGM
```

Langkah

1. Kompilasi dan jalankan sebagai batch dengan `LOG(4 00 *SECLVL)`
2. Temukan job-nya dan baca job log lengkap
3. Identifikasi message ID kegagalan
4. Baca teks tingkat kedua pesan itu
5. Temukan nomor statement tempat kegagalan terjadi
6. Baca job log lewat SQL dan bandingkan dengan tampilan layar

```
SELECT ORDINAL_POSITION, MESSAGE_ID, SEVERITY, MESSAGE_TEXT,
       FROM_PROGRAM, FROM_PROCEDURE, FROM_STATEMENT
  FROM TABLE(QSYS2.JOBLOG_INFO('nomor/user/nama'))
 WHERE SEVERITY >= 30 ORDER BY ORDINAL_POSITION;
```

Verifikasi. Anda bisa menyebutkan message ID, nomor baris, dan penyebabnya tanpa melihat kode sumber.

P-03: Menulis utilitas pemantauan pertama

Tujuan. Menghasilkan program CL yang benar-benar dipakai.

Waktu. 4 jam.

Spesifikasi yang harus dipenuhi

Ketentuan	Rincian
Memeriksa pemakaian ASP	Peringatan di 75%, kritis di 85%
Memeriksa job MSGW	Peringatan bila lebih dari nol
Memeriksa job LCKW	Peringatan bila lebih dari tiga
Mencatat hasil ke tabel	Dengan stempel waktu
Mengirim pesan	Hanya bila ada yang melewati ambang
Menangani kesalahan	`MONMSG` pada setiap operasi
Tidak boleh	Menggantung, atau gagal tanpa pesan

Langkah

1. Buat tabel penampung hasil
2. Tulis program CL yang menjalankan pemeriksaan
3. Uji dengan sengaja memicu salah satu kondisi
4. Jadwalkan tiap lima belas menit dengan `ADDJOBSCDE`
5. Biarkan berjalan seminggu, lalu periksa riwayatnya

Verifikasi. Setelah seminggu, tabel riwayat berisi data yang bisa Anda pakai untuk menentukan ambang yang lebih tepat daripada tebakan awal.

Pengembangan lanjutan. Tambahkan pemeriksaan ketujuh: apakah pemantauan itu sendiri masih berjalan.

💾 Bab 146. Praktikum Pemulihan

P-04: Restore satu objek

Tujuan. Melakukan restore untuk pertama kali di lingkungan terkendali, sebelum Anda perlu melakukannya di bawah tekanan.

Waktu. 2 jam.

Langkah

1. Buat library uji dan isi dengan satu tabel berisi seribu baris
2. Catat jumlah baris dan jumlah nilai sebuah kolom numerik
3. Backup library itu ke save file
4. Hapus sepuluh baris dari tabel
5. Catat jumlah baru
6. Restore tabel dari save file ke library sementara
7. Bandingkan isi kedua tabel
8. Restore menimpa tabel asli
9. Verifikasi jumlah baris dan nilai kembali seperti catatan awal

```
CRTLIB LIB(LATIHAN)
CRTSAVF FILE(LATIHAN/BACKUP01)
SAVLIB LIB(LATIHAN) DEV(*SAVF) SAVF(LATIHAN/BACKUP01)
RSTOBJ OBJ(TABELUJI) SAVLIB(LATIHAN) DEV(*SAVF) +
       SAVF(LATIHAN/BACKUP01) RSTLIB(QTEMP)
```

Verifikasi wajib. Jumlah baris dan total nilai kolom numerik harus persis sama dengan catatan langkah dua. Bila tidak, restore Anda belum berhasil meskipun tidak ada pesan kesalahan.

Pelajaran yang ingin dicapai. Restore yang selesai tanpa error bukan bukti keberhasilan. Angka yang cocok adalah buktinya.

P-05: Journaling dan pemulihan ke titik waktu

Tujuan. Memahami hubungan antara save, journal, dan titik pemulihan.

Waktu. 3 jam.

Langkah

1. Buat journal dan receiver di library latihan
2. Aktifkan journaling pada tabel uji
3. Catat waktu, lalu backup tabel
4. Tambahkan seratus baris, catat waktunya
5. Tunggu satu menit, lalu hapus lima puluh baris, catat waktunya
6. Restore tabel dari backup langkah tiga
7. Terapkan perubahan journal sampai waktu antara langkah empat dan lima
8. Periksa jumlah baris

```
CRTJRNRCV JRNRCV(LATIHAN/RCV0001)
CRTJRN JRN(LATIHAN/JRNUJI) JRNRCV(LATIHAN/RCV0001)
STRJRNPF FILE(LATIHAN/TABELUJI) JRN(LATIHAN/JRNUJI) IMAGES(*BOTH)

APYJRNCHG JRN(LATIHAN/JRNUJI) FILE((LATIHAN/TABELUJI)) +
          RCVRNG(*LASTSAVE) TOTIME('[tanggal]' '[jam]')
```

Hasil yang diharapkan. Seratus baris tambahan ada, lima puluh penghapusan belum terjadi. Anda telah memulihkan ke titik waktu di tengah.

Pertanyaan renungan. Apa yang terjadi bila langkah dua dilewati? Coba ulangi seluruh praktikum tanpa mengaktifkan journaling, dan catat apa yang menjadi tidak mungkin.

P-06: Menelusuri perubahan data lewat journal

Tujuan. Menjawab pertanyaan audit klasik: siapa mengubah nilai ini dan kapan.

Waktu. 2 jam.

Langkah

1. Pakai tabel dan journal dari P-05
2. Ubah satu nilai lewat SQL, catat waktunya
3. Ubah nilai yang sama lewat program RPG atau CL, catat waktunya
4. Ekstraksi journal ke outfile
5. Analisis dengan SQL untuk menemukan kedua perubahan
6. Identifikasi program, job, dan pengguna untuk masing-masing

```
DSPJRN JRN(LATIHAN/JRNUJI) FILE((LATIHAN/TABELUJI)) +
       RCVRNG(*CURCHAIN) ENTTYP(*RCD) +
       OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4) OUTFILE(QTEMP/JRNOUT)
```

```
SELECT JOENTT AS tipe, JOTSTP AS waktu, JOUSER AS pengguna,
       JOJOB AS job, JOPGM AS program
  FROM QTEMP.JRNOUT
 WHERE JOENTT IN ('UP','UB','PT','DL')
 ORDER BY JOTSTP;
```

Verifikasi. Anda bisa menunjukkan dua baris yang berbeda nama programnya, sesuai dua cara perubahan di langkah dua dan tiga.

⚡ Bab 147. Praktikum Kinerja

P-07: Membuktikan dampak sebuah index

Tujuan. Mengukur, bukan mengira.

Waktu. 3 jam.

Langkah

1. Buat tabel dengan dua juta baris data sintetis
2. Jalankan kueri dengan kondisi pada kolom tanpa index, catat durasinya

3. Aktifkan pesan debug optimizer dan baca pesan `CPI43xx` yang muncul
4. Buat index yang sesuai
5. Jalankan kueri yang sama, catat durasinya
6. Baca pesan optimizer lagi dan bandingkan
7. Hitung berapa kali lipat perbaikannya

```
CALL QSYS2.OVERRIDE_QAQQINI(1, 'MESSAGES_DEBUG', '*YES');

-- Jalankan kueri, lalu periksa keputusan optimizer
SELECT MESSAGE_ID, MESSAGE_TEXT
  FROM TABLE(QSYS2.JOBLOG_INFO('*'))
 WHERE MESSAGE_ID LIKE 'CPI43%'
 ORDER BY ORDINAL_POSITION DESC FETCH FIRST 10 ROWS ONLY;
```

Hasil yang diharapkan. Sebelum index, pesan `CPI4322` muncul. Sesudahnya, `CPI4327` atau `CPI4326`.

Bagian kedua praktikum. Ukur juga biayanya. Sisipkan seratus ribu baris sebelum dan sesudah index dibuat, dan bandingkan durasinya. Ini memberi Anda kedua sisi dari keputusan yang di banyak tempat hanya dilihat dari satu sisi.

P-08: Menemukan job yang boros

Tujuan. Melatih diagnosis dari data, bukan dari dugaan.

Waktu. 2 jam.

Langkah

1. Tulis program yang sengaja boros: perulangan membaca tabel besar tanpa index
2. Jalankan sebagai batch
3. Temukan job itu lewat `ACTIVE_JOB_INFO` diurutkan berdasarkan CPU
4. Catat CPU, jumlah I/O, dan penyimpanan sementara
5. Ubah program agar memakai index yang sesuai
6. Jalankan ulang dan bandingkan ketiga angka

```
SELECT JOB_NAME, ELAPSED_CPU_PERCENTAGE,
       ELAPSED_TOTAL_DISK_IO_COUNT, TEMPORARY_STORAGE, FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC FETCH FIRST 10 ROWS ONLY;
```

Verifikasi. Anda punya tabel perbandingan tiga angka sebelum dan sesudah, dengan selisih yang bisa dijelaskan.

P-09: Mengamati kontensi kunci

Tujuan. Melihat sendiri bagaimana kunci record menghentikan proses.

Waktu. 1 jam.

Langkah

1. Dari sesi pertama, jalankan program yang melakukan `CHAIN` pada satu record lalu menunggu tanpa `update` maupun `unlock`
2. Dari sesi kedua, jalankan program yang mencoba memperbarui record yang sama

3. Amati status sesi kedua
4. Temukan job penahan dari sesi ketiga
5. Lepaskan kunci dan amati sesi kedua melanjutkan

```
SELECT JOB_NAME, LOCK_STATE, TABLE_SCHEMA, TABLE_NAME, RRN
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'LATIHAN';
```

Hasil yang diharapkan. Sesi kedua berstatus `LCKW`, dan Anda bisa menunjuk job penahannya dengan tepat.

Pelajaran yang ingin dicapai. Kunci yang tertinggal bukan konsep abstrak. Anda baru saja menciptakannya dan memperbaikinya sendiri.

🌙 Bab 148. Praktikum End of Day

P-10: Membangun kerangka EOD sederhana

Tujuan. Menerapkan seluruh prinsip Bab 31 dan 125 dalam skala kecil.

Waktu. 8 jam, bisa dipecah beberapa hari.

Spesifikasi

Bangun proses tiga fase pada data uji:

Fase	Sifat	Yang dilakukan
1. Validasi	Baca saja	Periksa seluruh rekening punya kode produk yang terdaftar
2. Akrual	Tulis, tidak idempoten	Hitung dan posting bunga
3. Rekonsiliasi	Baca dan tulis kontrol	Bandingkan total log dengan total posting

Ketentuan yang wajib dipenuhi

- ☐ Tabel kontrol mencatat status tiap fase
- ☐ Fase 2 menolak berjalan bila fase 1 belum selesai
- ☐ Fase 2 menolak berjalan dua kali pada tanggal buku sama
- ☐ Hasil perhitungan disimpan sebelum diposting
- ☐ Commit tiap seribu record
- ☐ Rekonsiliasi menolak melanjutkan bila ada selisih
- ☐ Setiap fase mencatat jumlah record dan durasinya

Langkah pengujian

1. Jalankan seluruh rangkaian, pastikan berhasil
2. Jalankan fase 2 lagi, pastikan ditolak
3. Hapus rate satu produk, jalankan dari awal, pastikan fase 1 menghentikannya
4. Hentikan fase 2 di tengah dengan `ENDJOB`, lalu jalankan lagi dan amati apakah ia melanjutkan atau mengulang
5. Ubah satu nilai bunga secara manual, jalankan fase 3, pastikan selisih terdeteksi

Verifikasi. Kelima pengujian menghasilkan perilaku yang Anda harapkan. Bila salah satu tidak, temukan

kenapa — di situlah pelajaran sebenarnya.

P-11: Membangun reversal

Tujuan. Menyiapkan kemampuan yang menyelamatkan Anda dari restore.

Waktu. 4 jam.

Prasyarat. P-10 selesai.

Langkah

1. Tulis program reversal yang membalik akrual berdasarkan hasil tersimpan
2. Pastikan ia menandai catatan sebagai dibalik, bukan menghapusnya
3. Pastikan ia menulis entri penyeimbang, bukan menghapus entri asli
4. Jalankan akrual, lalu jalankan reversal
5. Verifikasi saldo kembali ke nilai sebelum akrual
6. Verifikasi jejak kedua operasi masih ada

Verifikasi. Total saldo kembali persis seperti sebelum akrual, dan Anda bisa menunjukkan riwayat lengkap: kapan diposting, kapan dibalik, oleh siapa, dengan alasan apa.

P-12: Simulasi EOD gagal

Tujuan. Melatih pengambilan keputusan sebelum Anda menghadapinya sungguhan.

Waktu. 3 jam.

Skenario yang dijalankan satu per satu

Skenario	Cara membuatnya	Keputusan yang harus Anda ambil
Fase validasi gagal	Hapus rate satu produk	Rerun atau perbaiki dulu
Posting berhenti di tengah	`ENDJOB` saat berjalan	Lanjut dari penanda atau ulang
Akrual gagal, hasil tersimpan	`ENDJOB` setelah sebagian posting	Lanjut, reversal, atau restore
Akrual gagal, hasil tidak tersimpan	Matikan penyimpanan hasil lalu ulangi	Apa pilihan yang tersisa
Rekonsiliasi menemukan selisih	Ubah satu nilai manual	Lanjutkan atau hentikan

Cara mengerjakan. Untuk setiap skenario, tulis keputusan Anda **sebelum** melihat matriks di Bab 125.3. Lalu bandingkan. Perbedaan antara jawaban Anda dan matriks adalah bahan diskusi dengan tim.

Verifikasi. Skenario keempat harus membuat Anda menyimpulkan bahwa restore adalah satu-satunya pilihan. Bila Anda menemukan jalan lain, tuliskan — mungkin Anda menemukan sesuatu yang layak dibahas.

🌐 Bab 149. Praktikum Jaringan dan Keamanan

P-13: Diagnosis jaringan berlapis

Tujuan. Melatih urutan diagnosis dari bawah ke atas.

Waktu. 2 jam.

Langkah

1. Catat konfigurasi interface, rute, dan DNS sistem Anda
2. Hitung secara manual: alamat jaringan, host pertama dan terakhir, broadcast
3. Ping gateway, lalu satu alamat di subnet lain
4. Ping dengan `PKTLEN(1472)` dan bandingkan hasilnya
5. Jalankan `TRACEROUTE` ke satu tujuan di luar subnet
6. Daftar seluruh port yang mendengarkan, dan jelaskan fungsi masing-masing
7. Identifikasi port yang tidak Anda kenali

Verifikasi. Untuk setiap port yang mendengarkan, Anda bisa menyebutkan layanan apa dan apakah ia memang dibutuhkan. Port yang tidak bisa Anda jelaskan adalah temuan.

P-14: Audit wewenang

Tujuan. Menghasilkan laporan audit yang benar-benar bisa dipakai.

Waktu. 4 jam.

Langkah

1. Daftar seluruh profil dengan wewenang khusus
2. Untuk masing-masing, telusuri aktivitas sembilan puluh hari terakhir dari audit journal
3. Tentukan wewenang minimum yang benar-benar dipakai
4. Daftar objek produksi yang `*PUBLIC`-nya bukan `*EXCLUDE`
5. Daftar program yang mengadopsi wewenang pemiliknya
6. Susun rekomendasi dengan prioritas dan perkiraan risiko perubahan

Verifikasi. Laporan Anda memuat, untuk setiap temuan, tiga hal: kondisi saat ini, kondisi yang disarankan, dan risiko dari perubahan itu. Rekomendasi tanpa penilaian risiko tidak akan disetujui.

P-15: Membangun view kepatuhan

Tujuan. Memangkas persiapan audit dari berhari-hari menjadi satu kueri.

Waktu. 4 jam.

Langkah

1. Kumpulkan sepuluh pertanyaan audit yang pernah diajukan ke institusi Anda
2. Untuk masing-masing, tulis kueri yang menjawabnya
3. Gabungkan menjadi satu view dengan kolom pemeriksaan, nilai, dan hasil
4. Jalankan dan tinjau hasilnya
5. Perbaiki temuan yang bisa diperbaiki
6. Jadwalkan view itu dijalankan bulanan

Verifikasi. Seluruh baris berstatus lulus, atau setiap baris yang gagal punya tiket perbaikan yang terbuka.

✍️ Bagian XLIV — Latihan Soal dan Pekerjaan Rumah

Bagian ini melengkapi Bagian XXXI. Soal di sini lebih menuntut, dan sebagian sengaja tidak punya jawaban tunggal.

✍️ Bab 150. Soal Latihan Lanjutan

150.1 Soal analisis kode

S-01 Program berikut menghitung biaya administrasi bulanan. Temukan tiga masalah.

```
**free
ctl-opt dftactgrp(*yes);
dcl-f REKENING usage(*update) keyed;
dcl-s wBiaya packed(7:2);

read REKENING;
dow not %eof(REKENING);
   if STATUS = 'A';
      wBiaya = SALDO * 0.001;
      SALDO = SALDO - wBiaya;
      update REKENINGR;
   endif;
   read REKENING;
enddo;
*inlr = *on;
```

S-02 Program COBOL berikut memposting ke buku besar. Temukan dua masalah.

```
       PROSES-POSTING.
           COMPUTE WS-DEBET = WS-POKOK * WS-RATE / 12
           COMPUTE WS-KREDIT = WS-POKOK * WS-RATE / 12
           WRITE GL-REC
           IF WS-FILE-STATUS NOT = '00'
               DISPLAY 'Gagal tulis GL'
           END-IF
           STOP RUN.
```

S-03 Sebuah program CL menjalankan fase EOD. Temukan empat masalah.

```
PGM
   MONMSG MSGID(CPF0000)
   CALL PGM(PROSESEOD)
   CALL PGM(POSTINGGL)
   CALL PGM(BENTUKLAPOR)
   SNDMSG MSG('EOD selesai') TOUSR(*SYSOPR)
ENDPGM
```

S-04 Kueri berikut dipakai untuk laporan harian pada tabel berisi 600 juta baris. Jelaskan mengapa ia lambat dan bagaimana memperbaikinya tanpa menambah index.

```
SELECT KD_CABANG, SUM(NOMINAL)
  FROM DTALIB.TRXHIST
 WHERE YEAR(TGL_TRX) = 2026 AND MONTH(TGL_TRX) = 9
 GROUP BY KD_CABANG;
```

150.2 Soal perancangan

S-05 Rancang mekanisme pemberian nomor referensi transaksi yang unik, berurutan, dan tahan terhadap

seratus job paralel. Jelaskan pilihan Anda dan apa yang Anda korbankan.

S-06 Sebuah bank ingin RPO lima detik dan RTO sepuluh menit. Jelaskan apa yang secara teknis dibutuhkan, dan apa yang perlu Anda sampaikan ke manajemen tentang biaya dan batasannya.

S-07 Rancang skema pemantauan yang akan mendeteksi ketiga hal ini, dan jelaskan mengapa ketiganya butuh pendekatan berbeda: interface yang gagal, interface yang tidak berjalan sama sekali, dan interface yang berjalan tetapi mengirim data kosong.

S-08 Tabel riwayat transaksi tumbuh dua juta baris per hari dan harus disimpan sepuluh tahun sesuai ketentuan. Rancang strategi penyimpanan, arsip, dan akses yang tetap memungkinkan pencarian data lima tahun lalu dalam waktu wajar.

150.3 Soal pengambilan keputusan

S-09 Pukul 03:15, fase akrual gagal setelah memposting 1,2 juta dari 1,8 juta rekening. Hasil perhitungan tersimpan. Batch window berakhir pukul 05:00. Manajer TI tidak menjawab telepon. Apa yang Anda lakukan, dan mengapa?

S-10 Auditor meminta bukti bahwa tidak ada seorang pun yang dapat mengubah data nasabah tanpa terekam. Anda tahu bahwa satu program batch berjalan dengan wewenang `*ALLOBJ` dan tabel yang disentuhnya tidak diaudit di tingkat objek. Bagaimana Anda menjawab, dan apa yang Anda lakukan setelahnya?

S-11 Vendor aplikasi memberitahu bahwa rilis IBM i yang Anda pakai tidak lagi didukung mulai delapan bulan lagi. Upgrade menuntut penggantian perangkat keras yang belum dianggarkan. Susun argumen untuk manajemen, lengkap dengan pilihan dan konsekuensinya.

S-12 Tim pengembang meminta akses `*ALLOBJ` di produksi untuk mempercepat penyelesaian insiden. Argumen mereka: setiap kali ada masalah, mereka harus menunggu System Engineer. Tulis tanggapan Anda yang menangani kebutuhan nyata mereka tanpa memberikan apa yang mereka minta.

✍️ Bab 151. Pekerjaan Rumah

Berbeda dari soal, pekerjaan rumah dikerjakan pada sistem Anda sendiri dan menghasilkan sesuatu yang dipakai.

PR-01: Inventarisasi risiko tersembunyi

Tenggat. Satu minggu.

Jalankan tujuh kueri berikut pada sistem produksi Anda, dan susun hasilnya menjadi satu laporan dengan penilaian risiko per temuan.

1. Tabel produksi yang belum dijurnal
2. Kolom teks dengan CCSID 65535
3. Program yang sumbernya tidak diketahui
4. Profil `*ALLOBJ` yang tidak dipakai lebih dari sembilan puluh hari

5. Library yang ada tetapi tidak muncul di riwayat backup tujuh hari terakhir
6. Constraint dalam status check pending
7. Sertifikat yang kedaluwarsa dalam seratus dua puluh hari

Keluaran. Laporan dengan format: temuan, jumlah, dampak bila terjadi, usaha perbaikan, prioritas.

Kenapa ini pekerjaan rumah pertama. Ketujuh kueri itu hampir selalu menemukan sesuatu, dan setiap temuan adalah insiden yang belum terjadi. Menemukannya sekarang jauh lebih murah daripada nanti.

PR-02: Tiga runbook dari insiden nyata

Tenggat. Dua minggu.

1. Ambil tiga insiden yang paling sering terjadi di sistem Anda selama setahun terakhir
2. Tulis runbook untuk masing-masing, mengikuti templat Lampiran C
3. Minta rekan yang belum pernah menangani insiden itu menjalankannya di lingkungan uji
4. Catat setiap kali ia bertanya, dan perbaiki runbook di titik itu
5. Ulangi sampai ia bisa menjalankannya tanpa bertanya

Keluaran. Tiga runbook teruji, plus catatan berapa kali revisi dibutuhkan.

PR-03: Baseline dan pembanding

Tenggat. Satu bulan.

1. Bangun program pengambil baseline konfigurasi bulanan
2. Jalankan bulan ini dan simpan hasilnya
3. Jalankan lagi bulan depan
4. Bandingkan, dan telusuri setiap perbedaan
5. Untuk setiap perbedaan, temukan tiket perubahan yang menjelaskannya

Keluaran. Laporan perbandingan, dengan daftar perubahan yang tidak punya tiket.

Yang biasanya ditemukan. Hampir selalu ada perubahan tanpa tiket. Itu bukan tuduhan kepada siapa pun; itu ukuran kualitas proses manajemen perubahan Anda.

PR-04: Studi kinerja satu fase EOD

Tenggat. Tiga minggu.

1. Pilih satu fase EOD yang durasinya paling bervariasi
2. Kumpulkan durasinya selama tiga bulan dari tabel kontrol
3. Cari korelasi dengan volume transaksi, hari dalam bulan, dan beban sistem
4. Bentuk satu hipotesis tentang penyebab variasinya
5. Ukur untuk mengujinya
6. Tulis laporan mengikuti templat Bab 79.3

Keluaran. Laporan studi kinerja lengkap, termasuk hipotesis yang ditolak dan alasannya.

PR-05: Uji pemulihan penuh

Tenggat. Satu bulan, dengan persetujuan.

1. Ajukan permohonan jendela untuk restore satu library produksi ke lingkungan uji
2. Siapkan angka kontrol sebelum
3. Lakukan restore lengkap dengan apply journal
4. Verifikasi dengan angka kontrol
5. Catat durasi aktual setiap langkah
6. Bandingkan dengan RTO yang dijanjikan institusi Anda
7. Tulis berita acara

Keluaran. Berita acara dengan RTO aktual, dan bila berbeda dari yang dijanjikan, analisis penyebabnya.

Kenapa ini yang paling penting. Sebagian besar institusi tidak tahu RTO aktualnya. Mereka tahu RTO yang tertulis di dokumen. Perbedaan antara keduanya baru ditemukan saat bencana, kecuali seseorang mengukurnya lebih dulu.

📦 Bagian XLV — Proyek Studi Mandiri

Proyek di bagian ini berdurasi satu sampai tiga bulan dan menghasilkan sesuatu yang dipakai institusi Anda. Masing-masing dirancang agar bisa dikerjakan sambil menjalankan tugas harian.

📦 Bab 152. Proyek Tingkat Menengah

PM-01: Sistem pemantauan terpadu

Durasi. Dua bulan.

Hasil akhir. Sistem pemantauan yang berjalan sendiri, mengikuti Bagian XXXII.

Tahap	Pekan	Keluaran
Rancang skema	1	Tabel metrik, ambang, peringatan
Bangun pengumpul	2–3	Program CL terjadwal
Kumpulkan baseline	4–5	Data dua pekan
Tetapkan ambang	6	Ambang berbasis data, bukan tebakan
Bangun evaluasi dan peringatan	7	Termasuk penekanan berulang
Bangun dasbor dan tren	8	View siap dibaca alat luar

Kriteria keberhasilan. Setelah dua bulan, sistem mendeteksi minimal satu kondisi yang sebelumnya baru diketahui setelah ada keluhan.

PM-02: Migrasi kode sumber ke Git

Durasi. Tiga bulan.

Hasil akhir. Seluruh kode sumber aplikasi di kendali versi, dengan pipeline build.

Tahap	Pekan	Keluaran
Inventarisasi sumber	1–2	Daftar member, dan yang sumbernya hilang
Ekspor ke IFS	3–4	Berkas UTF-8 dengan akhir baris LF
Bangun struktur repositori	5	Struktur folder dan konvensi
Impor ke Git	6	Commit awal
Bangun skrip build	7–9	Build ke library uji berhasil
Bangun pipeline	10–11	Build otomatis dari commit
Dokumentasi dan pelatihan	12	Tim bisa memakainya

Kriteria keberhasilan. Perubahan kode bisa ditelusuri ke orang, waktu, dan alasannya. Pertanyaan "siapa mengubah baris ini" punya jawaban.

PM-03: Katalog runbook institusi

Durasi. Dua bulan.

Hasil akhir. Dua puluh runbook teruji, terindeks, dan tersedia di luar sistem.

Tahap	Pekan	Keluaran
Kumpulkan riwayat insiden	1	Daftar insiden setahun, diurutkan frekuensi
Prioritaskan	2	Dua puluh teratas
Tulis batch pertama	3–4	Sepuluh runbook
Uji dengan rekan	5	Revisi berdasarkan pertanyaan
Tulis batch kedua	6–7	Sepuluh runbook lagi
Uji dan indeks	8	Indeks cetak, salinan luar sistem

Kriteria keberhasilan. Seorang anggota tim yang baru bisa menangani insiden dari daftar itu tanpa membangunkan siapa pun.

📦 Bab 153. Proyek Tingkat Lanjut

PL-01: Program kesiapan pemulihan bencana

Durasi. Tiga bulan.

Hasil akhir. Kesiapan DR yang terbukti, bukan yang diasumsikan.

Tahap	Pekan	Keluaran
Inventarisasi objek luar iASP	1–2	Daftar dan cara sinkronisasinya
Bandingkan primer dan DR	3–4	Laporan selisih konfigurasi
Perbaiki selisih	5–6	Kedua sisi identik
Susun prosedur switchover	7–8	Dokumen langkah demi langkah
Susun daftar uji kanal	9	Setiap kanal dengan penanggung jawab
Latihan tingkat 2	10	Simulasi lisan, temuan tercatat
Latihan tingkat 3	11–12	Switchover nyata, RTO terukur

Kriteria keberhasilan. RTO aktual terukur dan terdokumentasi. Bila berbeda dari yang dijanjikan, selisihnya

diketahui manajemen.

PL-02: Modernisasi satu modul menjadi API

Durasi. Tiga bulan.

Hasil akhir. Satu fungsi bisnis yang tadinya hanya bisa diakses dari layar 5250, kini tersedia sebagai REST API.

Tahap	Pekan	Keluaran
Pilih modul dan pahami	1–2	Dokumen alur data dan aturan bisnis
Bangun service program	3–5	Logika terpisah dari antarmuka
Tulis uji otomatis	6	Uji untuk setiap aturan bisnis
Bangun lapisan API	7–8	Endpoint dengan kode hasil konsisten
Tambahkan idempotensi dan log	9–10	Termasuk korelasi ID
Uji beban dan keamanan	11	Batas dan autentikasi terverifikasi
Dokumentasi	12	Kontrak API untuk konsumen

Kriteria keberhasilan. Layar 5250 lama dan API baru memakai logika yang sama persis, sehingga tidak mungkin menghasilkan angka berbeda.

PL-03: Program pengurangan ketergantungan pada satu orang

Durasi. Tiga bulan.

Hasil akhir. Setiap area kritikal punya minimal dua orang yang bisa menanganinya.

Tahap	Pekan	Keluaran
Petakan ketergantungan	1	Matriks area dan orang, dari data audit
Identifikasi titik tunggal	2	Daftar area dengan satu orang
Susun rencana pasangan	3	Pasangan senior dan pendamping
Dokumentasikan area kritis	4–7	Runbook dan dokumen arsitektur
Rotasi terbimbing	8–10	Pendamping memegang kemudi
Uji tanpa senior	11–12	Pendamping menangani sendiri

Kriteria keberhasilan. Matriks ketergantungan yang diambil ulang di akhir program menunjukkan tidak ada area dengan satu orang.

Catatan tentang proyek ini. Ini proyek yang paling sering ditunda karena tidak menghasilkan sesuatu yang terlihat. Tetapi di lingkungan dengan sedikit praktisi IBM i, ia adalah proyek dengan pengurangan risiko terbesar per jam yang diinvestasikan — dan satu-satunya yang manfaatnya terasa justru ketika seseorang tidak ada.

📖 Bagian XLVI — Glosarium

Disusun menurut abjad. Istilah yang punya padanan Indonesia diberi keduanya, karena dokumentasi di lapangan mencampur keduanya.

A

Activation group — Wadah tempat sumber daya program ILE hidup: file terbuka, override, dan definisi commitment control. Dua program di activation group berbeda melakukan commit secara terpisah, dan ini penyebab transaksi setengah jadi yang paling halus. Lihat Bab 50.1.

ACTLVL (activity level) — Jumlah maksimum thread yang boleh aktif bersamaan dalam sebuah memory pool. Terlalu rendah membuat job mengantre meskipun memori cukup; gejalanya `ACTIVE_TO_INELIGIBLE` tidak nol.

Adopted authority — Mekanisme di mana program berjalan dengan wewenang pemiliknya, bukan pemanggilnya. Sah dan berguna, tetapi berbahaya bila pemiliknya punya `*ALLOBJ`.

AF (authority failure) — Jenis entri audit journal yang mencatat percobaan akses tanpa wewenang.

ASP (Auxiliary Storage Pool) — Kelompok unit disk yang diperlakukan sebagai satu kolam penyimpanan. ASP 1 adalah system ASP dan wajib ada.

Authorization list — Objek yang memuat daftar pengguna dan wewenangnya, lalu dikaitkan ke banyak objek sekaligus. Cara paling rapi mengelola wewenang pada banyak objek serupa.

B

Batch window — Rentang waktu yang tersedia untuk menjalankan proses batch tanpa mengganggu layanan. Di perbankan biasanya antara cut-off dan jam buka layanan.

Binder language — Sumber yang mendefinisikan prosedur mana yang diekspor sebuah service program, beserta signature-nya. Perubahan yang salah di sini memutus seluruh program pemakainya.

BRMS (Backup Recovery and Media Services) — Produk pengelola backup yang mencatat objek apa tersimpan di media mana, sehingga pemulihan tidak bergantung pada ingatan seseorang.

C

CCSID (Coded Character Set Identifier) — Penanda set karakter sebuah objek atau kolom. Nilai 37 adalah EBCDIC Amerika, 1208 adalah UTF-8, dan 65535 berarti tanpa konversi. Yang terakhir adalah jebakan; lihat Bab 86.2.

Checkpoint — Titik konsisten dalam proses, dipakai sebagai tempat kembali bila terjadi kegagalan. Dalam konteks EOD, checkpoint sebelum fase akrual adalah yang paling kritikal.

CL (Control Language) — Bahasa perintah dan skrip IBM i. Bukan bahasa aplikasi, melainkan bahasa otomasi operasi.

Commitment control — Mekanisme yang memastikan sekelompok perubahan basis data berhasil atau gagal bersama-sama. Prasyaratnya tabel harus dijurnal.

CPW (Commercial Processing Workload) — Satuan kapasitas transaksional lama, masih dipakai untuk perbandingan relatif antar model.

Cut-off — Waktu penutupan penerimaan transaksi untuk tanggal buku berjalan. Transaksi setelahnya masuk tanggal buku berikutnya.

D

Data area — Objek kecil penyimpan satu nilai yang dibaca banyak job. Lazim dipakai menyimpan tanggal buku atau penanda status.

Data queue — Antrean pesan antar job di dalam satu sistem. Cepat dan ringan, tetapi isinya hilang saat sistem berhenti tidak normal kecuali dibuat dengan `FORCE(*YES)`.

DCM (Digital Certificate Manager) — Pengelola sertifikat digital sistem. Sertifikat kedaluwarsa yang tidak terpantau adalah penyebab pemadaman layanan yang sepenuhnya bisa dicegah.

DDM (Distributed Data Management) — Mekanisme mengakses objek di sistem IBM i lain sebagai objek lokal.

DDS (Data Description Specification) — Bahasa definisi file gaya lama, dipakai untuk physical file, logical file, display file, dan printer file. Untuk tabel, penggantinya adalah SQL DDL.

Deadlock — Kondisi dua job saling menunggu kunci yang dipegang yang lain. Db2 for i mendeteksinya dan mengakhiri salah satu dengan `SQLCODE -911`.

Duplex mismatch — Ketidaksesuaian setelan duplex antara dua ujung tautan jaringan. Gejalanya interface naik, ping berhasil, tetapi throughput anjlok — salah satu masalah jaringan yang paling membingungkan.

E

EBCDIC — Set karakter warisan yang dipakai IBM i secara native, berbeda dari ASCII. Sumber hampir seluruh masalah konversi karakter.

EOD (End of Day) — Rangkaian proses batch penutup hari: posting, akrual, klasifikasi, penyeimbangan buku besar, pelaporan, dan pemajuan tanggal buku.

EOM / EOY — End of Month dan End of Year. EOY paling berisiko karena hanya berjalan sekali setahun, artinya tidak diuji di produksi selama dua belas bulan.

EVI (Encoded Vector Index) — Jenis index yang efisien untuk kolom dengan sedikit nilai berbeda. Dengan klausa `INCLUDE`, ia bisa menyimpan agregat yang sudah dihitung.

Exit program — Program yang dipanggil sistem pada titik masuk tertentu, misalnya koneksi ODBC atau perintah FTP, untuk menyaring akses. Mengisi celah yang tidak bisa ditutup wewenang objek.

F

Failover — Perpindahan peran ke sistem cadangan karena sistem utama hilang. Berbeda dari switchover yang terencana.

FIELDPROC — Mekanisme enkripsi transparan pada tingkat kolom. Program aplikasi tidak perlu diubah,

tetapi kolomnya tidak bisa dipakai untuk pencarian rentang.

FILE STATUS — Kode hasil operasi file dalam COBOL. Nilai `00` berarti berhasil; pemeriksaannya wajib setelah setiap operasi.

G

GOBACK — Pernyataan COBOL untuk kembali ke pemanggil. Berbeda dari `STOP RUN` yang mengakhiri seluruh unit eksekusi dan bisa menjatuhkan pemanggilnya.

Group PTF — Kumpulan perbaikan yang dikelompokkan menurut area, misalnya Db2 atau HTTP. Levelnya dipantau lewat `WRKPTFGRP`.

H

HMC (Hardware Management Console) — Konsol pengelola partisi dan perangkat keras. Mengelola partisi, bukan sistem operasi di dalamnya.

I

iASP (Independent ASP) — ASP yang bisa dipindahkan antar sistem. Prasyarat switchover PowerHA. Objek di luar iASP tidak ikut berpindah; lihat Bab 55.3.

Idempoten — Sifat sebuah operasi yang bila dijalankan berulang menghasilkan keadaan akhir yang sama. Fase akrual bunga secara alami tidak idempoten, dan itulah sumber risiko terbesar dalam EOD.

Idempotency key — Pengenal unik per permintaan yang dipakai mencegah pemrosesan ganda saat klien mengirim ulang. Harus ditulis dalam transaksi yang sama dengan operasinya.

IFS (Integrated File System) — Sistem berkas bergaya Unix di IBM i, hidup berdampingan dengan penyimpanan berbasis library.

ILE (Integrated Language Environment) — Model program modern IBM i yang mendukung modul, service program, dan activation group. Penggantinya OPM.

IPL (Initial Program Load) — Proses penyalaan sistem. Operasi paling berisiko yang dilakukan secara rutin.

IWS (Integrated Web Services) — Fasilitas yang membungkus program RPG atau CL menjadi layanan REST atau SOAP tanpa menulis ulang logikanya.

J

Job description — Objek yang memuat atribut bawaan sebuah job: job queue, prioritas, library list, tingkat pencatatan.

Journal — Objek yang mencatat setiap perubahan record sebelum dan sesudah. Fondasi commitment control, replikasi, dan pemulihan ke titik waktu.

Journal receiver — Objek tempat entri journal sesungguhnya disimpan. Tumbuh terus dan harus dikelola

siklus hidupnya.

K

Kolektibilitas — Klasifikasi kualitas kredit menurut ketentuan regulator, umumnya lima tingkat berdasarkan hari tunggakan. Perpindahan massal dalam satu malam hampir selalu berarti kesalahan perhitungan.

L

Level check — Kondisi di mana format record sebuah file berubah tetapi program pemakainya belum dikompilasi ulang. Muncul sebagai `CPF4131`.

Library list — Urutan pencarian objek untuk sebuah job. Kesalahan urutan adalah penyebab nomor satu insiden yang tampak sebagai objek hilang.

Logical file — Objek DDS yang menyediakan jalur akses atau pandangan atas physical file. Padanan modernnya index atau view SQL.

LPAR (Logical Partition) — Pembagian satu server fisik menjadi beberapa sistem logis yang terisolasi.

M

Memory pool — Kolam memori tempat job berjalan. Pemisahan pool antara beban interaktif dan batch adalah salah satu penyetelan kinerja paling berdampak.

MSGW — Status job yang menunggu jawaban pesan dari manusia. Satu job MSGW yang tidak terdeteksi bisa menahan seluruh rantai EOD sampai pagi.

MTU (Maximum Transmission Unit) — Ukuran paket terbesar yang dapat dilewatkan sebuah jalur. Ketidaksesuaian MTU menghasilkan gejala khas: koneksi terbentuk, transfer kecil berhasil, transfer besar menggantung.

O

OPM (Original Program Model) — Model program lama sebelum ILE. Tidak mendukung activation group bernama dan tidak bisa di-bind dengan service program ILE.

Optimizer — Komponen Db2 for i yang memutuskan bagaimana sebuah kueri dijalankan. Keputusannya terbaca lewat pesan `CPI43xx`.

Override — Mekanisme mengubah atribut file atau perangkat untuk sebuah job tanpa mengubah objeknya. Sangat berguna, dan sering menjadi penyebab keluaran muncul di tempat tak terduga.

P

PASE (Portable Application Solutions Environment) — Lingkungan runtime AIX di dalam IBM i, tempat Node.js, Python, dan alat open source lain berjalan. Job-nya terlihat sebagai `QP0ZSPWP`.

Physical file — Objek DDS yang menyimpan data. Padanan modernnya tabel SQL, yang punya kemampuan tambahan seperti constraint, trigger, dan RCAC.

Plan cache — Penyimpanan rencana eksekusi kueri yang dipakai ulang. Sumber informasi terbaik untuk menemukan kueri mahal.

PowerHA SystemMirror for i — Solusi ketersediaan tinggi berbasis cluster dan iASP.

Prestart job — Job yang disiapkan lebih dulu agar koneksi baru tidak menanggung biaya pembuatan job. Penting untuk beban ODBC dan layanan web.

PTF (Program Temporary Fix) — Perbaikan yang diterbitkan IBM untuk sistem operasi dan produk berlisensi.

Q

QAQQINI — Tabel opsi kueri yang memungkinkan penyetelan perilaku optimizer, termasuk batas waktu kueri dan tingkat paralelisme.

QAUDJRN — Audit journal sistem. Sumber kebenaran untuk hampir seluruh pertanyaan audit.

QSECURITY — System value yang menentukan tingkat keamanan sistem. Nilai 40 adalah minimum yang dapat diterima untuk perbankan.

QTEMP — Library sementara yang unik untuk setiap job dan dihapus saat job berakhir. Tempat yang benar untuk objek kerja.

R

RCAC (Row and Column Access Control) — Mekanisme pembatasan akses di tingkat basis data yang berlaku untuk semua jalur akses, termasuk untuk profil dengan `*ALLOBJ`. Jawaban untuk temuan audit klasik tentang administrator yang bisa melihat seluruh data nasabah.

Reversal — Pembalikan sebuah posting dengan entri berlawanan, bukan dengan penghapusan. Prinsip akuntansi yang menentukan bentuk kode: entri salah dibalik, jejak keduanya tetap ada.

RPO (Recovery Point Objective) — Batas kehilangan data yang dapat diterima, dinyatakan dalam waktu. RPO nol menuntut replikasi sinkron dan membawa batasan jarak fisik.

RTO (Recovery Time Objective) — Batas waktu pemulihan layanan yang dapat diterima. Sebagian besar institusi tahu RTO yang tertulis, bukan RTO aktualnya.

RUNPTY — Prioritas job saat berjalan. Angka lebih kecil berarti prioritas lebih tinggi.

S

Save While Active — Kemampuan menyimpan objek sementara aplikasi tetap berjalan, dengan mengambil titik konsisten lebih dulu. Kunci agar backup tidak memakan batch window.

Seize — Mekanisme penguncian internal sistem pada tingkat di bawah kunci record. Kontensi seize terlihat

sebagai job yang lambat tanpa penyebab yang jelas.

Service program — Objek ILE yang memuat prosedur untuk dipakai bersama oleh banyak program. Punya signature yang harus dikelola lewat binder language.

SIBS (Silverlake Axis Integrated Banking Solution) — Sistem core banking dari Silverlake Axis yang berjalan di IBM i, dipakai di banyak bank ASEAN termasuk Indonesia. Tidak berhubungan dengan Jack Henry SilverLake System.

Single-level storage — Arsitektur di mana memori utama dan disk membentuk satu ruang alamat tunggal. Program tidak pernah tahu data berada di mana.

SLIC (System Licensed Internal Code) — Lapisan kode di bawah sistem operasi yang menerjemahkan instruksi TIMI ke instruksi prosesor.

Split brain — Kondisi kedua node cluster mengira dirinya primer dan keduanya menerima transaksi. Pemulihannya menuntut rekonsiliasi manual yang bisa memakan berhari-hari.

Subfile — Mekanisme daftar bergulir pada layar 5250. Pola yang mendominasi aplikasi perbankan, dengan batas 9999 record.

Subsystem — Wadah tempat job berjalan, dengan konfigurasi pool memori, job queue, dan routing sendiri.

Switchover — Perpindahan peran ke sistem cadangan secara terencana, dengan kedua sisi sehat. Berbeda dari failover.

T

Temporal table — Tabel yang menyimpan riwayat perubahannya sendiri secara otomatis. Menjawab pertanyaan audit tentang nilai sebuah parameter pada tanggal tertentu tanpa kode tambahan.

Temporary storage — Penyimpanan sementara yang dipakai job, misalnya untuk file hasil kueri. Lonjakan pemakaian ASP yang cepat hampir selalu berasal dari sini, bukan dari data permanen.

TIMI (Technology Independent Machine Interface) — Lapisan instruksi virtual yang memisahkan program dari perangkat keras. Alasan program tahun 1990 masih berjalan di Power10 tanpa kompilasi ulang.

Technology Refresh (TR) — Paket fitur baru yang diterbitkan tanpa mengganti nomor rilis.

Trigger — Program atau logika SQL yang dijalankan otomatis pada perubahan tabel. Berguna, tetapi mahal pada pemrosesan massal karena berjalan sekali per record.

V

VIOS (Virtual I/O Server) — Partisi khusus yang menyediakan akses I/O tervirtualisasi ke partisi lain. Dipasang berpasangan agar tidak ada titik kegagalan tunggal.

Virtual IP — Alamat IP yang tidak terikat pada satu kartu fisik, sehingga bisa berpindah saat failover. Cara menghindari masalah daftar putih alamat IP di sisi mitra.

W

Wait accounting — Kerangka analisis kinerja yang bertanya ke mana waktu sebuah job pergi, bukan berapa CPU yang dipakai. Kerangka berpikir paling berguna untuk kinerja di IBM i.

Watch — Mekanisme sistem yang memantau kemunculan pesan tertentu dan memanggil program penangan. Dasar pemantauan otomatis tanpa perangkat tambahan.

Work management — Kumpulan mekanisme yang menentukan bagaimana job dijalankan: subsystem, job queue, routing entry, class, dan memory pool. Prasyarat untuk memahami hampir seluruh pembahasan operasional.

Bagian XLVII — Anatomi Insiden

Bagian XXIV dan XXXVI menyajikan studi kasus dalam bentuk ringkas: gejala, diagnosis, akar masalah, pelajaran. Bentuk itu efisien untuk dibaca berulang, tetapi ia menghilangkan sesuatu yang penting — bagaimana rasanya berada di tengahnya.

Bagian ini menyajikan empat insiden dalam bentuk narasi panjang, jam demi jam, termasuk keraguan, dugaan yang salah, keputusan yang harus diambil tanpa informasi lengkap, dan tekanan dari orang-orang yang menunggu jawaban. Tujuannya bukan menambah pengetahuan teknis; seluruh teknik yang dipakai sudah dijelaskan di bab-bab sebelumnya. Tujuannya melatih penilaian.

Penilaian tidak bisa diajarkan lewat daftar langkah. Ia terbentuk dari pengalaman, dan pengalaman orang lain adalah pengganti terdekat yang tersedia. Bacalah keempat narasi ini bukan untuk menghafal solusinya, melainkan untuk bertanya pada diri sendiri di setiap titik keputusan: apa yang akan saya lakukan di posisi itu, dengan informasi yang tersedia saat itu?

Seluruh narasi disusun dari pola yang berulang di lapangan dan telah diteranonimkan sepenuhnya. Nama, angka, dan institusi adalah rekaan.

Bab 154. Narasi Pertama: Malam Ketika Akrual Hampir Berjalan Dua Kali

154.1 Pukul 22:40 — Sesuatu yang tampak normal

Fase posting transaksi selesai pukul 22:38, dua belas menit lebih cepat dari biasanya. Operator jaga, seorang staf bernama Dian yang sudah delapan bulan memegang giliran malam, mencatatnya di lembar pemantauan dan memberi tanda centang. Lebih cepat dari biasanya bukan hal yang memicu kecurigaan siapa pun. Malam itu volume transaksi memang sedikit di bawah rata-rata karena hari sebelumnya libur nasional.

Di sinilah pelajaran pertama dari narasi ini terletak, dan ia bertentangan dengan naluri. **Penyimpangan ke arah yang menguntungkan tetap penyimpangan.** Proses yang selesai lebih cepat tanpa penjelasan yang jelas layak mendapat perhatian yang sama dengan proses yang melambat. Alasannya sederhana: keduanya berarti sistem berperilaku berbeda dari yang Anda pahami, dan pemahaman yang keliru tentang sistem adalah akar dari hampir semua insiden besar.

Dian melanjutkan ke fase akrual pukul 22:45. Tabel kontrol menunjukkan statusnya berubah menjadi berjalan. Ia menuangkan kopi dan membuka laporan yang harus ia periksa setiap malam.

154.2 Pukul 23:52 — Kegagalan pertama

Satu jam tujuh menit kemudian, layar pemantauan menunjukkan job akrual berstatus gagal. Pesan di antrean operator menyebut `RNQ0100` — pembagian dengan nol.

Dian melakukan apa yang tertulis di runbook: mencatat waktunya, memeriksa tabel kontrol untuk melihat sejauh mana proses sempat berjalan, dan menghubungi Application Support yang bertugas. Sampai di sini seluruhnya benar.

Tabel kontrol menunjukkan angka yang membingungkan. Kolom jumlah record menunjukkan 1.247.883 dari perkiraan 1.810.000 rekening aktif. Artinya proses berhenti di sekitar dua pertiga jalan.

Application Support yang bertugas, seorang staf bernama Arif, terhubung dari rumah dalam sebelas menit. Ia meminta Dian membacakan pesan kesalahan lengkap dan nomor baris program. Keduanya butuh waktu karena runbook tidak mencantumkan cara membaca teks tingkat kedua pesan, dan Dian belum pernah melakukannya. Sebelas menit lagi berlalu.

Catatan atas titik ini. Sebelas menit yang hilang bukan kesalahan Dian. Itu kesalahan dokumentasi. Runbook yang menyebutkan "baca pesan kesalahan" tanpa menjelaskan caranya mengasumsikan pengetahuan yang belum tentu ada pada orang yang menjalankannya pukul tengah malam. Setiap kali Anda menulis runbook, tanyakan: apakah seseorang yang belum pernah melakukan ini bisa mengikutinya tanpa bertanya?

154.3 Pukul 00:20 — Dugaan pertama, yang salah

Arif menduga masalahnya pada satu rekening dengan data rusak. Dugaan itu masuk akal: `RNQ0100` pada perhitungan bunga biasanya berarti ada pembagi bernilai nol, dan pembagi dalam rumus bunga adalah jumlah hari dalam setahun — nilai yang diambil dari tabel parameter.

Ia menjalankan kueri untuk mencari baris parameter yang kosong. Tidak ada. Seluruh parameter terisi dengan benar.

Dugaan kedua: mungkin ada rekening dengan kode produk yang tidak terdaftar, sehingga fungsi pengambil rate mengembalikan struktur kosong dan salah satu fieldnya bernilai nol. Ia menjalankan kueri validasi produk. Juga bersih.

Pukul 00:45 Arif masih belum menemukan penyebabnya, dan mulai merasakan tekanan yang khas: batch window berakhir pukul 05:00, masih ada empat fase setelah akrual, dan ia belum bergerak dari titik awal.

Catatan atas titik ini. Dua dugaan Arif keduanya masuk akal dan keduanya salah. Ini kondisi normal dalam diagnosis, bukan tanda ketidakcakapan. Yang menjadi masalah adalah ia menghabiskan dua puluh lima menit menguji dugaan tanpa lebih dulu mengumpulkan data yang sudah tersedia. Job log lengkap ada di sistem sejak pukul 23:52, memuat nomor baris persis tempat kegagalan terjadi, dan belum dibaca sampai pukul 00:45.

Pola ini sangat umum. Ketika seseorang punya hipotesis yang terasa meyakinkan, ia cenderung mengujinya lebih dulu daripada mengumpulkan pengamatan. Struktur algoritma diagnosis di Bab 139 dirancang khusus untuk menahan kecenderungan ini: langkah pertama adalah pengamatan, bukan hipotesis.

154.4 Pukul 00:52 — Penyebab ditemukan

Arif akhirnya membaca job log. Nomor baris menunjuk ke sebuah perhitungan yang tidak ia duga: bukan pembagian dengan jumlah hari setahun, melainkan pembagian dengan jumlah hari dalam bulan berjalan — nilai yang dihitung program, bukan diambil dari parameter.

Program menghitung jumlah hari bulan itu dari selisih dua tanggal yang diambil dari tabel kalender hari kerja.

Dan di sana masalahnya. Karena hari sebelumnya libur nasional yang baru ditetapkan pemerintah tiga pekan sebelumnya, seseorang telah memperbarui tabel kalender. Pembaruan itu benar untuk keperluannya, tetapi menghapus satu baris penanda yang dipakai program akrual untuk menentukan batas bulan. Untuk rekening dengan produk tertentu, yang jatuh pada cabang kode yang memakai penanda itu, selisih tanggal menjadi nol.

Catatan atas titik ini. Penyebabnya bukan kegagalan teknis, melainkan perubahan data yang tidak dikomunikasikan lintas fungsi. Orang yang memperbarui kalender tidak tahu bahwa program akrual bergantung pada baris yang ia hapus, dan tidak ada mekanisme yang akan memberitahunya. Ini pola nomor satu dalam distribusi penyebab di Bab 124.1, dan ia tidak bisa diperbaiki dengan teknologi.

154.5 Pukul 01:05 — Keputusan yang sesungguhnya sulit

Setelah penyebab diketahui, perbaikannya sederhana: kembalikan baris kalender yang terhapus. Arif melakukannya dalam tiga menit.

Tetapi pertanyaan berikutnya jauh lebih sulit, dan inilah inti dari narasi ini. **Apa yang dilakukan terhadap 1.247.883 rekening yang sudah terposting?**

Arif mengajukan pilihan yang terasa paling praktis: jalankan ulang fase akrual. Program akan memproses seluruh rekening, termasuk yang sudah terposting, dan menurut pemahamannya hasilnya akan sama karena rumusnya sama.

Dian, yang sudah delapan bulan menjalankan EOD dan pernah membaca dokumen prosedur, mengingat sesuatu. Ia bertanya apakah fase akrual boleh dijalankan ulang.

Jawaban Arif: seharusnya bisa, karena rumusnya deterministik.

Jawaban itu salah, dan kesalahannya sangat mudah dilakukan. Rumusnya memang deterministik: bunga yang dihitung untuk sebuah rekening dengan saldo tertentu akan selalu bernilai sama. Tetapi operasinya bukan penetapan, melainkan penambahan. Rekening yang saldonya sudah bertambah akan dihitung ulang berdasarkan saldo baru, lalu ditambah lagi.

Dian tidak yakin cukup untuk membantah dengan tegas, tetapi cukup tidak yakin untuk menahan. Ia mengatakan bahwa ia perlu mengonfirmasi ke manajer sebelum menjalankan ulang fase akrual, karena itu

tertulis dalam prosedur.

Catatan atas titik ini. Aturan yang menyelamatkan malam itu bukan keahlian teknis, melainkan aturan prosedural yang ditulis jauh sebelumnya oleh seseorang yang mungkin sudah tidak bekerja di sana lagi: keputusan rerun fase akrual menuntut dua orang. Dian tidak perlu memahami mengapa; ia hanya perlu mematuhinya.

Inilah alasan aturan semacam itu ditulis. Bukan karena orang yang berjaga tidak kompeten, melainkan karena pada pukul satu pagi setelah tiga jam menangani kegagalan, bahkan orang yang kompeten mengambil keputusan yang lebih buruk daripada biasanya. Aturan dua orang bukan penghinaan terhadap kemampuan siapa pun; ia pengakuan atas keterbatasan manusia dalam kondisi tertentu.

154.6 Pukul 01:30 — Manajer terbangun

Manajer TI, seorang perempuan bernama Ratna, dihubungi dan bangun dalam lima menit. Ia meminta tiga angka sebelum apa pun: berapa rekening sudah terposting, berapa total nilai yang sudah masuk, dan apakah nilai yang sudah masuk benar.

Dua angka pertama tersedia dari tabel log bunga. Angka ketiga tidak, karena belum ada yang memeriksanya.

Ratna meminta pemeriksaan itu dilakukan lebih dulu, dengan mengambil sepuluh rekening acak dari yang sudah terposting dan menghitung ulang bunganya secara manual.

Hasilnya: kesepuluhnya benar. Kegagalan terjadi pada cabang kode yang hanya dijalankan untuk produk tertentu, dan seluruh rekening yang sudah terposting berasal dari produk lain yang tidak menyentuh cabang itu.

Catatan atas titik ini. Pertanyaan ketiga Ratna adalah pertanyaan yang membedakan penanganan yang baik dari penanganan yang tampak baik. Mengetahui *berapa banyak* yang sudah terproses itu mudah. Mengetahui apakah yang sudah terproses *benar* menuntut pekerjaan tambahan yang di bawah tekanan waktu sangat menggoda untuk dilewati.

Bila pemeriksaan itu dilewati dan ternyata nilainya salah, melanjutkan posting untuk sisanya akan menggandakan masalah alih-alih menyelesaikannya. Lima belas menit yang dipakai untuk memeriksa sepuluh rekening bisa menghemat berhari-hari koreksi.

154.7 Pukul 02:10 — Jalan yang dipilih

Dengan informasi bahwa yang sudah terposting benar, pilihan menjadi jelas dan aman: lanjutkan dari titik berhenti, jangan ulang dari awal.

Tetapi program tidak dirancang untuk itu. Ia memproses seluruh rekening aktif dari awal setiap kali dijalankan, tanpa penanda per rekening yang menunjukkan mana yang sudah selesai.

Yang tersedia adalah tabel log bunga, yang mencatat setiap rekening yang sudah diposting beserta nilainya. Tabel itu dibuat bertahun-tahun sebelumnya untuk keperluan pelaporan, bukan untuk pemulihan. Tetapi malam itu ia menjadi penyelamat.

Arif menulis kueri yang memakai tabel log itu sebagai penanda, lalu menjalankan program dengan parameter yang membatasi pemrosesan hanya pada rekening yang belum ada di log. Penulisan dan pengujiannya memakan empat puluh menit — di lingkungan uji, dengan Ratna dan Dian mengawasi, dan dengan salinan data yang direstore khusus untuk itu.

Fase akrual selesai pukul 03:41. Rekonsiliasi menunjukkan total log bunga cocok dengan total posting ke buku besar. Empat fase berikutnya berjalan tanpa masalah. Layanan dibuka tepat waktu pukul 08:00.

154.8 Apa yang berubah setelahnya

Analisis pasca-insiden yang dilakukan empat hari kemudian menghasilkan enam tindakan. Yang menarik adalah bahwa hanya satu di antaranya bersifat teknis murni.

Tindakan	Jenis	Tenggat
Tambahkan penanda per rekening di fase akrual	Teknis	1 bulan
Tambahkan validasi kalender ke pemeriksaan pra-EOD	Teknis	2 pekan
Wajibkan tiket perubahan untuk tabel parameter apa pun	Proses	Segera
Petakan seluruh program yang bergantung pada tabel kalender	Dokumentasi	1 bulan
Perbaiki runbook: cara membaca pesan tingkat kedua	Dokumentasi	1 pekan
Latih operator jaga membaca job log	Kemampuan	2 bulan

Tindakan ketiga adalah yang paling murah dan paling berdampak. Tabel parameter aplikasi diperlakukan sebagai data oleh tim bisnis, sehingga perubahannya tidak melewati manajemen perubahan. Tetapi program bergantung padanya seperti bergantung pada kode. Perubahan pada tabel parameter adalah perubahan pada perilaku sistem, dan layak diperlakukan demikian.

154.9 Empat pertanyaan untuk direnungkan

Sebelum melanjutkan ke narasi berikutnya, pertimbangkan pertanyaan-pertanyaan ini dalam konteks sistem Anda sendiri.

Pertama. Bila fase akrual di sistem Anda gagal di tengah malam ini, apakah ada penanda yang memungkinkan Anda melanjutkan dari titik berhenti? Bila tidak, restore adalah satu-satunya pilihan Anda, dan Anda baru akan mengetahuinya pada malam ketika itu terjadi.

Kedua. Apakah tabel parameter aplikasi di sistem Anda dilindungi manajemen perubahan? Bila tidak, ada jalur yang bisa mengubah perilaku sistem tanpa persetujuan siapa pun.

Ketiga. Apakah operator jaga Anda bisa membaca job log dan menemukan nomor baris kegagalan tanpa bertanya? Bila tidak, setiap insiden malam akan menanggung penundaan yang sama dengan sebelas menit dalam narasi ini.

Keempat. Apakah aturan dua orang untuk keputusan berisiko tinggi tertulis, atau hanya dipahami? Aturan

yang tidak tertulis akan dilanggar pada malam ketika orang yang memahaminya tidak berjaga.

Bab 155. Narasi Kedua: Cabang yang Lambat Selama Tiga Pekan

155.1 Pekan pertama — keluhan yang diabaikan

Keluhan pertama masuk pada Selasa sore melalui meja bantuan. Seorang teller di cabang di pinggiran kota melaporkan bahwa transaksi terasa lambat. Tiket dibuat dengan tingkat keparahan rendah, dan ditutup dua hari kemudian dengan catatan bahwa tidak ada masalah ditemukan pada sistem pusat.

Penutupan itu secara teknis benar. Seluruh metrik di IBM i normal: CPU di bawah lima puluh persen, disk tidak sibuk, tidak ada job dalam status menunggu, waktu respons rata-rata sesuai target. Orang yang menangani tiket memeriksa semuanya dan tidak menemukan apa pun.

Catatan atas titik ini. Metrik rata-rata pada sistem pusat tidak akan pernah menunjukkan masalah yang hanya memengaruhi satu dari tiga ratus cabang. Satu cabang menyumbang sepertiga persen dari total beban; masalah di sana tenggelam sempurna dalam angka agregat.

Ini keterbatasan struktural dari pemantauan berbasis sistem, dan bukan kelemahan yang bisa diperbaiki dengan menambah metrik pusat. Yang dibutuhkan adalah pengukuran dari sisi pengguna — dan itu jenis pemantauan yang sangat jarang dipasang karena jauh lebih rumit.

155.2 Pekan kedua — pola mulai terlihat

Keluhan kedua dan ketiga masuk dari cabang yang sama. Kali ini disertai detail yang lebih berguna: transaksi setoran tunai terasa normal, tetapi pencetakan rekening koran dan pengiriman berkas akhir hari memakan waktu berkali-kali lebih lama.

Seorang staf meja bantuan yang teliti menyadari bahwa ketiga tiket berasal dari cabang yang sama, dan mengeskalasikannya ke tim infrastruktur.

Detail baru itu sesungguhnya sangat informatif, tetapi nilainya baru terlihat kemudian. Transaksi setoran mengirim pesan kecil. Pencetakan rekening koran dan pengiriman berkas mengirim data dalam jumlah besar. Perbedaan perilaku antara keduanya menunjuk langsung ke satu kelas masalah.

Catatan atas titik ini. Pola "operasi kecil berhasil, operasi besar lambat atau gagal" adalah tanda tangan khas masalah MTU atau fragmentasi paket. Bab 68.8 mencantumkannya, tetapi tidak ada seorang pun di rantai penanganan yang pernah membaca tabel itu, karena tabel itu ada di dokumentasi tim infrastruktur sementara tiket ditangani oleh meja bantuan.

Pengetahuan yang tersedia tetapi tidak terjangkau oleh orang yang membutuhkannya sama tidak bergunanya dengan pengetahuan yang tidak ada. Ini masalah distribusi, bukan masalah kompetensi.

155.3 Pekan ketiga — dugaan yang menghabiskan waktu

Tim infrastruktur mengambil tiket dan memulai penyelidikan dengan dugaan yang paling lazim: bandwidth ke cabang itu tidak memadai.

Mereka menghubungi penyedia jaringan, meminta laporan pemakaian bandwidth, dan menunggu tiga hari kerja. Laporan yang datang menunjukkan pemakaian rata-rata dua puluh dua persen dari kapasitas, dengan puncak empat puluh satu persen. Bandwidth bukan masalahnya.

Dugaan kedua: perangkat di cabang bermasalah. Seorang teknisi dikirim, memeriksa switch dan router, mengganti satu kabel yang terlihat aus, dan melaporkan tidak ada temuan. Dua hari lagi berlalu.

Dugaan ketiga: masalah pada aplikasi klien di terminal teller. Tim aplikasi memeriksa versi perangkat lunak di seluruh terminal cabang itu dan membandingkannya dengan cabang lain. Identik.

Pada akhir pekan ketiga, sebelas hari kerja telah dihabiskan, tiga tim telah terlibat, dan masalahnya masih ada.

Catatan atas titik ini. Tiga dugaan itu diuji secara berurutan, masing-masing memakan dua sampai tiga hari, dan tidak ada satu pun yang didahului oleh pengukuran murah yang bisa menyingkirkan atau menguatkannya.

Uji MTU dengan `PING PKTLEN(1472)` memakan waktu lima detik dan bisa dijalankan pada hari pertama. Ia tidak dijalankan karena tidak ada seorang pun yang menghubungkan gejala dengan kemungkinan itu, dan karena urutan diagnosis yang dipakai berdasarkan kelaziman penyebab, bukan berdasarkan biaya pengujian.

Ini prinsip yang layak dijadikan kebiasaan: **ketika beberapa dugaan sama-sama masuk akal, uji yang paling murah lebih dulu, bukan yang paling mungkin.** Sebuah dugaan yang kemungkinannya hanya sepuluh persen tetapi bisa diuji dalam lima detik harus diuji sebelum dugaan yang kemungkinannya lima puluh persen tetapi menuntut tiga hari menunggu laporan pihak ketiga.

155.4 Hari kedua puluh — lima detik yang menyelesaikannya

Seorang System Engineer yang baru bergabung di tim, dan yang belum terlibat dalam penyelidikan sebelumnya, diminta melihat tiket itu karena ia sedang tidak sibuk.

Ia membaca riwayat tiket, memperhatikan detail bahwa operasi kecil berhasil dan operasi besar lambat, lalu menjalankan dua perintah:

```
PING RMTSYS('10.40.12.5') PKTLEN(64)
PING RMTSYS('10.40.12.5') PKTLEN(1472)
```

Yang pertama berhasil. Yang kedua gagal seluruhnya.

Penelusuran lebih lanjut menemukan bahwa penyedia jaringan telah mengganti perangkat di salah satu titik transit jalur ke cabang itu, dua puluh dua hari sebelumnya. Perangkat baru punya MTU lebih kecil dan tidak meneruskan pesan ICMP yang memberi tahu pengirim untuk memperkecil paket. Akibatnya paket besar dibuang tanpa pemberitahuan, dan pengiriman hanya berhasil setelah serangkaian timeout dan pengiriman ulang.

Penyedia menyesuaikan konfigurasi dalam empat jam setelah dilaporkan dengan bukti yang spesifik. Masalah yang berlangsung tiga pekan selesai dalam setengah hari kerja begitu penyebabnya diketahui.

155.5 Mengapa orang baru yang menemukannya

Ini bagian yang paling layak direnungkan, dan penjelasannya bukan tentang kemampuan individu.

Tiga tim sebelumnya masuk ke penyelidikan dengan konteks: pengalaman tentang penyebab yang lazim, hubungan dengan penyedia yang perlu dijaga, dan dugaan awal yang terbentuk dari kasus-kasus serupa di masa lalu. Konteks itu biasanya membantu, dan itulah sebabnya pengalaman berharga.

Tetapi konteks juga mempersempit. Seseorang yang telah menangani lima puluh kasus cabang lambat yang semuanya disebabkan bandwidth akan mendekati kasus kelima puluh satu dengan dugaan yang sama, dan akan menghabiskan waktu untuk menguatkan dugaan itu alih-alih menyingkirkannya.

Orang yang baru tidak punya beban itu. Ia membaca gejala apa adanya, mencocokkannya dengan pengetahuan umum yang baru ia pelajari, dan menguji hal yang paling murah.

Konsekuensi praktis untuk tim Anda. Ketika sebuah penyelidikan berjalan lebih dari beberapa hari tanpa kemajuan, memasukkan orang yang belum terlibat bukan tanda ketidakpercayaan terhadap yang sudah bekerja. Ia adalah teknik diagnosis. Sediakan mekanisme untuk itu, dan buat agar tidak terasa sebagai penghinaan bagi siapa pun.

155.6 Perubahan yang diterapkan

Tindakan	Alasan
Uji MTU masuk daftar periksa diagnosis jaringan, di urutan atas	Biayanya lima detik
Tabel gejala dan penyebab jaringan disebarkan ke meja bantuan	Pengetahuan harus terjangkau yang membutuhkan
Penyedia jaringan wajib memberi tahu setiap perubahan perangkat transit	Perubahan tak terkomunikasi adalah pola nomor satu
Tiket dari cabang yang sama lebih dari dua kali sebulan dieskalasi otomatis	Pola tidak boleh bergantung pada ketelitian individu
Aturan: bila penyelidikan lewat tiga hari, masukkan orang baru	Konteks mempersempit

Butir ketiga adalah yang paling sering diabaikan dalam kontrak dengan penyedia jaringan. Kewajiban memberi tahu perubahan konfigurasi biasanya tidak tercantum, karena dari sudut pandang penyedia, penggantian perangkat transit adalah pemeliharaan rutin yang tidak mengubah layanan yang dijanjikan. Dari sudut pandang Anda, ia mengubah perilaku sistem.

💾 Bab 156. Narasi Ketiga: Backup yang Enam Bulan Tidak Lengkap

156.1 Bagaimana ia dimulai

Bulan Maret, sebuah library baru dibuat di produksi untuk menampung data modul baru yang mulai dipakai. Pembuatannya melalui tiket perubahan yang lengkap, disetujui, dan didokumentasikan.

Tiket itu mencantumkan sebelas langkah. Satu langkah yang tidak tercantum: menambahkan library baru ke daftar backup.

Tidak ada yang lalai. Templat tiket untuk pembuatan library tidak memuat langkah itu, karena templat itu disusun bertahun-tahun sebelumnya ketika seluruh library produksi tercakup oleh satu control group yang memakai pola `*ALLUSR`. Pada suatu titik di masa lalu, seseorang mengubah control group agar menyebut library satu per satu — kemungkinan untuk mengecualikan library kerja yang besar dan tidak perlu di-backup. Perubahan itu sah dan masuk akal pada saat dilakukan. Tetapi ia mengubah asumsi yang mendasari templat tiket, dan templat itu tidak diperbarui.

Catatan atas titik ini. Ini contoh dari kelas masalah yang sangat sulit dicegah: sebuah keputusan yang benar di masa lalu membuat prosedur lain menjadi salah, tanpa ada yang menyadari hubungan di antara keduanya.

Satu-satunya pertahanan yang efektif bukan kehati-hatian individu, melainkan pemeriksaan otomatis yang membandingkan kondisi nyata dengan kondisi yang seharusnya. Kueri yang membandingkan daftar library yang ada dengan daftar library yang muncul di riwayat backup akan menemukan ini dalam sepekan.

156.2 Enam bulan pesan yang tidak dibaca

Setiap malam sejak Maret, job backup menghasilkan pesan `CPF3730` yang menyatakan bahwa sebagian objek tidak disimpan.

Setiap pagi, laporan otomatis mencantumkan status backup sebagai selesai. Laporan itu memeriksa apakah job berakhir normal, dan job memang berakhir normal — kegagalan menyimpan sebagian objek tidak membuat job gagal.

Seratus delapan puluh tiga pesan `CPF3730` tercatat di history log dan tidak pernah dibaca.

Catatan atas titik ini. Ini kegagalan diam dalam bentuk paling murni, dan ia punya dua lapisan.

Lapisan pertama: pesan ada tetapi tidak dibaca. Ini bisa diperbaiki dengan menambahkan pemeriksaan pesan itu ke daftar verifikasi harian.

Lapisan kedua, yang lebih penting: laporan status memeriksa hal yang salah. Ia memeriksa apakah proses berjalan, bukan apakah proses menghasilkan apa yang seharusnya. Perbedaan antara keduanya adalah perbedaan antara memantau proses dan memantau hasil, dan hampir semua sistem pemantauan yang dibangun tergesa-gesa memilih yang pertama karena lebih mudah.

Ukuran backup harian adalah contoh pemantauan hasil yang sangat murah. Bila ukuran total turun dua puluh persen tanpa penjelasan, ada yang tidak ikut tersimpan. Pemeriksaan sesederhana itu akan menemukan masalah ini pada hari kedua.

156.3 September — uji restore triwulanan

Masalah ditemukan pada uji restore triwulanan, yang dijadwalkan sebagai bagian dari program kepatuhan dan dijalankan oleh seorang System Engineer bernama Hendra.

Skenario uji: restore satu library produksi ke lingkungan uji, jalankan aplikasi, verifikasi data. Library yang dipilih untuk triwulan itu adalah library modul baru — dipilih justru karena ia yang paling baru dan belum pernah diuji.

Restore gagal pada langkah pertama. Library itu tidak ada di media mana pun.

Hendra memeriksa riwayat BRMS. Tidak ada satu pun catatan penyimpanan untuk library itu sejak ia dibuat pada bulan Maret.

Catatan atas titik ini. Uji restore triwulanan adalah program yang di banyak institusi diperlakukan sebagai formalitas kepatuhan: dijalankan karena diwajibkan, dilaporkan sebagai lulus, dan tidak dianggap sebagai pekerjaan yang menghasilkan nilai.

Hari itu ia menghasilkan nilai yang sangat besar. Enam bulan data produksi tanpa backup adalah risiko yang, bila terwujud, akan menjadi insiden pelaporan ke regulator dan kemungkinan kehilangan data yang tidak dapat dipulihkan.

Yang layak diperhatikan: masalah ditemukan bukan oleh pemantauan, bukan oleh audit, bukan oleh peninjauan proses. Ia ditemukan oleh satu-satunya kegiatan yang benar-benar menguji apakah kemampuan pemulihan itu ada.

156.4 Enam jam berikutnya

Hendra mengeskalasi segera, dan penanganannya mengikuti urutan yang tepat.

Waktu	Tindakan
09:15	Temuan dilaporkan ke manajer TI
09:30	Backup darurat library itu dijalankan dengan `SAVACT(*SYNCLIB)`
10:45	Backup selesai dan diverifikasi dapat dibaca
11:00	Control group diperbarui untuk menyertakan library itu
11:30	Kueri pembanding library dijalankan untuk seluruh sistem
12:15	Ditemukan dua library lain dengan masalah yang sama
14:00	Ketiganya tercakup, backup diverifikasi
15:30	Laporan insiden disusun

Langkah pukul 11:30 adalah yang paling bernilai dari seluruh rangkaian. Setelah menemukan satu kasus, Hendra tidak berhenti pada perbaikan kasus itu, melainkan mencari apakah pola yang sama ada di tempat lain. Ia menemukan dua lagi.

Catatan atas titik ini. Ini kebiasaan yang layak dilatih hingga menjadi naluri. Setiap kali Anda menemukan satu masalah yang berasal dari celah prosedur, kemungkinan besar celah itu telah menghasilkan lebih dari satu masalah. Memperbaiki satu dan berhenti berarti meninggalkan sisanya untuk ditemukan nanti, dalam kondisi yang mungkin jauh kurang menguntungkan.

Daftar tindakan pasca-insiden di Bab 92.3 mencantumkan langkah ini dengan tenggat tiga puluh hari. Dalam narasi ini ia dilakukan dalam tiga puluh menit, dan itu jauh lebih baik.

156.5 Pertanyaan yang lebih besar

Laporan insiden menyampaikan temuan dan perbaikan, tetapi manajer TI mengajukan pertanyaan yang lebih dalam pada rapat tinjauan: **berapa banyak lagi hal seperti ini yang tidak kita ketahui?**

Pertanyaan itu menghasilkan program pemeriksaan yang kemudian menjadi rutin, dan yang menemukan hal-hal berikut dalam dua pekan pertama:

Temuan	Jumlah
Library tanpa backup	3
Tabel produksi tanpa journaling	17
Kolom teks berCCSID 65535	41
Constraint dalam status check pending	6
Program tanpa sumber yang diketahui	28
Profil `*ALLOBJ` tidak dipakai lebih dari 90 hari	4
Sertifikat kedaluwarsa dalam 60 hari	2

Tidak satu pun dari temuan ini menyebabkan insiden pada saat ditemukan. Semuanya adalah insiden yang belum terjadi.

Catatan penutup narasi ini. Daftar di atas hampir pasti punya padanan di sistem Anda. Kueri untuk menemukannya ada di PR-01 pada Bab 151, dan menjalankannya memakan waktu kurang dari satu hari kerja.

Yang membuat orang tidak menjalankannya bukan kesulitan teknis, melainkan ketiadaan pemicu. Tidak ada yang meminta, tidak ada tiket, tidak ada keluhan. Pekerjaan pencegahan selalu bersaing dengan pekerjaan yang mendesak, dan selalu kalah kecuali seseorang memutuskan untuk melindungi waktunya.

💾 Bab 157. Narasi Keempat: Uji Pemulihan yang Gagal Sebagian

157.1 Persiapan yang terlihat menyeluruh

Uji pemulihan bencana tahunan dijadwalkan pada Sabtu di bulan Oktober. Persiapan berlangsung enam pekan dan mencakup hal-hal berikut: konfigurasi DR diverifikasi identik dengan produksi, prosedur switchover ditulis dan ditinjau, peran ditetapkan dengan nama, jendela diumumkan, dan pembekuan perubahan diberlakukan sejak sepekan sebelumnya.

Daftar periksa verifikasi memuat dua puluh tiga butir. Seluruhnya ditinjau bersama tim dua hari sebelum pelaksanaan.

Secara wajar, persiapan ini layak disebut menyeluruh. Ia jauh lebih baik daripada yang dilakukan sebagian besar institusi, yang sering memperlakukan uji DR sebagai kegiatan setengah hari tanpa persiapan berarti.

157.2 Sabtu, 06:00 — Pelaksanaan dimulai

Switchover dijalankan sesuai prosedur dan berhasil tanpa hambatan. Kanal ditutup pukul 06:00, lag replikasi diverifikasi nol pukul 06:20, perpindahan peran selesai pukul 06:35, iASP aktif di node DR pukul 06:41, dan subsystem aplikasi berjalan pukul 06:52.

Angka kontrol diverifikasi: jumlah rekening, total saldo, jumlah transaksi hari sebelumnya. Seluruhnya cocok persis.

Pukul 07:10, tim menyatakan verifikasi teknis lulus. Suasana di ruang operasi baik. Mereka lebih cepat dua

puluh menit dari perkiraan.

157.3 Pukul 07:30 — Butir yang tidak ada di daftar

Uji kanal dimulai. Teller cabang percontohan berhasil melakukan setoran. ATM berhasil melakukan cek saldo dan tarik tunai. Internet banking berhasil.

Mobile banking gagal.

Seluruh permintaan dari aplikasi mobile mengembalikan kesalahan koneksi. Tim digital dihubungi, dan setelah dua puluh menit pemeriksaan melaporkan bahwa middleware mobile tidak dapat menyambung ke basis data.

System Engineer memeriksa dari sisi IBM i: server database berjalan, port terbuka, koneksi dari alamat lain berhasil. Tidak ada masalah yang terlihat.

Pukul 08:15, penyebabnya ditemukan. Middleware mobile memakai daftar putih alamat IP untuk keamanan. Daftar itu memuat alamat partisi produksi. Partisi DR punya alamat berbeda, dan alamat itu tidak pernah didaftarkan.

157.4 Mengapa ini tidak tertangkap

Pertanyaan yang diajukan pada analisis pasca-kegiatan: persiapan enam pekan dengan dua puluh tiga butir verifikasi, bagaimana hal sesederhana ini bisa terlewat?

Jawabannya menyingkap sesuatu yang berlaku umum. Daftar periksa itu disusun oleh tim infrastruktur, dan mencakup dengan sangat teliti seluruh hal yang berada dalam kendali tim infrastruktur: partisi, penyimpanan, jaringan internal, sistem operasi, basis data, subsystem, dan aplikasi core banking.

Daftar putih alamat IP di middleware mobile berada dalam kendali tim digital. Tim digital tidak dilibatkan dalam penyusunan daftar periksa; mereka hanya diminta menguji kanalnya pada hari pelaksanaan.

Catatan atas titik ini. Batas antar tim adalah tempat di mana verifikasi paling sering bocor. Bukan karena ada yang lalai, melainkan karena setiap tim memverifikasi wilayahnya dengan baik dan tidak ada yang memverifikasi ruang di antaranya.

Perbaikan strukturalnya bukan menambah butir ke daftar, melainkan mengubah cara daftar disusun: setiap tim yang punya ketergantungan pada sistem menyumbang butir verifikasinya sendiri, dan daftar gabungan ditinjau bersama.

Ini pola yang berlaku jauh melampaui uji DR. Setiap kali sebuah daftar periksa disusun oleh satu pihak untuk memverifikasi sistem yang disentuh banyak pihak, celah akan muncul persis di batas tanggung jawab.

157.5 Pukul 08:15 — Keputusan di bawah jendela yang menipis

Jendela uji berakhir pukul 12:00, termasuk waktu untuk switchback dan verifikasi. Artinya tersisa sekitar dua jam untuk menyelesaikan masalah dan menuntaskan sisa pengujian.

Tim digital bisa menambahkan alamat DR ke daftar putih dalam sekitar tiga puluh menit, tetapi perubahan itu

sendiri menuntut tiket perubahan pada sistem mereka, yang pada hari Sabtu memerlukan persetujuan khusus.

Komandan uji, seorang manajer bernama Bagas, menghadapi tiga pilihan.

Pilihan pertama: tunggu persetujuan, lakukan perubahan, lanjutkan pengujian. Risikonya jendela terlampaui, dan switchback yang dilakukan tergesa membawa risiko sendiri.

Pilihan kedua: lewati pengujian mobile banking, catat sebagai temuan, lanjutkan ke sisa pengujian dan switchback sesuai jadwal. Risikonya uji tidak lengkap.

Pilihan ketiga: hentikan uji sekarang, lakukan switchback, jadwalkan ulang setelah temuan diperbaiki.

Bagas memilih yang kedua, dan menurut penilaian saya itu pilihan yang tepat — meskipun ia bukan pilihan yang paling nyaman untuk dilaporkan.

Alasan mengapa pilihan kedua tepat. Tujuan uji DR bukan mendapatkan laporan yang menyatakan semuanya berhasil. Tujuannya menemukan apa yang tidak akan bekerja saat bencana nyata. Temuan mobile banking adalah hasil uji yang bernilai tinggi, dan ia sudah diperoleh. Menghabiskan sisa jendela untuk memperbaikinya lalu membuktikan bahwa perbaikan itu bekerja memberi nilai tambahan yang kecil, sementara mengorbankan pengujian kanal lain yang belum dilakukan.

Pilihan pertama, yang terasa paling bertanggung jawab, sesungguhnya yang paling berisiko: ia mempertaruhkan switchback yang tergesa demi menyempurnakan laporan.

Pilihan ketiga terlalu mahal. Jendela uji DR sulit didapat, menuntut persetujuan banyak pihak, dan membatalkannya berarti menunggu berbulan-bulan untuk kesempatan berikutnya — sementara seluruh kanal lain belum diuji sama sekali.

157.6 Sisa hari

Pengujian kanal lain dilanjutkan. Sistem pembayaran antar bank berhasil. Sistem kartu berhasil setelah penyesuaian kecil pada timeout. Pelaporan berhasil.

Satu siklus EOD dijalankan di lingkungan DR dan selesai dalam 3 jam 22 menit, dibandingkan 3 jam 10 menit di produksi — selisih dua belas menit yang dicatat sebagai temuan minor dan kemudian ditelusuri ke perbedaan alokasi memori antar partisi.

Switchback dimulai pukul 11:05 dan selesai pukul 11:48. Angka kontrol kembali diverifikasi dan cocok.

RTO aktual yang tercatat: dari deklarasi sampai layanan siap dibuka, 1 jam 10 menit. RTO yang tertulis dalam dokumen kesinambungan usaha institusi itu: 2 jam. Aktualnya lebih baik dari yang dijanjikan.

Catatan atas angka RTO. Hasil yang lebih baik dari target terdengar seperti kabar baik tanpa syarat, tetapi ia layak diperiksa. Uji dilakukan pada Sabtu pagi dengan seluruh tim siap, seluruh peran terisi, dan pembekuan perubahan berlaku sepekan. Bencana nyata tidak datang dengan persiapan enam pekan.

RTO aktual dalam kondisi terkendali adalah batas bawah, bukan perkiraan. Institusi yang matang mencatatnya sebagai "RTO dalam kondisi latihan" dan memberi margin untuk kondisi nyata.

157.7 Laporan yang ditulis dengan benar

Laporan uji DR menyatakan hasilnya sebagai **lulus dengan temuan**, bukan lulus, dan bukan gagal. Lima temuan dicatat:

Temuan	Tingkat	Tindakan
Daftar putih IP middleware mobile tidak memuat alamat DR	Tinggi	Virtual IP yang berpindah saat failover
Daftar periksa tidak melibatkan tim di luar infrastruktur	Tinggi	Ubah cara daftar disusun
Timeout sistem kartu perlu penyesuaian	Sedang	Selaraskan konfigurasi kedua sisi
EOD di DR 12 menit lebih lambat	Rendah	Bandingkan alokasi sumber daya partisi
Perubahan pada sistem mitra sulit dilakukan di akhir pekan	Sedang	Sepakati prosedur khusus untuk jendela uji

Catatan tentang cara melaporkan. Laporan yang menyatakan lulus tanpa temuan dari sebuah uji DR adalah laporan yang patut dicurigai. Ia berarti salah satu dari dua hal: ujinya tidak cukup menyeluruh, atau temuannya tidak dilaporkan.

Uji yang baik selalu menghasilkan temuan, karena sistem nyata selalu punya celah. Nilai dari uji terletak pada temuannya, bukan pada kesimpulannya. Manajemen yang memahami ini akan menerima laporan dengan lima temuan sebagai kabar baik; manajemen yang tidak memahaminya akan menekan tim untuk menghasilkan laporan bersih, dan dengan begitu merusak nilai dari seluruh kegiatan.

Bila Anda berada dalam organisasi yang menekan ke arah laporan bersih, ada satu cara membingkai ulang yang sering berhasil: sajikan temuan bukan sebagai kegagalan uji, melainkan sebagai hasil yang dicari. Uji DR adalah investasi untuk menemukan celah; uji yang tidak menemukan apa-apa adalah investasi yang tidak menghasilkan.

157.8 Dua belas bulan kemudian

Uji DR tahun berikutnya dilakukan dengan daftar periksa yang disusun bersama enam tim. Jumlah butirnya naik dari dua puluh tiga menjadi enam puluh satu.

Uji itu menghasilkan tiga temuan baru, tidak satu pun terkait daftar putih alamat IP karena virtual IP sudah diterapkan. Dua di antaranya berada di batas antar tim yang berbeda dari tahun sebelumnya.

Ini bukan tanda bahwa perbaikan tahun sebelumnya gagal. Ini tanda bahwa sistem nyata cukup kompleks sehingga selalu ada celah baru, dan bahwa satu-satunya cara menemukannya adalah dengan menguji secara berkala dan menerima bahwa setiap uji akan menghasilkan temuan.

Institusi yang matang dalam hal ini tidak mengukur keberhasilan dari jumlah temuan yang nol, melainkan dari dua hal lain: kecepatan temuan ditutup, dan apakah temuan tahun ini berbeda dari temuan tahun lalu. Temuan yang berulang berarti perbaikan tidak berjalan; temuan yang selalu baru berarti Anda menggali lebih dalam setiap tahun.

Bab 158. Benang Merah Empat Narasi

158.1 Apa yang berulang

Keempat narasi berasal dari domain yang berbeda — batch, jaringan, backup, dan pemulihan bencana. Tetapi pola yang mendasarinya sangat mirip.

Pertama, penyebabnya hampir selalu perubahan yang tidak terkomunikasi. Baris kalender yang dihapus, perangkat transit yang diganti, control group yang diubah bertahun-tahun lalu, daftar putih yang tidak diperbarui. Tidak satu pun dari keempatnya adalah kegagalan perangkat keras atau cacat perangkat lunak. Semuanya adalah keputusan manusia yang benar dalam konteksnya sendiri dan menghasilkan akibat di tempat lain.

Kedua, informasi yang dibutuhkan hampir selalu sudah tersedia. Job log ada sejak pukul 23:52. Pesan `CPF3730` tercatat seratus delapan puluh tiga kali. Gejala operasi besar yang gagal disebutkan di tiket pekan kedua. Yang kurang bukan data, melainkan perhatian terhadap data yang ada.

Ketiga, keputusan terpenting selalu diambil di bawah tekanan waktu dan informasi tidak lengkap. Ini tidak bisa dihindari. Yang bisa dilakukan adalah menyiapkan kerangka keputusan sebelumnya, ketika tidak ada tekanan, sehingga yang perlu dilakukan saat tekanan datang hanyalah mengikutinya.

Keempat, yang menyelamatkan sering bukan keahlian, melainkan prosedur. Aturan dua orang. Uji restore triwulanan. Verifikasi angka kontrol. Semuanya terasa birokratis sampai malam ketika ia menjadi satu-satunya hal yang mencegah keputusan buruk.

158.2 Apa yang membedakan penanganan baik dari yang buruk

Dalam keempat narasi, penanganan yang baik punya ciri yang sama.

Ciri	Wujudnya dalam narasi
Mengamati sebelum menduga	Job log dibaca, bukan hipotesis diuji
Memeriksa kebenaran, bukan hanya jumlah	Sepuluh rekening dihitung ulang manual
Mencari pola yang sama di tempat lain	Dua library lain ditemukan
Memilih uji termurah lebih dulu	Ping MTU lima detik
Melaporkan temuan apa adanya	Lulus dengan temuan, bukan lulus
Menghormati aturan meski tidak memahaminya	Dian menahan rerun

Tidak satu pun dari keenam ciri ini menuntut keahlian teknis yang langka. Semuanya adalah kebiasaan yang bisa dilatih, dan yang nilainya justru paling terasa pada orang yang belum berpengalaman.

158.3 Cara memakai bagian ini untuk melatih tim

Saya sarankan pola berikut, yang menuntut sekitar sembilan puluh menit per sesi.

1. Bagikan satu narasi kepada peserta, dipotong sampai titik keputusan utama
2. Minta masing-masing menuliskan keputusan mereka, sendiri, tanpa diskusi
3. Kumpulkan jawabannya sebelum ada yang berbicara
4. Bacakan kelanjutan narasi

5. Diskusikan perbedaan antara jawaban peserta dan apa yang terjadi
6. Tutup dengan pertanyaan: apakah sistem kita punya masalah yang sama?

Langkah kedua dan ketiga penting dan sering dilewati. Bila diskusi dibuka sebelum jawaban dikumpulkan, orang yang paling senior atau paling lantang akan mengarahkan seluruh ruangan, dan Anda kehilangan informasi tentang apa yang sesungguhnya dipikirkan anggota tim yang lain.

Langkah keenam adalah yang menghasilkan nilai jangka panjang. Narasi ini bukan tentang institusi rekaan dalam cerita; ia alat untuk memeriksa institusi Anda sendiri.

Bagian XLVIII — Esai Teknis

Bab-bab sebelumnya menjelaskan cara mengerjakan sesuatu. Bagian ini membahas mengapa hal-hal itu dikerjakan dengan cara demikian, dan apa yang terjadi bila alasannya dilupakan sementara prosedurnya dipertahankan.

Esai bukan bentuk yang lazim dalam buku teknis, dan saya menyadari sebagian pembaca akan melewatinya. Tetapi ada jenis pemahaman yang tidak bisa disampaikan lewat daftar langkah maupun tabel. Seseorang yang tahu bahwa fase akrual tidak boleh diulang akan mematuhi aturan itu; seseorang yang memahami mengapa akan mengenali situasi baru yang punya sifat serupa, meskipun tidak ada aturan yang mencakupnya.

Bab 159. Mengapa Integritas Dibangun ke Dalam, Bukan Ditambahkan

159.1 Sebuah keputusan yang diambil pada 1970-an

Ketika System/38 dirancang, keputusan yang paling berpengaruh bukanlah keputusan tentang kecepatan atau kapasitas, melainkan tentang di mana meletakkan jaminan integritas.

Pada sebagian besar arsitektur, integritas data adalah tanggung jawab lapisan atas. Sistem operasi menyediakan penyimpanan byte, basis data ditambahkan di atasnya sebagai produk terpisah, dan aplikasi memakai basis data itu bila programmernya memilih demikian. Setiap lapisan bisa dilewati oleh lapisan di atasnya, dan biasanya ada cara untuk menulis langsung ke penyimpanan tanpa melewati basis data sama sekali.

Pendekatan itu punya kelebihan nyata: fleksibilitas, kemampuan mengganti komponen, dan kemungkinan mengoptimasi jalur tertentu dengan melewati lapisan yang tidak dibutuhkan.

System/38 memilih arah sebaliknya, dan IBM i mewarisinya. Basis data bukan produk di atas sistem operasi; ia bagian dari sistem operasi. Tidak ada cara menulis ke sebuah tabel tanpa melewati mekanisme basis data, karena tidak ada lapisan di bawahnya yang terbuka untuk ditulisi. Objek bertipe, dan tipe itu menentukan operasi apa yang sah, dijaga pada tingkat di bawah sistem operasi sehingga tidak ada program yang bisa melanggarnya.

159.2 Harga yang dibayar

Keputusan itu punya harga, dan jujur menyebutkannya lebih berguna daripada memujinya tanpa syarat.

Platform ini kurang fleksibel. Anda tidak bisa mengganti basis datanya. Anda tidak bisa memilih sistem berkas yang berbeda untuk beban tertentu. Optimasi yang dilakukan dengan melewati lapisan tidak tersedia, karena lapisan itu tidak bisa dilewati. Untuk beban kerja tertentu — pemrosesan data tidak terstruktur dalam jumlah sangat besar, misalnya — ini membuat platform lain jelas lebih cocok.

Harga lain yang lebih halus: karena integritas dijamin di bawah, programmer tidak perlu memikirkannya, dan karena tidak perlu memikirkannya, sebagian tidak memahaminya. Seorang programmer RPG bisa bekerja bertahun-tahun tanpa pernah menghadapi akibat dari penulisan yang tidak atomik, sederhana karena arsitektur telah mencegahnya. Pemahaman itu kemudian tidak terbentuk, dan ia dibutuhkan justru ketika orang itu merancang sesuatu yang berada di luar jangkauan jaminan arsitektural — misalnya rangkaian transaksi yang melintasi activation group berbeda.

159.3 Apa yang didapat sebagai gantinya

Yang didapat adalah kelas kesalahan yang tidak mungkin terjadi.

Sebuah program tidak bisa menulis ke memori milik program lain, karena penunjuk dijaga pada tingkat mesin. Sebuah file data tidak bisa dieksekusi sebagai program, karena tipenya bukan program. Sebuah penulisan tidak bisa menghasilkan record setengah tertulis, karena operasinya atomik pada tingkat di bawah yang bisa disentuh program.

Di lingkungan yang memindahkan uang, kelas-kelas kesalahan itu adalah kelas yang paling mahal. Kerusakan data yang senyap — data yang salah tetapi terlihat sah — jauh lebih merugikan daripada sistem yang berhenti. Sistem yang berhenti diketahui segera dan dipulihkan; data yang salah menyebar ke laporan, ke keputusan, ke saldo nasabah, dan baru ditemukan berbulan-bulan kemudian ketika memperbaikinya jauh lebih rumit.

Inilah alasan sesungguhnya mengapa sistem perbankan masih berjalan di platform ini setelah empat dekade. Bukan karena inersia, bukan karena biaya migrasi, dan bukan semata karena kompatibilitas biner. Melainkan karena sifat yang paling dibutuhkan oleh sistem yang memindahkan uang adalah sifat yang dibangun ke dalam fondasinya, bukan ditambahkan di atasnya dan bisa dilewati.

159.4 Konsekuensi untuk pekerjaan Anda

Bila pemahaman ini diterima, sebagian besar isi buku ini menjadi satu tema tunggal yang diulang dalam banyak konteks.

Mengapa tabel produksi harus dijurnal? Karena tanpa journaling, tabel itu berada di luar jaminan yang membuat platform ini layak dipakai untuk uang. Ia menjadi penyimpanan biasa, dan seluruh sifat yang Anda andalkan tidak berlaku padanya.

Mengapa seluruh program dalam satu rangkaian transaksi harus berada di activation group yang sama? Karena jaminan atomik berlaku dalam satu cakupan commit, dan rangkaian yang melintasi cakupan berbeda

kehilangan jaminan itu tanpa pemberitahuan apa pun.

Mengapa program yang berjalan dengan `*ALLOBJ` berbahaya? Karena wewenang objek adalah bagian dari mekanisme integritas, dan program yang melewatinya beroperasi di luar jaminan.

Mengapa fase akrual yang tidak menyimpan hasilnya menjadi masalah? Karena ia membuang informasi yang dibutuhkan untuk memulihkan keadaan, dan pemulihan keadaan adalah bagian dari integritas dalam pengertian yang lebih luas.

Tugas seorang System Engineer, dirumuskan setajam mungkin, adalah menjaga agar sifat-sifat itu tetap utuh ketika sistem berubah. Setiap tabel baru, setiap program baru, setiap integrasi baru adalah kesempatan bagi sesuatu untuk keluar dari lingkup jaminan. Sebagian besar pemeriksaan rutin yang dibahas di buku ini — tabel tanpa journaling, kolom berCCSID 65535, program OPM dalam rantai transaksional, constraint dalam status pending — adalah cara menemukan hal-hal yang telah keluar dari lingkup itu tanpa ada yang menyadari.

Bab 160. Ekonomi Keandalan

160.1 Mengapa argumen teknis sering kalah

Seorang System Engineer yang mengajukan permintaan anggaran untuk meningkatkan keandalan menghadapi kesulitan struktural yang layak dipahami, karena memahaminya mengubah cara Anda menyusun argumen.

Kesulitan itu bukan bahwa manajemen tidak peduli pada keandalan. Kesulitannya adalah bahwa manfaat dari keandalan berbentuk **peristiwa yang tidak terjadi**, dan peristiwa yang tidak terjadi tidak terlihat, tidak terukur, dan tidak bisa dikreditkan kepada siapa pun.

Sebaliknya, biaya dari keandalan sangat terlihat: angka di dalam usulan anggaran, waktu tim yang dialokasikan, dan jendela pemeliharaan yang mengganggu.

Asimetri ini tidak bisa dihilangkan, tetapi bisa diimbangi. Caranya adalah dengan membuat peristiwa yang tidak terjadi menjadi terlihat.

160.2 Membuat pencegahan terlihat

Pertimbangkan dua laporan bulanan yang berisi informasi yang sama.

Laporan A: "Tidak ada insiden pada bulan ini. Sistem berjalan normal."

Laporan B: "Tidak ada insiden pada bulan ini. Pemantauan mendeteksi tiga kondisi yang sebelumnya akan berkembang menjadi gangguan: pemakaian ASP yang mendekati ambang dan diperbaiki sebelum berdampak, satu interface yang berhenti berjalan dan diketahui dalam dua puluh menit alih-alih pada saat data diminta keesokan harinya, dan satu sertifikat yang akan kedaluwarsa dalam tiga pekan. Uji restore bulanan berhasil dengan durasi 41 menit."

Keduanya melaporkan bulan tanpa insiden. Yang pertama membuat pekerjaan tim tidak terlihat; yang kedua membuatnya terlihat.

Perbedaan ini bukan soal pemasaran diri. Ia soal menyediakan informasi yang dibutuhkan manajemen untuk mengambil keputusan alokasi sumber daya yang benar. Manajemen yang hanya menerima laporan jenis pertama akan menyimpulkan bahwa tim TI tidak banyak melakukan apa-apa, dan pada putaran anggaran berikutnya akan memangkas.

160.3 Menghitung biaya tidak bertindak

Argumen yang paling kuat sering bukan tentang biaya melakukan, melainkan tentang biaya tidak melakukan. Tetapi angka itu harus disusun, karena ia tidak tersedia secara otomatis.

Contoh penyusunan untuk kasus uji restore yang tidak pernah dijalankan:

Unsur	Cara memperkirakan
Kemungkinan dibutuhkan dalam setahun	Dari riwayat industri atau institusi
Kemungkinan gagal bila tidak pernah diuji	Berdasarkan temuan pada uji pertama di institusi lain
Durasi pemulihan bila prosedur bermasalah	Perkiraan konservatif
Biaya layanan berhenti per jam	Volume transaksi, biaya manual, dampak reputasi
Biaya menjalankan uji	Waktu tim, jendela, media

Angka-angka ini tidak perlu presisi, dan mengejar presisi justru melemahkan argumen karena mengundang perdebatan tentang asumsi. Yang dibutuhkan adalah urutan besaran yang jelas: biaya uji terukur dalam puluhan jam kerja, biaya kegagalan pemulihan terukur dalam hari layanan berhenti.

Ketika perbandingannya disajikan secara eksplisit, keputusan menjadi mudah. Ketika ia dibiarkan sebagai asumsi tak tertulis, keputusan diambil berdasarkan apa yang terlihat — dan yang terlihat hanyalah biaya uji.

160.4 Tentang utang keandalan

Ada konsep dalam pengembangan perangkat lunak yang disebut utang teknis: keputusan cepat yang menghemat waktu sekarang dengan biaya yang dibayar kemudian. Konsep serupa berlaku untuk keandalan, dan di lingkungan operasional ia punya sifat yang sedikit berbeda.

Utang teknis pada kode dibayar oleh orang yang mengubah kode itu. Utang keandalan dibayar oleh orang yang berjaga pada malam ketika sesuatu gagal — yang sering bukan orang yang menciptakan utangnya.

Bentuk-bentuk utang keandalan yang paling umum:

Bentuk	Dibayar ketika
Program tanpa penanda idempotensi	Fase gagal di tengah
Tabel tanpa journaling	Data perlu dipulihkan
Runbook yang tidak pernah diuji orang lain	Orang yang menulisnya tidak berjaga
Prosedur DR yang tidak pernah dilatih	Bencana nyata
Pemantauan yang memeriksa proses, bukan hasil	Kegagalan diam berlangsung berbulan-bulan
Pengetahuan yang hanya ada di satu kepala	Orang itu pergi
Ambang yang ditetapkan tanpa baseline	Peringatan palsu membuat tim berhenti membaca

Yang membuat utang jenis ini berbahaya adalah bahwa bunganya tidak terasa sampai tiba-tiba seluruh pokoknya jatuh tempo sekaligus. Sebuah sistem bisa berjalan bertahun-tahun dengan tujuh utang di daftar itu tanpa satu pun insiden, sehingga muncul kesimpulan bahwa utang itu tidak nyata.

Kesimpulan itu salah, tetapi sulit dibantah dengan bukti, karena buktinya baru tersedia setelah terlambat.

160.5 Melindungi waktu untuk pekerjaan pencegahan

Ini masalah yang sudah disinggung di Bab 95.3, dan layak dibahas lebih dalam karena ia adalah akar dari sebagian besar masalah lain.

Tim operasional bekerja dalam lingkaran yang menguatkan diri sendiri. Banyak insiden berarti banyak waktu habis menanganinya. Waktu yang habis berarti tidak ada waktu membangun hal yang mengurangi insiden. Tidak adanya perbaikan struktural berarti insiden tetap banyak.

Lingkaran ini tidak bisa diputus dari dalam oleh orang yang berada di dalamnya, karena setiap insiden secara sah lebih mendesak daripada pekerjaan pencegahan apa pun. Seorang engineer yang memilih mengerjakan perbaikan struktural sementara ada insiden yang menunggu sedang mengambil keputusan yang salah dalam kerangka jangka pendek, dan benar hanya dalam kerangka jangka panjang yang tidak ia kendalikan.

Pemutusan lingkaran harus datang dari keputusan alokasi di tingkat yang lebih tinggi: sejumlah kapasitas tim dilindungi dari pekerjaan reaktif, secara terjadwal dan tidak bisa diganggu kecuali oleh insiden tingkat satu.

Angka yang lazim berhasil adalah dua puluh persen — satu hari kerja dalam sepekan. Angka yang lebih kecil cenderung habis dimakan interupsi. Angka yang lebih besar sulit dipertahankan secara politis di sebagian besar organisasi.

Cara mengajukannya yang sering berhasil. Jangan mengajukannya sebagai permintaan waktu untuk perbaikan. Ajukan sebagai investasi dengan hasil yang diukur: dalam enam bulan, laporkan berapa banyak insiden yang tercegah, berapa waktu penanganan yang berkurang, dan berapa temuan yang ditutup. Bila angkanya bagus, program itu akan mempertahankan dirinya sendiri. Bila tidak, Anda telah belajar sesuatu yang layak diketahui.

Bab 161. Tentang Dokumentasi yang Sesungguhnya Dipakai

161.1 Mengapa sebagian besar dokumentasi tidak dibaca

Hampir setiap institusi punya dokumentasi. Hampir tidak ada yang punya dokumentasi yang dipakai.

Jarak antara keduanya jarang disebabkan oleh kemalasan pembaca. Ia disebabkan oleh ketidakcocokan antara bentuk dokumen dan situasi pemakaiannya.

Seseorang yang membaca dokumentasi pada pukul tiga pagi, di tengah insiden, dengan telepon berbunyi, punya kebutuhan yang sangat spesifik: langkah yang tepat, perintah yang bisa disalin, dan tidak ada apa pun lagi. Ia tidak membutuhkan latar belakang, tidak membutuhkan penjelasan arsitektur, dan tidak punya kesabaran untuk mencari di antara empat puluh halaman.

Seseorang yang mempelajari sistem pada Selasa siang punya kebutuhan yang berlawanan: konteks, alasan, hubungan antar bagian, dan contoh. Langkah tanpa penjelasan justru tidak berguna baginya karena ia tidak akan memahami kapan langkah itu berlaku.

Dokumen yang mencoba melayani keduanya gagal melayani salah satunya. Ini kesalahan yang sangat umum

dan sangat mudah diperbaiki: pisahkan.

161.2 Tiga dokumen yang berbeda untuk tiga kebutuhan

Kebutuhan	Bentuk	Ciri
Bertindak dalam darurat	Runbook	Langkah, perintah persis, tanpa penjelasan panjang
Memahami sistem	Dokumen arsitektur	Diagram, hubungan, alasan desain
Mengingat mengapa	Catatan keputusan	Konteks, pilihan yang ditimbang, konsekuensi

Ketiganya punya siklus hidup yang berbeda pula. Runbook diperbarui setiap kali dipakai. Dokumen arsitektur diperbarui saat arsitektur berubah. Catatan keputusan tidak pernah diperbarui — ia direkam sekali dan dibiarkan, karena nilainya justru pada kesetiaannya terhadap saat keputusan diambil.

161.3 Uji yang jujur untuk sebuah runbook

Saya telah menyebut uji ini beberapa kali dalam buku ini, dan mengulangnya di sini karena ia satu-satunya uji yang benar-benar bekerja.

Berikan runbook kepada seseorang yang belum pernah menangani masalah itu. Minta ia menjalankannya di lingkungan uji. Larang ia bertanya.

Setiap kali ia berhenti, ragu, atau melakukan sesuatu yang tidak Anda maksudkan, Anda telah menemukan kebocoran dalam dokumen. Perbaiki di titik itu, lalu ulangi dengan orang lain.

Uji ini terasa tidak nyaman bagi penulisnya, karena ia mengungkap bahwa dokumen yang terasa jelas bagi penulis ternyata penuh asumsi yang tidak tertulis. Itulah justru nilainya. Penulis runbook selalu menulis dari dalam kepalanya sendiri, dan yang ada di dalam kepalanya tidak ikut tertulis di atas kertas.

161.4 Catatan keputusan, dan mengapa ia jarang ditulis

Dari ketiga jenis dokumen, catatan keputusan adalah yang paling jarang ada dan yang paling sering dibutuhkan bertahun-tahun kemudian.

Bentuknya sederhana: konteks yang menuntut keputusan, pilihan yang dipertimbangkan beserta kelebihan dan kekurangannya, keputusan yang diambil, dan konsekuensi yang diterima. Satu halaman biasanya cukup.

Alasan ia jarang ditulis: pada saat keputusan diambil, alasannya terasa jelas bagi semua orang yang terlibat. Menuliskannya terasa seperti pekerjaan administratif yang tidak menambah apa-apa.

Tiga tahun kemudian, orang yang terlibat sudah pindah, konteksnya sudah berubah, dan yang tersisa hanyalah hasil keputusan tanpa alasannya. Penerus Anda akan melihat konfigurasi yang tampak aneh, menyimpulkan bahwa itu kecerobohan, dan mengubahnya — lalu menemukan alasan aslinya dengan cara yang mahal.

Contoh konkret dari buku ini. Narasi ketiga di Bab 156 dimulai dari perubahan control group bertahun-tahun sebelumnya, dari pola `*ALLUSR` menjadi daftar library eksplisit. Perubahan itu masuk akal pada saat dilakukan. Bila ada catatan keputusan yang menjelaskan alasannya, orang yang menyusun templat tiket

pembuatan library akan menemukannya dan menyadari bahwa templat perlu diperbarui.

Satu halaman yang tidak ditulis menghasilkan enam bulan data tanpa backup. Ini bukan hubungan sebab akibat yang langsung, dan tidak adil untuk menyalahkan siapa pun atasnya. Tetapi ia menggambarkan mengapa catatan keputusan bernilai jauh melampaui usaha yang dibutuhkan untuk menulisnya.

161.5 Dokumentasi sebagai bentuk kepedulian

Saya ingin menutup esai ini dengan sesuatu yang kurang teknis.

Dokumentasi yang baik adalah bentuk kepedulian terhadap orang yang akan datang setelah Anda. Ia dikerjakan untuk seseorang yang mungkin belum Anda kenal, yang akan menghadapi masalah pada malam ketika Anda sudah tidur atau sudah bekerja di tempat lain, dan yang keberhasilannya tidak akan pernah dikreditkan kepada Anda.

Ini sifat pekerjaan yang jarang dihargai dan mudah dilewati, karena tidak ada mekanisme yang memaksanya. Tidak ada yang akan menegur Anda karena tidak menulis catatan keputusan. Sistem akan tetap berjalan.

Tetapi di platform yang praktisinya semakin sedikit, di mana sistem berusia lebih lama daripada karier siapa pun yang mengelolanya, kesinambungan pengetahuan bukan kemewahan. Ia satu-satunya cara agar sistem yang Anda jaga masih bisa dijaga oleh orang lain setelah Anda pergi.

Seluruh buku ini, pada akhirnya, adalah upaya melakukan hal itu dalam skala yang lebih besar.

Bagian XLIX — Setahun dalam Hidup System Engineer

Bagian ini menyusun seluruh isi buku ke dalam urutan waktu. Alih-alih menyajikan pekerjaan menurut topik, ia menyajikannya menurut kapan pekerjaan itu muncul, sehingga pembaca yang baru memegang peran bisa melihat bentuk pekerjaan secara utuh sebelum tenggelam dalam rinciannya.

Susunan bulanannya mengikuti pola yang lazim di institusi keuangan Indonesia: tutup buku tahunan di awal tahun, audit di pertengahan, pemeliharaan besar di paruh kedua, dan persiapan tahun berikutnya menjelang akhir. Sesuaikan dengan siklus institusi Anda sendiri; yang penting adalah gagasan bahwa pekerjaan ini punya irama tahunan, dan mengetahui iramanya membuat Anda bisa merencanakan alih-alih sekadar bereaksi.

Bab 162. Kuartal Pertama

162.1 Januari — bayangan tutup buku tahunan

Bulan pertama tahun berjalan dimulai dengan sesuatu yang sebenarnya milik tahun sebelumnya. Proses tutup buku tahunan berjalan pada malam pergantian tahun, dan akibatnya masih terasa berpekan-pekan sesudahnya.

Yang khas muncul di Januari:

Pekerjaan	Sifat	Catatan
Verifikasi hasil tutup buku	Mendesak	Selisih ditemukan di sini, bukan saat

Pekerjaan	Sifat	Catatan
		prosesnya
Penyusunan laporan tahunan	Tenggat keras	Regulator dan audit eksternal
Pembersihan data tahun lalu	Dapat dijadwalkan	Sering ditunda dan menumpuk
Pemulihan kapasitas	Mendesak bila tertunda	Tutup buku meninggalkan banyak data sementara

Pelajaran yang sering diperoleh pada Januari. Kapasitas penyimpanan yang tampak memadai sepanjang tahun bisa mendekati batas persis pada pekan pertama Januari, karena tutup buku tahunan menghasilkan data kerja dalam jumlah yang tidak muncul pada bulan lain. Bila Anda baru memegang peran ini, periksa riwayat pemakaian ASP pada Januari tahun-tahun sebelumnya; angkanya akan memberi tahu apakah institusi Anda punya masalah ini.

Januari juga bulan ketika program End of Year dievaluasi. Program itu baru saja berjalan setelah dua belas bulan tidak disentuh, dan setiap masalah yang muncul masih segar. Ini jendela terbaik untuk memperbaikinya, dan jendela itu akan tertutup begitu perhatian berpindah ke pekerjaan lain. Saya sarankan menjadwalkan tinjauan EOY pada pekan kedua Januari, sebagai kegiatan tetap yang tidak bisa dipindahkan.

162.2 Februari — jendela yang tenang

Februari cenderung menjadi bulan paling tenang dalam kalender operasional. Tutup buku tahunan sudah selesai, audit belum dimulai, dan tidak ada tenggat besar.

Inilah alasan Februari adalah bulan terbaik untuk pekerjaan yang menuntut konsentrasi dan tidak boleh diburu: perbaikan struktural, pembangunan pemantauan, penulisan runbook, dan pemetaan sistem.

Yang sering terjadi sebaliknya: Februari dipakai untuk mengejar pekerjaan yang tertunda sejak akhir tahun. Itu pilihan yang bisa dipahami, tetapi ia membuang satu-satunya jendela tenang dalam setahun untuk pekerjaan yang sebenarnya bisa dilakukan kapan saja.

Saran yang mungkin terdengar berlawanan. Lindungi Februari. Tetapkan satu proyek perbaikan struktural pada bulan itu, umumkan kepada tim bahwa bulan itu akan dipakai untuk proyek tersebut, dan tolak pekerjaan non-mendesak yang masuk. Pekerjaan yang ditolak akan tetap ada di bulan Maret; jendela tenang tidak.

162.3 Maret — persiapan audit

Audit internal dan eksternal umumnya dimulai pada kuartal kedua, dan persiapannya jatuh di Maret.

Persiapan yang baik terlihat seperti ini:

1. Jalankan view kepatuhan dan tinjau setiap baris yang gagal
2. Untuk setiap temuan, tentukan: perbaiki sekarang, atau siapkan penjelasan
3. Kumpulkan bukti untuk pertanyaan audit yang berulang setiap tahun
4. Perbarui dokumen baseline konfigurasi
5. Pastikan berita acara uji restore dan uji DR tersedia

Persiapan yang buruk terlihat seperti ini: menunggu daftar pertanyaan auditor, lalu mencari jawabannya satu

per satu di bawah tekanan waktu.

Perbedaan antara keduanya tidak terletak pada kerja keras, melainkan pada waktu. Temuan yang ditemukan pada Maret bisa diperbaiki sebelum auditor datang. Temuan yang sama yang ditemukan pada Mei menjadi temuan resmi yang harus dilaporkan, ditindaklanjuti, dan dipantau penyelesaiannya — dengan biaya administratif yang jauh melebihi biaya perbaikannya sendiri.

Bagian XXXI dan Bab 27 memuat kerangka yang bisa dipakai. Kueri pemeriksaan di PR-01 pada Bab 151 adalah titik awal yang paling efisien.

Bab 163. Kuartal Kedua

163.1 April — audit berjalan

Selama audit berlangsung, sebagian besar waktu System Engineer habis untuk menyediakan data. Ini pekerjaan yang terasa tidak produktif, dan sebagiannya memang demikian. Tetapi ada dua hal yang bisa mengubahnya menjadi bernilai.

Pertama, setiap pertanyaan audit yang membutuhkan lebih dari sepuluh menit untuk dijawab adalah calon kueri tetap. Catat pertanyaannya, tulis kuerinya, simpan sebagai view di schema `TOOLS`. Tahun depan pertanyaan yang sama akan dijawab dalam sepuluh detik.

Kedua, temuan auditor sering menunjuk hal yang memang perlu diperbaiki. Reaksi yang lazim adalah defensif — mencari cara menjelaskan mengapa temuan itu tidak berlaku. Reaksi yang lebih berguna adalah menerima temuan yang valid dan memakainya sebagai dasar permintaan sumber daya. Temuan audit punya kekuatan politis yang tidak dimiliki permintaan biasa dari tim TI, dan memakainya bukan kecurangan melainkan pemanfaatan mekanisme yang tersedia.

163.2 Mei — musim Technology Refresh

IBM umumnya menerbitkan Technology Refresh pada paruh pertama tahun, dan Mei sering menjadi bulan ketika tim mulai menimbangnya.

Pertanyaan yang perlu dijawab sebelum memutuskan:

Pertanyaan	Sumber jawaban
Apa yang berubah	Catatan rilis IBM
Apakah ada perbaikan yang kita butuhkan	Daftar masalah terbuka kita sendiri
Apakah vendor aplikasi mendukung	Konfirmasi tertulis
Apakah ada masalah yang diketahui	Forum komunitas, IT Jungle
Berapa lama uji regresi dibutuhkan	Pengalaman sebelumnya

Godaan untuk menerapkan setiap TR karena ia tersedia layak ditahan. TR yang tidak membawa perbaikan yang Anda butuhkan hanya menambah risiko tanpa manfaat. Sebaliknya, menunda terlalu lama membuat lompatan berikutnya lebih besar dan lebih berisiko.

Pola yang wajar bagi sebagian besar institusi perbankan: terapkan satu TR per tahun, dijadwalkan, dengan uji regresi penuh. Terapkan di luar jadwal hanya bila ada perbaikan keamanan atau perbaikan atas masalah yang

sedang Anda alami.

163.3 Juni — tengah tahun dan tinjauan kapasitas

Paruh pertama tahun berakhir, dan ini titik alami untuk meninjau apakah proyeksi kapasitas yang dibuat pada awal tahun masih berlaku.

Yang ditinjau:

- Pertumbuhan penyimpanan aktual dibanding proyeksi
- Durasi EOD aktual dibanding baseline awal tahun
- Jumlah transaksi dibanding rencana bisnis
- Pemakaian CPU pada jam puncak

Bila salah satu menyimpang lebih dari dua puluh persen dari proyeksi, ini saatnya menyesuaikan rencana — bukan pada akhir tahun ketika siklus anggaran sudah berjalan.

Juni juga bulan yang tepat untuk mulai menyusun bahan anggaran tahun depan, karena siklus anggaran di sebagian besar institusi dimulai pada kuartal ketiga dan usulan yang disusun tergesa selalu terlihat seperti tebakan. Bab 115 memuat kerangkanya.

Bab 164. Kuartal Ketiga

164.1 Juli — jendela pemeliharaan besar

Banyak institusi memilih paruh kedua tahun untuk pekerjaan pemeliharaan besar: upgrade rilis, penggantian perangkat keras, reorganisasi tabel besar, atau penyeimbangan ASP.

Alasannya praktis. Tutup buku tahunan masih jauh, audit sudah selesai, dan bila ada masalah masih tersisa waktu untuk memulihkannya sebelum akhir tahun.

Prinsip yang layak dipegang: **pekerjaan besar tidak dilakukan di kuartal keempat.** Alasannya bukan takhayul, melainkan bahwa kuartal keempat adalah periode dengan beban transaksi tertinggi, tenggat pelaporan terpadat, dan jendela pemulihan tersempit bila terjadi masalah. Institusi yang menerapkan pembekuan perubahan menjelang akhir tahun melakukannya karena alasan yang baik.

164.2 Agustus — uji pemulihan bencana

Uji DR tahunan sering dijadwalkan pada kuartal ketiga karena ia menuntut jendela panjang dan ketersediaan banyak tim.

Bagian XL memuat prosedurnya secara rinci. Yang ingin saya tambahkan di sini adalah sesuatu tentang persiapan sosialnya, yang sering lebih menentukan keberhasilan daripada persiapan teknisnya.

Uji DR melibatkan tim yang dalam keseharian jarang bekerja bersama: infrastruktur, aplikasi, kanal digital, keamanan, operasi bisnis, dan kadang mitra eksternal. Masing-masing punya prioritas sendiri, dan uji DR bukan prioritas utama bagi sebagian besar mereka.

Konsekuensinya, sebagian besar kegagalan persiapan bukan kegagalan teknis melainkan kegagalan

koordinasi: tim yang tidak hadir, mitra yang tidak diberi tahu, persetujuan yang tidak diurus, atau daftar periksa yang disusun tanpa masukan dari pihak yang punya ketergantungan.

Saran praktis. Mulai koordinasi enam pekan sebelumnya, bukan dua. Minta setiap tim menyumbang butir verifikasinya sendiri. Dan sebutkan dengan jelas kepada setiap pihak apa yang dituntut dari mereka pada hari pelaksanaan, termasuk perkiraan durasinya.

164.3 September — penyelesaian temuan

Temuan dari audit, dari uji DR, dan dari pemeriksaan rutin menumpuk sepanjang tahun. September adalah bulan terakhir yang wajar untuk menyelesaikannya sebelum pembekuan perubahan akhir tahun.

Cara memprioritaskan yang saya sarankan:

1. Temuan yang menghilangkan kemampuan pemulihan — dikerjakan lebih dulu, tanpa pengecualian
2. Temuan keamanan dengan jalur eksploitasi yang nyata
3. Temuan yang berulang dari tahun sebelumnya — karena pengulangan menandakan perbaikan sebelumnya tidak berjalan
4. Temuan yang perbaikannya murah, apa pun tingkatnya
5. Sisanya, dijadwalkan ke tahun berikutnya secara eksplisit

Butir ketiga layak diberi perhatian khusus. Temuan yang muncul kembali setelah dinyatakan selesai berarti salah satu dari dua hal: perbaikannya tidak pernah benar-benar diterapkan, atau ia diterapkan tetapi tidak bertahan. Keduanya menunjukkan masalah proses yang lebih besar daripada temuan itu sendiri.

Bab 165. Kuartal Keempat

165.1 Oktober — persiapan tutup buku

Persiapan tutup buku tahunan yang baik dimulai pada Oktober, bukan Desember.

Yang dikerjakan:

Pekerjaan	Alasan dilakukan sedini ini
Uji program EOY di UAT dengan data terkini	Program tidak berjalan dua belas bulan
Verifikasi kapasitas mencukupi untuk lonjakan	Tutup buku menghasilkan data kerja besar
Konfirmasi jendela dengan bisnis	Jadwal akhir tahun padat
Siapkan prosedur khusus dan personel jaga	Malam pergantian tahun
Pastikan backup dan DR siap	Risiko tertinggi dalam setahun

Uji EOY di UAT adalah yang paling penting dan paling sering dilewati. Program itu tidak berjalan sejak Januari, sementara di sekelilingnya ada perubahan PTF, perubahan aplikasi, perubahan struktur tabel, dan pertumbuhan data. Asumsi bahwa ia akan berjalan seperti tahun lalu tidak punya dasar.

165.2 November — pembekuan dan pemantapan

Sebagian besar institusi keuangan memberlakukan pembekuan perubahan mulai pertengahan November atau awal Desember sampai pekan kedua Januari.

Periode pembekuan sering dianggap sebagai periode menganggur bagi tim teknis. Ia tidak perlu demikian. Pekerjaan yang bisa dilakukan tanpa mengubah produksi cukup banyak:

- Dokumentasi yang tertunda sepanjang tahun
- Penulisan dan pengujian runbook di lingkungan uji
- Analisis tren kapasitas untuk usulan tahun depan
- Pelatihan dan transfer pengetahuan
- Persiapan proyek tahun berikutnya
- Peninjauan kode untuk perubahan yang akan diterapkan Januari

Ada satu hal yang sebaiknya dikerjakan setiap November dan hampir tidak pernah dikerjakan: memperbarui dokumen pemulihan darurat yang dicetak. Konfigurasi berubah sepanjang tahun, dan dokumen yang dicetak Januari lalu kemungkinan sudah tidak akurat. November adalah waktu yang tepat karena periode risiko tertinggi justru akan datang beberapa pekan lagi.

165.3 Desember — malam yang paling diperhatikan

Tutup buku tahunan adalah proses batch paling berisiko dalam setahun, karena tiga alasan yang menumpuk.

Pertama, volumenya paling besar. Perhitungan tahunan menyentuh seluruh rekening, seluruh kredit, dan seluruh akun buku besar sekaligus.

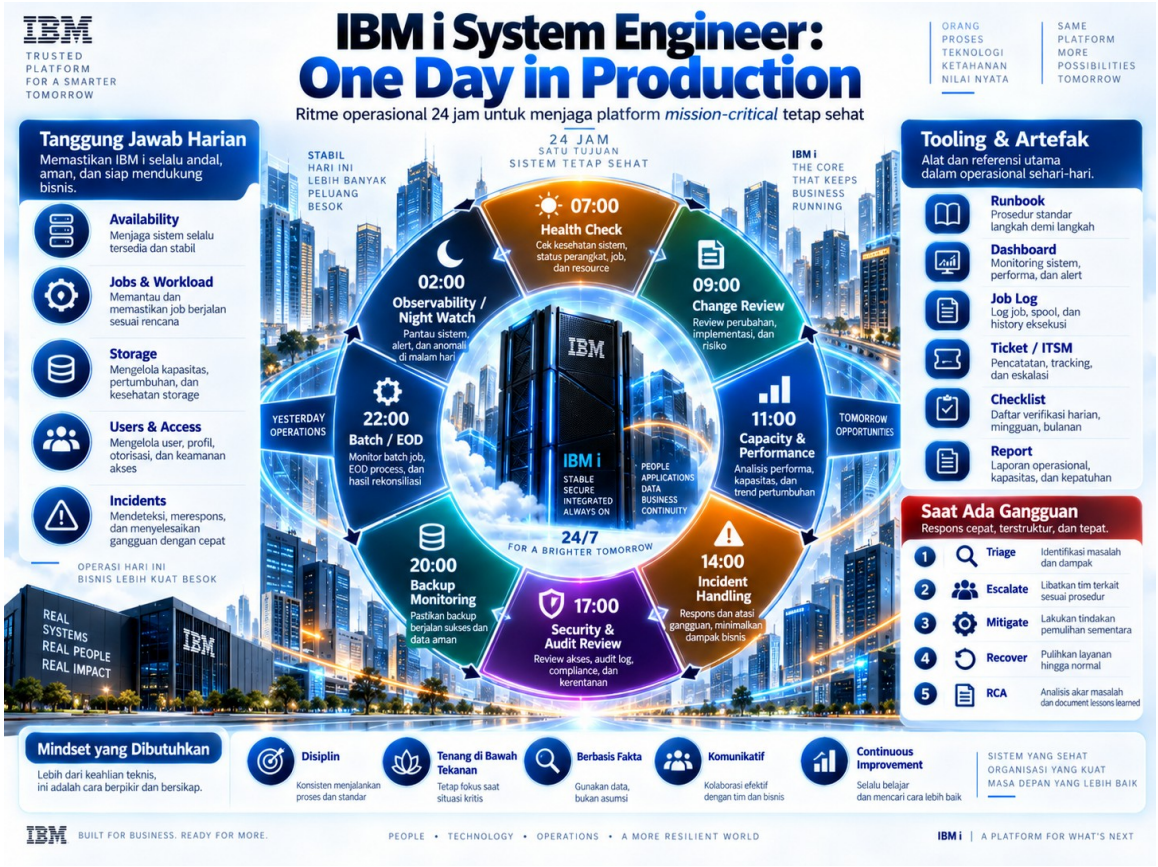
Kedua, programnya paling jarang diuji. Ia berjalan sekali setahun.

Ketiga, jendela pemulihannya paling sempit. Layanan harus buka pada hari kerja pertama tahun baru, dan tidak ada ruang untuk menunda.

Kombinasi ketiganya berarti persiapan yang untuk EOD harian terasa berlebihan menjadi wajar untuk EOY: personel jaga penuh, manajer siaga, prosedur tertulis yang sudah dibaca bersama, angka kontrol yang dicatat sebelum dan sesudah, dan batas waktu keputusan yang disepakati di muka.

Saya menyarankan satu kebiasaan sederhana yang nilainya jauh melampaui usahanya: **tulis catatan kronologis selama malam itu berlangsung**, bukan setelahnya. Setiap tahap, waktunya, dan angka yang diverifikasi. Bila semuanya berjalan lancar, catatan itu menjadi dasar perencanaan tahun depan. Bila terjadi masalah, ia menjadi satu-satunya rekaman yang akurat tentang apa yang terjadi.

Bab 166. Irama Mingguan dan Harian



Gambar 112 — Ritme operasional 24 jam seorang System Engineer

166.1 Bentuk sebuah pekan

Di luar siklus tahunan, ada irama mingguan yang layak dibangun secara sadar.

Hari	Cocok untuk	Alasan
Senin	Tinjauan pekan lalu, perencanaan	Konteks masih segar dari laporan akhir pekan
Selasa dan Rabu	Perubahan terkendali	Cukup waktu memulihkan sebelum akhir pekan
Kamis	Verifikasi perubahan, dokumentasi	Perubahan sudah berjalan sehari
Jumat	Pekerjaan yang tidak menyentuh produksi	Jangan meninggalkan perubahan baru di akhir pekan
Akhir pekan	Pemeliharaan besar terjadwal	Beban rendah, jendela panjang

Aturan tidak menerapkan perubahan pada Jumat bukan takhayul. Ia berdasarkan kenyataan bahwa masalah yang muncul dari sebuah perubahan sering baru terlihat setelah satu siklus penuh, dan siklus penuh dari perubahan Jumat sore jatuh pada Sabtu — ketika tim yang memahami perubahan itu tidak bekerja.

166.2 Bentuk sebuah hari

Hari kerja seorang System Engineer di lingkungan core banking punya bentuk yang cukup konsisten.

Pagi, sebelum jam layanan. Pemeriksaan pagi adalah ritual yang tidak bisa ditawar. Daftar periksanya ada di

Lampiran D, dan yang terpenting di antaranya adalah membandingkan tanggal buku aplikasi dengan tanggal sistem — pemeriksaan sepuluh detik yang langsung memberi tahu apakah EOD semalam selesai.

Pagi sampai siang. Ini jendela dengan konsentrasi terbaik, dan di sebagian besar tim ia habis untuk rapat dan permintaan yang masuk. Melindungi sebagian darinya untuk pekerjaan yang menuntut pemikiran adalah keputusan yang harus diambil sadar, karena ia tidak akan terjadi sendiri.

Siang sampai sore. Jam dengan interupsi tertinggi. Cocok untuk pekerjaan yang bisa dipotong dan dilanjutkan: tinjauan, dokumentasi, penanganan tiket.

Menjelang sore. Persiapan malam: verifikasi tidak ada perubahan yang belum selesai, profil memori malam terjadwal, media backup tersedia, personel jaga terkonfirmasi.

Malam. Bagi yang berjaga, malam adalah periode menunggu yang diselingi momen menuntut. Keterampilan yang dibutuhkan bukan kecepatan mengetik, melainkan kemampuan mengambil keputusan yang benar dalam keadaan lelah — dan itu dilatih dengan menyiapkan kerangka keputusan sebelumnya, bukan dengan berusaha lebih keras pada saat itu.

166.3 Menyusun ulang pekerjaan Anda sendiri

Bila Anda merasa pekerjaan Anda didominasi reaksi terhadap masalah, latihan berikut sering membuka mata.

Selama dua pekan, catat setiap pekerjaan yang Anda lakukan beserta durasinya, dan beri label salah satu dari empat kategori:

Kategori	Contoh
Reaktif	Menangani insiden, menjawab permintaan mendadak
Rutin	Pemeriksaan pagi, backup, laporan berkala
Proyek	Upgrade, migrasi, pembangunan sistem baru
Pencegahan	Pemantauan baru, runbook, dokumentasi, latihan

Setelah dua pekan, hitung proporsinya. Angka yang saya lihat paling sering di lapangan: reaktif empat puluh sampai enam puluh persen, rutin dua puluh sampai tiga puluh persen, proyek sepuluh sampai dua puluh persen, dan pencegahan di bawah lima persen.

Kategori keempat adalah satu-satunya yang mengurangi kategori pertama. Selama ia di bawah lima persen, proporsi reaktif tidak akan turun, dan tahun depan hasil pengukuran yang sama akan menunjukkan angka yang sama.

Inilah alasan mengapa Bab 160.5 membahas perlindungan waktu sebagai keputusan alokasi, bukan sebagai soal disiplin pribadi. Seorang engineer tidak bisa memilih untuk mengabaikan insiden demi mengerjakan pencegahan; keputusan itu harus dibuat di tingkat yang bisa menetapkan prioritas.

💻 Bagian L — Pendalaman Pemrograman CL

Bab 10A memberi primer CL yang cukup untuk menulis utilitas operasional sederhana. Bagian ini melanjutkannya ke tingkat yang dibutuhkan ketika Anda mulai membangun perkakas yang dipakai orang lain:

perintah buatan sendiri, pemanggilan API sistem, dan pustaka utilitas yang bisa dipelihara.

Ada alasan mengapa pendalaman ini layak diberi ruang tersendiri. CL adalah satu-satunya bahasa di IBM i yang dirancang khusus untuk pekerjaan System Engineer. RPG untuk aplikasi, SQL untuk data, Python untuk otomasi lintas sistem — tetapi ketika Anda perlu mengendalikan subsystem, menangani pesan sistem, mengatur job, atau membungkus prosedur operasional agar bisa dijalankan orang lain dengan aman, CL adalah alat yang paling tepat dan paling tidak dihargai.

Bab 167. Membuat Perintah Sendiri

167.1 Mengapa membuat perintah, bukan sekadar program

Sebuah program CL bisa dipanggil dengan `CALL PGM(APPPGM/PROSESEOD) PARM('20260918' 'AKRUAL')`. Itu berfungsi, tetapi punya beberapa kelemahan yang baru terasa ketika orang lain yang memakainya.

Pemanggil harus mengingat urutan parameter. Tidak ada validasi bahwa nilai yang diberikan masuk akal. Tidak ada bantuan yang muncul bila ia menekan F4. Tidak ada nilai bawaan. Dan tidak ada cara membatasi siapa yang boleh menjalankannya selain membatasi wewenang atas programnya.

Perintah buatan sendiri menyelesaikan semuanya sekaligus. Ia memberi antarmuka yang sama dengan perintah bawaan sistem: prompt dengan F4, validasi nilai, nilai bawaan, teks bantuan, dan wewenang objek tersendiri yang terpisah dari program yang dijalankannya.

Ini pembedaan yang penting bagi tata kelola. Anda bisa memberi seorang operator wewenang menjalankan perintah `RUNEOD` tanpa memberinya wewenang apa pun atas program, tabel, atau objek yang disentuh program itu — karena program berjalan dengan wewenang pemiliknya lewat mekanisme adopsi.

167.2 Anatomi definisi perintah

Definisi perintah ditulis dalam bahasa tersendiri, disimpan sebagai sumber, lalu dikompilasi dengan `CRTCMD`.

```
            CMD        PROMPT('Jalankan fase End of Day')

            PARM       KWD(FASE) TYPE(*CHAR) LEN(10) MIN(1) +
                         RSTD(*YES) +
                         VALUES(VALIDASI AKRUAL POSTING +
                                KLASIFIKASI GLBALANCE LAPORAN) +
                         PROMPT('Fase yang dijalankan')

            PARM       KWD(TGLBUKU) TYPE(*DATE) MIN(0) +
                         DFT(*CURRENT) +
                         SPCVAL((*CURRENT '0000000')) +
                         PROMPT('Tanggal buku')

            PARM       KWD(MODE) TYPE(*CHAR) LEN(6) MIN(0) +
                         RSTD(*YES) DFT(*NORMAL) +
                         VALUES(*NORMAL *SIMUL *ULANG) +
                         PROMPT('Mode eksekusi')

            PARM       KWD(JMLJOB) TYPE(*DEC) LEN(2 0) MIN(0) +
                         DFT(4) RANGE(1 8) +
```

```
                          PROMPT('Jumlah job paralel')

            PARM       KWD(TIKET) TYPE(*CHAR) LEN(15) MIN(0) +
                          PROMPT('Nomor tiket perubahan')
```

167.3 Kata kunci yang menentukan perilaku

Kata kunci	Fungsi	Kapan dipakai
`MIN(1)`	Parameter wajib diisi	Nilai yang tidak punya bawaan aman
`MIN(0)`	Parameter opsional	Selalu sertakan `DFT`
`DFT`	Nilai bawaan	Nilai yang paling sering benar
`RSTD(*YES)`	Hanya nilai dalam `VALUES` yang sah	Mencegah salah ketik
`VALUES`	Daftar nilai yang diizinkan	Fase, mode, kode
`RANGE`	Rentang numerik yang sah	Jumlah job, batas waktu
`SPCVAL`	Nilai khusus yang diterjemahkan	`*CURRENT`, `*ALL`
`SNGVAL`	Nilai tunggal yang mengecualikan yang lain	`*ALL` pada daftar
`CHOICE`	Teks petunjuk di layar prompt	Membantu pengguna
`PMTCTL`	Tampilkan parameter hanya bila kondisi tertentu	Menyembunyikan yang tidak relevan
`CASE(*MIXED)`	Pertahankan huruf besar kecil	Kata sandi, jalur IFS
`EXPR(*YES)`	Izinkan variabel CL sebagai nilai	Dipanggil dari program lain

Tentang `RSTD(*YES)` dan `VALUES`. Ini kombinasi yang paling banyak memberi manfaat dengan usaha paling kecil. Tanpanya, seorang operator yang mengetik `AKRUL` alih-alih `AKRUAL` akan menjalankan program dengan nilai yang tidak dikenali, dan apa yang terjadi selanjutnya bergantung pada seberapa baik program itu menangani nilai tak terduga. Dengannya, sistem menolak sebelum program dipanggil.

167.4 Kompilasi dan wewenang

```
/* Kompilasi definisi perintah */
CRTCMD CMD(APPPGM/RUNEOD) +
       PGM(APPPGM/EODCTL) +
       SRCSTMF('/src/cmd/runeod.cmd') +
       VLDCKR(APPPGM/EODVLD) +
       ALLOW(*IPGM *IREXX *BPGM *BREXX *INTERACT *BATCH) +
       MAXPOS(*NOMAX) +
       TEXT('Jalankan satu fase End of Day')

/* Wewenang: operator boleh menjalankan perintah, bukan programnya */
GRTOBJAUT OBJ(APPPGM/RUNEOD) OBJTYPE(*CMD) +
          USER(GRPEODOPS) AUT(*USE)

GRTOBJAUT OBJ(APPPGM/EODCTL) OBJTYPE(*PGM) +
          USER(*PUBLIC) AUT(*EXCLUDE)

/* Program berjalan dengan wewenang pemiliknya */
CHGPGM PGM(APPPGM/EODCTL) USRPRF(*OWNER) USEADPAUT(*NO)
```

`USEADPAUT(*NO)` penting dan sering dilewatkan. Ia mencegah program ini mewarisi wewenang adopsi dari program yang memanggilnya, sehingga rantai adopsi tidak menumpuk tanpa disadari.

Parameter `ALLOW` menentukan dari mana perintah boleh dijalankan. Untuk perintah yang hanya boleh

dipakai dalam job batch terjadwal, batasi ke `*BPGM` dan `*BATCH` saja — ini mencegah seseorang menjalankannya dari sesi interaktif secara tidak sengaja.

167.5 Program validasi

Validasi yang tidak bisa dinyatakan dengan `VALUES` atau `RANGE` ditulis sebagai program tersendiri yang dipanggil sistem sebelum perintah dijalankan.

```
PGM PARM(&CMDSTR)
   DCL VAR(&CMDSTR)  TYPE(*CHAR) LEN(3000)
   DCL VAR(&FASE)    TYPE(*CHAR) LEN(10)
   DCL VAR(&MODE)    TYPE(*CHAR) LEN(6)
   DCL VAR(&TIKET)   TYPE(*CHAR) LEN(15)
   DCL VAR(&SYSNAME) TYPE(*CHAR) LEN(8)
   DCL VAR(&JML)     TYPE(*DEC)  LEN(3 0)

   /* Uraikan parameter dari string perintah; di praktik nyata
      dipakai API pengurai, disederhanakan di sini */

   RTVNETA SYSNAME(&SYSNAME)

   /* Aturan 1: mode ULANG hanya boleh dengan nomor tiket */
   IF COND(&MODE *EQ '*ULANG' *AND &TIKET *EQ ' ') THEN(DO)
      SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
                MSGDTA('Mode *ULANG menuntut nomor tiket perubahan') +
                MSGTYPE(*DIAG)
      SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
   ENDDO

   /* Aturan 2: fase AKRUAL tidak boleh dijalankan di produksi
      tanpa persetujuan yang tercatat */
   IF COND(&SYSNAME *EQ 'PRDIBMI' *AND &FASE *EQ 'AKRUAL' +
           *AND &MODE *EQ '*ULANG') THEN(DO)
      RUNSQL SQL('CREATE TABLE QTEMP/CEKAPPR AS ( +
         SELECT COUNT(*) AS JML FROM APPWRK.PERSETUJUAN +
          WHERE NO_TIKET = ''' *CAT &TIKET *TCAT ''' +
            AND JENIS = ''RERUN_AKRUAL'' +
            AND JML_PENYETUJU >= 2) WITH DATA') COMMIT(*NONE)

      /* baca hasil; bila kurang dari dua penyetuju, tolak */
      IF COND(&JML *LT 2) THEN(DO)
         SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
                   MSGDTA('Rerun akrual menuntut persetujuan dua orang') +
                   MSGTYPE(*DIAG)
         SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
      ENDDO
   ENDDO
ENDPGM
```

Inilah tempat yang tepat untuk menegakkan aturan dua orang yang dibahas berulang kali di buku ini. Aturan yang hanya tertulis di prosedur bergantung pada kepatuhan; aturan yang ditegakkan oleh program validasi perintah tidak bisa dilewati pada pukul tiga pagi.

167.6 Teks bantuan

Perintah tanpa bantuan akan ditanyakan berulang kali kepada Anda. Bantuan ditulis sebagai sumber panel group.

```
CRTPNLGRP PNLGRP(APPPGM/RUNEODH) +
          SRCSTMF('/src/help/runeodh.pnlgrp')

CHGCMD CMD(APPPGM/RUNEOD) HLPPNLGRP(APPPGM/RUNEODH) +
       HLPID(RUNEOD)
```

Isi minimum yang layak ada dalam bantuan sebuah perintah operasional:

1. Apa yang dilakukan perintah ini, dalam satu kalimat
2. Apa yang terjadi bila gagal di tengah
3. Apakah aman dijalankan ulang
4. Siapa yang harus dihubungi bila ragu
5. Rujukan ke runbook yang relevan

Butir kedua dan ketiga jarang ada dan paling dibutuhkan. Seseorang yang membuka bantuan pada tengah malam bukan ingin tahu sintaksnya; ia ingin tahu apakah aman menekan Enter.

167.7 Mengubah perintah bawaan sistem

IBM i memungkinkan penggantian perintah sistem dengan versi Anda sendiri lewat `CRTPRXCMD` atau lewat pembuatan perintah bernama sama di library yang mendahului QSYS dalam library list.

Ini teknik yang kuat dan berbahaya. Ia berguna untuk menambahkan pencatatan atau pembatasan pada perintah berbahaya, tetapi ia juga berarti perintah yang dijalankan seseorang bukan perintah yang ia kira.

Bila Anda memakainya, dokumentasikan dengan sangat jelas. Seorang penerus yang menemukan bahwa `ENDSBS` di sistem Anda berperilaku berbeda dari dokumentasi IBM, tanpa menemukan catatan mengapa, akan menghabiskan waktu yang sangat panjang untuk memahaminya.

```
-- Menemukan perintah buatan sendiri yang menimpa perintah sistem
SELECT OBJNAME, OBJLONGSCHEMA, OBJTEXT, OBJCREATED
  FROM TABLE(QSYS2.OBJECT_STATISTICS('*ALLUSR', '*CMD'))
 WHERE OBJNAME IN (
       SELECT OBJNAME FROM TABLE(QSYS2.OBJECT_STATISTICS('QSYS','*CMD')))
 ORDER BY OBJNAME;
```

Kueri ini layak dijalankan sekali di lingkungan Anda. Hasilnya kadang mengejutkan.

🚀 Bab 168. Memanggil API Sistem dari CL

168.1 Kapan API dibutuhkan

Sebagian besar informasi sistem kini tersedia lewat IBM i Services, dan bila SQL bisa menjawab, SQL adalah pilihan yang lebih baik: lebih mudah dibaca, lebih mudah diuji, dan tidak menuntut penanganan penunjuk.

API tetap dibutuhkan dalam tiga situasi:

1. Informasi yang belum tersedia sebagai IBM i Service
2. Operasi yang harus dilakukan, bukan sekadar dibaca — misalnya mengirim pesan ke antrean program tertentu
3. Kinerja pada pemanggilan yang sangat sering, di mana biaya membuka kursor SQL tidak dapat diterima

168.2 Pola pemanggilan API

Hampir seluruh API IBM i mengikuti pola yang sama: Anda menyediakan ruang penerima, memberi tahu ukurannya, menyebut format yang diinginkan, dan menyediakan struktur kesalahan.

```
PGM
   DCL VAR(&RCVVAR)  TYPE(*CHAR) LEN(400)
   DCL VAR(&RCVLEN)  TYPE(*CHAR) LEN(4)
   DCL VAR(&FORMAT)  TYPE(*CHAR) LEN(8)  VALUE('JOBI0200')
   DCL VAR(&QJOB)    TYPE(*CHAR) LEN(26) VALUE('*                         ')
   DCL VAR(&INTJOB)  TYPE(*CHAR) LEN(16)
   DCL VAR(&ERRCOD)  TYPE(*CHAR) LEN(8)
   DCL VAR(&BIN4)    TYPE(*CHAR) LEN(4)
   DCL VAR(&JMLJOB)  TYPE(*DEC)  LEN(9 0)

   /* Panjang ruang penerima sebagai bilangan bulat empat byte */
   CHGVAR VAR(%BIN(&RCVLEN)) VALUE(400)

   /* Struktur kesalahan: nol berarti kesalahan dikembalikan
      sebagai pesan escape, bukan lewat struktur */
   CHGVAR VAR(%BIN(&ERRCOD 1 4)) VALUE(0)

   CALL PGM(QUSRJOBI) PARM(&RCVVAR &RCVLEN &FORMAT +
                           &QJOB &INTJOB &ERRCOD)
   MONMSG MSGID(CPF0000) EXEC(DO)
      SNDPGMMSG MSG('Gagal mengambil informasi job') MSGTYPE(*ESCAPE)
   ENDDO

   /* Ambil nilai dari posisi tertentu dalam ruang penerima */
   CHGVAR VAR(&BIN4) VALUE(%SST(&RCVVAR 61 4))
   CHGVAR VAR(&JMLJOB) VALUE(%BIN(&BIN4))
ENDPGM
```

Tiga hal yang selalu menjadi sumber kesalahan pada pemanggilan API dari CL.

Pertama, ukuran ruang penerima harus dinyatakan sebagai bilangan bulat empat byte, bukan sebagai angka desimal. Fungsi `%BIN` melakukan konversi itu, dan melupakannya menghasilkan hasil yang tampak acak.

Kedua, posisi field dalam ruang penerima dihitung dari dokumentasi API, dan dokumentasinya memakai penomoran mulai dari nol pada sebagian format dan dari satu pada sebagian lain. Selalu verifikasi dengan menampilkan isi heksa ruang penerima saat pertama kali membangun pemanggilan.

Ketiga, struktur kesalahan yang diisi nol berarti API akan mengirim pesan escape bila gagal. Bila Anda mengisinya dengan panjang struktur, API akan mengembalikan kode kesalahan di dalam struktur dan tidak mengirim pesan — dan program yang tidak memeriksa struktur itu akan melanjutkan seolah semuanya berhasil.

168.3 API yang paling berguna untuk pekerjaan operasional

API	Fungsi
`QUSRJOBI`	Informasi job
`QUSLJOB`	Daftar job ke user space
`QUSLOBJ`	Daftar objek ke user space
`QUSROBJD`	Deskripsi satu objek

API	Fungsi
QUSCRTUS	Buat user space
QUSRTVUS	Baca isi user space
QMHSNDPM	Kirim pesan program
QMHRCVPM	Terima pesan program
QMHRMVPM	Hapus pesan dari antrean
QSNDDTAQ / QRCVDTAQ	Kirim dan terima data queue
QCMDEXC	Jalankan perintah dari string
QCAPCMD	Jalankan perintah dengan pengurai
QSYRUSRI	Informasi profil pengguna
QWCRSVAL	Ambil system value
QEZSNDMG	Kirim pesan ke pengguna

168.4 Pola pembersihan job log

Salah satu pemakaian API yang paling praktis: menghapus pesan yang tidak perlu dari job log agar pesan penting tidak tenggelam.

```
PGM
   DCL VAR(&MSGKEY) TYPE(*CHAR) LEN(4)
   DCL VAR(&ERRCOD) TYPE(*CHAR) LEN(8)

   CHGVAR VAR(%BIN(&ERRCOD 1 4)) VALUE(0)

   /* Hapus seluruh pesan informasi dari antrean program ini,
      sisakan yang bertingkat diagnostik ke atas */
   CALL PGM(QMHRMVPM) PARM('*'          +
                           x'00000000'  +
                           &MSGKEY      +
                           '*INFO    '  +
                           &ERRCOD)
   MONMSG MSGID(CPF0000)
ENDPGM
```

Pemakaiannya: pada program yang menjalankan ratusan perintah dalam perulangan, job log terisi ribuan pesan penyelesaian yang tidak berguna, dan pesan kesalahan yang sesungguhnya menjadi sulit ditemukan. Membersihkan pesan informasi secara berkala membuat job log tetap dapat dibaca.

Peringatan. Jangan menghapus pesan diagnostik atau escape. Keduanya adalah bukti yang dibutuhkan saat penyelidikan, dan menghapusnya demi kerapian adalah kerugian bersih.

168.5 Menjalankan perintah dari string

```
PGM
   DCL VAR(&PERINTAH) TYPE(*CHAR) LEN(500)
   DCL VAR(&PANJANG)  TYPE(*DEC)  LEN(15 5)
   DCL VAR(&LIB)      TYPE(*CHAR) LEN(10) VALUE('DTALIB')

   CHGVAR VAR(&PERINTAH) +
          VALUE('SAVLIB LIB(' *CAT &LIB *TCAT ') DEV(*SAVF) +
                 SAVF(BACKUP/HARIAN) CLEAR(*ALL)')
   CHGVAR VAR(&PANJANG) VALUE(500)

   CALL PGM(QCMDEXC) PARM(&PERINTAH &PANJANG)
   MONMSG MSGID(CPF0000) EXEC(DO)
      SNDPGMMSG MSG('Perintah gagal: ' *CAT %SST(&PERINTAH 1 100)) +
                MSGTYPE(*ESCAPE)
   ENDDO
ENDPGM
```

Ini teknik yang memungkinkan perintah dibangun secara dinamis dari data — misalnya menjalankan `SAVLIB` untuk setiap library yang terdaftar di tabel konfigurasi.

Risiko yang perlu disadari. Perintah yang dibangun dari data yang berasal dari luar adalah jalur yang bisa disalahgunakan. Bila string perintah menyertakan nilai dari tabel yang bisa diubah pengguna, seseorang bisa menyisipkan perintah lain. Validasi nilainya sebelum menyusun string, terutama bila nilainya berasal dari luar kendali Anda.

Bab 169. Membangun Pustaka Utilitas CL

169.1 Mengapa pustaka, bukan kumpulan program lepas

Setelah beberapa tahun, sebagian besar institusi punya puluhan program CL utilitas yang ditulis oleh orang berbeda pada waktu berbeda. Masing-masing menyelesaikan masalahnya sendiri, dan tidak ada yang memakai ulang apa pun dari yang lain.

Akibatnya terlihat pada hal-hal kecil: dua puluh cara berbeda menulis pesan ke log, lima belas cara berbeda menangani kesalahan, dan tidak ada satu pun yang bisa diubah di satu tempat.

Pustaka utilitas bersama menyelesaikan ini, dan biayanya rendah karena CL mendukung pemanggilan program dengan mudah.

169.2 Utilitas yang layak ada di setiap institusi

Utilitas	Fungsi
`UTLLOG`	Tulis satu baris ke tabel log terpusat
`UTLALERT`	Kirim peringatan lewat jalur yang disepakati
`UTLLOCK`	Ambil kunci eksklusif dengan batas waktu
`UTLUNLOCK`	Lepaskan kunci
`UTLCEKENV`	Verifikasi lingkungan sebelum bertindak
`UTLTUNGGU`	Tunggu kondisi dengan batas waktu
`UTLPARAM`	Ambil nilai parameter dari tabel konfigurasi

Utilitas	Fungsi
`UTLJOBSTS`	Periksa status job lain
`UTLKIRIMBRK`	Kirim pesan interupsi terkendali

169.3 Contoh: utilitas pencatatan terpusat

```
PGM PARM(&PROGRAM &TINGKAT &PESAN)
   DCL VAR(&PROGRAM) TYPE(*CHAR) LEN(10)
   DCL VAR(&TINGKAT) TYPE(*CHAR) LEN(1)   /* I D W E */
   DCL VAR(&PESAN)   TYPE(*CHAR) LEN(500)
   DCL VAR(&JOBNAME) TYPE(*CHAR) LEN(10)
   DCL VAR(&JOBUSER) TYPE(*CHAR) LEN(10)
   DCL VAR(&JOBNBR)  TYPE(*CHAR) LEN(6)
   DCL VAR(&SQLSTR)  TYPE(*CHAR) LEN(1500)

   RTVJOBA JOB(&JOBNAME) USER(&JOBUSER) NBR(&JOBNBR)

   CHGVAR VAR(&SQLSTR) VALUE( +
      'INSERT INTO APPWRK.LOG_APLIKASI +
       (WAKTU, PROGRAM, TINGKAT, JOB_NAME, JOB_USER, JOB_NBR, PESAN) +
       VALUES(CURRENT TIMESTAMP, ''' *CAT &PROGRAM *TCAT ''', +
              ''' *CAT &TINGKAT *TCAT ''', +
              ''' *CAT &JOBNAME *TCAT ''', +
              ''' *CAT &JOBUSER *TCAT ''', +
              ''' *CAT &JOBNBR  *TCAT ''', +
              ''' *CAT %SST(&PESAN 1 400) *TCAT ''')')

   RUNSQL SQL(&SQLSTR) COMMIT(*NONE)
   MONMSG MSGID(CPF0000) EXEC(DO)
      /* Pencatatan tidak boleh menjatuhkan pemanggilnya */
      SNDPGMMSG MSG('Peringatan: pencatatan gagal') MSGTYPE(*INFO)
   ENDDO
ENDPGM
```

Baris `MONMSG` di akhir adalah keputusan desain yang layak dijelaskan. Sebuah utilitas pencatatan yang gagal tidak boleh menghentikan proses yang sedang dicatatnya. Bila tabel log penuh atau terkunci, proses bisnis harus tetap berjalan; kehilangan satu baris log jauh lebih murah daripada menghentikan EOD.

Ini berlawanan dengan prinsip gagal dalam keadaan aman yang dibahas di Bab 140.2, dan pengecualiannya disengaja. Prinsip itu berlaku untuk pemeriksaan yang menentukan apakah proses boleh berjalan. Pencatatan bukan pemeriksaan; ia pengamatan, dan pengamatan yang gagal tidak boleh menghentikan yang diamati.

169.4 Contoh: utilitas penguncian dengan batas waktu

```
PGM PARM(&NAMAKUNCI &BATASDETIK &HASIL)
   DCL VAR(&NAMAKUNCI)  TYPE(*CHAR) LEN(10)
   DCL VAR(&BATASDETIK) TYPE(*DEC)  LEN(5 0)
   DCL VAR(&HASIL)      TYPE(*CHAR) LEN(1)
   DCL VAR(&TERPAKAI)   TYPE(*DEC)  LEN(5 0) VALUE(0)

   CHGVAR VAR(&HASIL) VALUE('N')

   /* Pastikan objek kunci ada */
   CHKOBJ OBJ(APPWRK/&NAMAKUNCI) OBJTYPE(*DTAARA)
   MONMSG MSGID(CPF9801) EXEC(DO)
      CRTDTAARA DTAARA(APPWRK/&NAMAKUNCI) TYPE(*CHAR) LEN(1)
      MONMSG MSGID(CPF1023)
```

```
   ENDDO

 COBA:
   ALCOBJ OBJ((APPWRK/&NAMAKUNCI *DTAARA *EXCL)) WAIT(5)
   MONMSG MSGID(CPF1002) EXEC(DO)
      CHGVAR VAR(&TERPAKAI) VALUE(&TERPAKAI + 5)
      IF COND(&TERPAKAI *GE &BATASDETIK) THEN(DO)
         CHGVAR VAR(&HASIL) VALUE('T')   /* timeout */
         RETURN
      ENDDO
      GOTO CMDLBL(COBA)
   ENDDO

   CHGVAR VAR(&HASIL) VALUE('Y')
ENDPGM
```

Pemanggilnya menjadi ringkas dan seragam:

```
CALL PGM(UTILLIB/UTLLOCK) PARM('EODLOCK' 300 &HASIL)

IF COND(&HASIL *NE 'Y') THEN(DO)
   CALL PGM(UTILLIB/UTLLOG) PARM('EODCTL' 'E' +
           'Tidak bisa mengambil kunci EOD dalam 300 detik')
   SNDPGMMSG MSG('Proses lain sedang berjalan') MSGTYPE(*ESCAPE)
ENDDO
```

169.5 Menguji program CL

CL jarang diuji secara sistematis, dan itu salah satu alasan mengapa program CL operasional sering menyimpan kesalahan yang baru muncul dalam keadaan tertentu.

Pengujian yang praktis untuk CL:

Jenis	Cara
Jalur normal	Jalankan dengan parameter wajar, verifikasi hasilnya
Parameter tidak sah	Jalankan dengan nilai di luar batas
Objek tidak ada	Hapus objek yang dibutuhkan, jalankan
Objek terkunci	Kunci dari sesi lain, jalankan
Kehabisan waktu	Buat kondisi tunggu yang tidak pernah terpenuhi
Lingkungan salah	Jalankan di sistem yang salah, pastikan ditolak
Dijalankan dua kali	Jalankan berurutan, periksa apakah aman

Keenam pengujian pertama bisa dilakukan dalam satu jam untuk sebagian besar program utilitas, dan hampir selalu menemukan setidaknya satu perilaku yang tidak diharapkan.

169.6 Standar penulisan yang layak disepakati

Saya menyarankan tim menyepakati beberapa hal berikut, karena keseragaman pada hal-hal kecil sangat mengurangi waktu membaca kode orang lain.

1. Semua `DCL` di awal, dikelompokkan: parameter, variabel kerja, konstanta
2. Setiap perintah yang bisa gagal diikuti `MONMSG` yang eksplisit
3. Tidak ada `MONMSG` global tanpa penanganan
4. Nama library selalu dikualifikasi eksplisit di program produksi

5. Setiap program memeriksa lingkungan sebelum bertindak
6. Setiap operasi menunggu punya batas waktu
7. Pesan kesalahan menyebutkan program, konteks, dan tindakan yang disarankan
8. Pencatatan lewat utilitas bersama, bukan lewat `SNDMSG` langsung
9. Komentar menjelaskan mengapa, bukan apa
10. Nama variabel dalam bahasa yang konsisten, tidak campur

Butir kesembilan sering disalahpahami. Komentar yang menulis `/* Tambah satu ke penghitung */` di atas `CHGVAR VAR(&N) VALUE(&N + 1)` tidak menambah apa pun. Komentar yang menulis `/* Penghitung dipakai untuk commit berkala; jangan direset di dalam perulangan */` menyelamatkan orang berikutnya dari kesalahan yang tidak terlihat.

Bagian LI — Pendalaman Terapan

Bagian ini memperdalam sejumlah topik yang di bab-bab sebelumnya dibahas pada tingkat yang cukup untuk dipahami, tetapi belum cukup untuk diterapkan tanpa mencari rujukan lain. Susunannya mengikuti urutan bab aslinya, sehingga bisa dibaca sebagai pelengkap ketika Anda sedang mengerjakan area tersebut.

Bab 170. Pendalaman Exit Program

170.1 Merancang kebijakan sebelum menulis kode

Kesalahan yang paling sering terjadi saat memasang exit program bukan kesalahan teknis, melainkan mulai menulis program sebelum kebijakannya jelas. Akibatnya kebijakan tertanam di dalam kode, tersebar di beberapa program, dan tidak bisa dijelaskan kepada auditor tanpa membaca kode.

Rancang kebijakannya lebih dulu sebagai data. Bentuk minimum yang memadai:

```
CREATE TABLE SECLIB.KEBIJAKAN_AKSES (
    ID           BIGINT GENERATED ALWAYS AS IDENTITY,
    JENIS_AKSES  VARCHAR(20)  NOT NULL,
    USERID       VARCHAR(10)  NOT NULL,
    IP_POLA      VARCHAR(20)  NOT NULL,
    JAM_MULAI    TIME,
    JAM_SELESAI  TIME,
    HARI_BERLAKU VARCHAR(7)   DEFAULT '1234567',
    OBJEK_POLA   VARCHAR(30)  DEFAULT '*ALL',
    TINDAKAN     CHAR(1)      NOT NULL,
    ALASAN       VARCHAR(200),
    TIKET        VARCHAR(20),
    BERLAKU_DARI DATE         NOT NULL,
    BERLAKU_SAMPAI DATE,
    PRIMARY KEY (ID)
);

CREATE INDEX SECLIB.KEBIJAKAN_IX1
    ON SECLIB.KEBIJAKAN_AKSES (JENIS_AKSES, USERID);
```

Kolom `TINDAKAN` berisi `A` untuk izinkan, `T` untuk tolak, dan `L` untuk izinkan tetapi catat khusus. Kolom `TIKET` dan `ALASAN` memastikan setiap pengecualian punya jejak yang bisa ditinjau.

Keuntungan memisahkan kebijakan dari kode. Ketika auditor bertanya siapa saja yang boleh memakai ODBC di luar jam kerja, jawabannya satu kueri. Ketika kebijakan berubah, yang berubah adalah data dengan persetujuan, bukan program dengan siklus pengembangan penuh. Dan ketika terjadi insiden, Anda bisa menunjukkan kebijakan yang berlaku pada tanggal tertentu karena kolom masa berlaku menyimpannya.

170.2 Struktur exit program yang bisa dipelihara

```
**free
ctl-opt dftactgrp(*no) actgrp('SECGRP') option(*srcstmt);

dcl-proc periksaAkses export;
  dcl-pi *n ind;
    pJenis  varchar(20)  const;
    pUser   char(10)     const;
    pIP     varchar(15)  const;
    pObjek  varchar(30)  const;
    pAlasan varchar(100);
  end-pi;

  dcl-s wTindakan char(1) inz('T');   // bawaan: tolak

  exec sql
    SELECT TINDAKAN, COALESCE(ALASAN, '')
      INTO :wTindakan, :pAlasan
      FROM SECLIB.KEBIJAKAN_AKSES
     WHERE JENIS_AKSES = :pJenis
       AND USERID      = :pUser
       AND (:pIP LIKE IP_POLA OR IP_POLA = '*ALL')
       AND (:pObjek LIKE OBJEK_POLA OR OBJEK_POLA = '*ALL')
       AND (JAM_MULAI IS NULL
            OR CURRENT TIME BETWEEN JAM_MULAI AND JAM_SELESAI)
       AND BERLAKU_DARI <= CURRENT DATE
       AND (BERLAKU_SAMPAI IS NULL OR BERLAKU_SAMPAI >= CURRENT DATE)
     ORDER BY CASE TINDAKAN WHEN 'T' THEN 1 ELSE 2 END
     FETCH FIRST 1 ROW ONLY;

  if sqlcode <> 0 and sqlcode <> 100;
     // kegagalan kueri diperlakukan sebagai penolakan
     pAlasan = 'Kebijakan tidak dapat dibaca';
     catatAkses(pJenis: pUser: pIP: pObjek: 'T': pAlasan);
     return *off;
  endif;

  if sqlcode = 100;
     pAlasan = 'Tidak ada kebijakan yang mengizinkan';
     catatAkses(pJenis: pUser: pIP: pObjek: 'T': pAlasan);
     return *off;
  endif;

  catatAkses(pJenis: pUser: pIP: pObjek: wTindakan: pAlasan);
  return (wTindakan <> 'T');
end-proc;
```

Tiga keputusan desain yang layak diperhatikan.

Pertama, nilai bawaan `wTindakan` adalah tolak. Bila apa pun gagal di tengah jalan, hasilnya penolakan, bukan izin.

Kedua, `ORDER BY` menempatkan baris penolakan lebih dulu. Bila seorang pengguna punya satu kebijakan yang mengizinkan dan satu yang menolak, penolakan menang. Ini pilihan yang benar untuk keamanan, dan ia harus eksplisit karena tanpa `ORDER BY` hasilnya bergantung pada urutan yang dikembalikan basis data.

Ketiga, setiap jalur keluar memanggil `catatAkses`. Exit program yang hanya mencatat penolakan kehilangan separuh nilainya; catatan akses yang diizinkan adalah bahan untuk mendeteksi pola yang mencurigakan.

170.3 Menguji exit program tanpa mengunci diri sendiri

Ini risiko nyata: exit program yang keliru bisa memblokir seluruh akses termasuk akses Anda sendiri untuk memperbaikinya.

Urutan pemasangan yang aman:

1. Pasang di lingkungan uji, jalankan dalam mode pencatatan saja minimal dua pekan
2. Analisis catatan: apa yang akan ditolak bila penegakan diaktifkan
3. Pastikan ada satu jalur akses yang tidak melewati exit point yang bersangkutan — misalnya konsol fisik atau sesi 5250 lokal
4. Aktifkan penegakan di luar jam sibuk, dengan seseorang yang siap mencabut pendaftarannya
5. Pantau selama 48 jam

```
/* Pencabutan darurat, siapkan perintahnya sebelum mengaktifkan */
RMVEXITPGM EXITPNT(QIBM_QZDA_INIT) FORMAT(ZDAI0100) PGMNBR(1)
ENDTCPSVR SERVER(*DATABASE)
STRTCPSVR SERVER(*DATABASE)
```

Simpan tiga perintah itu di tempat yang bisa dijangkau tanpa ODBC.

170.4 Kinerja exit program

Exit program dipanggil pada setiap koneksi, dan pada sistem dengan ribuan koneksi per menit biayanya menjadi nyata.

Praktik	Dampak
Kueri kebijakan dengan index yang tepat	Menentukan; tanpa index, setiap koneksi memindai tabel
Simpan hasil dalam cache di activation group	Mengurangi kueri berulang untuk pengguna yang sama
Hindari membuka dan menutup kursor tiap panggilan	Pakai `CLOSQLCSR(*ENDACTGRP)`
Tulis catatan secara asinkron lewat data queue	Mencegah penulisan log memperlambat koneksi
Batasi ukuran tabel kebijakan	Arsipkan baris yang sudah tidak berlaku

Baris keempat adalah teknik yang layak dipertimbangkan bila volume koneksi tinggi: exit program mengirim catatan ke data queue dan langsung kembali, sementara job terpisah membaca antrean itu dan menulisnya ke tabel. Dengan begitu keputusan izin tidak menunggu penulisan basis data.

Bab 171. Pendalaman TLS dan Sertifikat

171.1 Rantai kepercayaan, dijelaskan dari sisi operasional

Sertifikat server tidak berdiri sendiri. Ia ditandatangani oleh sebuah Certificate Authority, yang mungkin ditandatangani oleh CA lain di atasnya, sampai ke root CA yang dipercaya secara bawaan oleh klien.

Kegagalan handshake TLS hampir selalu berasal dari salah satu dari empat hal:

Penyebab	Gejala di sisi klien	Cara memastikan
Sertifikat kedaluwarsa	Peringatan tanggal	Periksa `VALIDITY_END`
Rantai tidak lengkap	Sertifikat tidak dipercaya	`openssl s_client` menampilkan rantai
Nama tidak cocok	Peringatan nama host	Bandingkan CN dan SAN dengan nama yang dipakai
Protokol atau cipher tidak cocok	Handshake gagal tanpa detail	Periksa `QSSLPCL` dan `QSSLCSL`

```
# Menampilkan rantai lengkap dan protokol yang disepakati
openssl s_client -connect layanan.bank.co.id:443 -showcerts </dev/null 2>/dev/null \
  | grep -E 'subject=|issuer=|Protocol|Cipher'

# Memeriksa apakah nama host ada di SAN
echo | openssl s_client -connect layanan.bank.co.id:443 2>/dev/null \
  | openssl x509 -noout -text | grep -A1 'Subject Alternative Name'
```

Tentang rantai yang tidak lengkap. Ini penyebab yang paling membingungkan karena koneksi berhasil dari sebagian klien dan gagal dari sebagian lain. Klien yang kebetulan sudah punya sertifikat perantara di penyimpanannya akan berhasil; yang tidak punya akan gagal. Perbaikannya adalah memasang sertifikat perantara di sisi server, bukan di setiap klien.

171.2 Merencanakan perpanjangan

Perpanjangan sertifikat yang dilakukan pada hari kedaluwarsa hampir selalu berakhir dengan pemadaman singkat. Rencana yang wajar:

Waktu sebelum kedaluwarsa	Tindakan
120 hari	Peringatan otomatis pertama, buka tiket
90 hari	Ajukan permintaan sertifikat baru ke CA
60 hari	Sertifikat baru diterima dan diuji di lingkungan uji
30 hari	Pasang di produksi, kedua sertifikat berdampingan
14 hari	Alihkan layanan ke sertifikat baru, pantau
0 hari	Sertifikat lama kedaluwarsa tanpa dampak

Langkah pada 30 hari adalah yang membuat prosedur ini aman: sertifikat lama dan baru ada bersamaan, sehingga peralihan bisa dibalik dalam hitungan menit bila ada klien yang bermasalah.

```
-- Pemantauan yang menghasilkan tiket, bukan sekadar peringatan
SELECT CERTIFICATE_LABEL, SUBJECT_COMMON_NAME,
       VALIDITY_END,
       DAYS(VALIDITY_END) - DAYS(CURRENT DATE) AS sisa_hari,
       CASE
         WHEN DAYS(VALIDITY_END) - DAYS(CURRENT DATE) <= 14 THEN 'KRITIS'
         WHEN DAYS(VALIDITY_END) - DAYS(CURRENT DATE) <= 30 THEN 'TINGGI'
         WHEN DAYS(VALIDITY_END) - DAYS(CURRENT DATE) <= 90 THEN 'SEDANG'
         ELSE 'RENDAH'
       END AS prioritas
  FROM QSYS2.CERTIFICATE_INFO(
         CERTIFICATE_STORE_PASSWORD => '*NOPWD')
 WHERE VALIDITY_END < CURRENT DATE + 120 DAYS
 ORDER BY VALIDITY_END;
```

171.3 Sertifikat klien untuk integrasi mesin ke mesin

Untuk integrasi antar sistem, autentikasi berbasis sertifikat klien lebih kuat daripada nama pengguna dan kata sandi: tidak ada kredensial yang bisa bocor lewat log, dan pencabutannya seketika.

Pertimbangan operasional yang perlu disiapkan:

1. Setiap sistem mitra mendapat sertifikat sendiri, bukan sertifikat bersama
2. Masa berlaku lebih pendek daripada sertifikat server, karena pencabutan lebih sering dibutuhkan
3. Prosedur pencabutan harus ada dan pernah diuji
4. Pemetaan sertifikat ke profil pengguna IBM i didokumentasikan
5. Ada jalur cadangan bila sertifikat bermasalah di luar jam kerja

Butir pertama sering dilanggar demi kemudahan, dan akibatnya baru terasa ketika satu mitra harus dicabut aksesnya: mencabut sertifikat bersama memutus seluruh mitra sekaligus.

Bab 172. Pendalaman Pertukaran Data dan CCSID

172.1 Peta masalah menurut arah aliran data

Masalah encoding berbeda tergantung arah, dan membedakannya mempercepat diagnosis.

Arah	Masalah khas	Titik pemeriksaan
IBM i ke berkas	Karakter rusak di berkas	`STMFCCSID` pada perintah salin
Berkas ke IBM i	Karakter rusak di tabel	CCSID kolom tujuan
IBM i ke ODBC	Karakter rusak di aplikasi	CCSID kolom, terutama 65535
ODBC ke IBM i	Data tersimpan salah	CCSID job dan koneksi
IBM i ke IBM i	Jarang bermasalah	CCSID kedua sistem
API JSON	Karakter non-ASCII rusak	CCSID 1208 di seluruh jalur

172.2 Menghitung panjang dalam byte versus karakter

Pada CCSID satu byte per karakter, panjang byte dan panjang karakter sama, dan seluruh kode yang mengasumsikan kesamaan itu berjalan benar. Pada UTF-8 asumsi itu runtuh.

```
-- Menemukan data yang akan terpotong bila dipindahkan ke kolom lebih pendek
SELECT ID_NASABAH, NAMA,
       LENGTH(NAMA) AS byte,
       CHARACTER_LENGTH(NAMA) AS karakter,
       LENGTH(NAMA) - CHARACTER_LENGTH(NAMA) AS selisih
  FROM DTALIB.CIF_UTAMA
 WHERE LENGTH(NAMA) > CHARACTER_LENGTH(NAMA)
 ORDER BY selisih DESC
 FETCH FIRST 50 ROWS ONLY;
```

Akibat yang paling merusak bukan pemotongan itu sendiri, melainkan pemotongan di tengah urutan byte sebuah karakter. Hasilnya adalah byte tidak lengkap yang merusak seluruh berkas ekspor, bukan hanya satu record. Program yang memotong teks untuk memenuhi panjang field harus memotong pada batas karakter, bukan pada batas byte.

```
**free
// Salah: memotong pada batas byte
wHasil = %subst(wNama: 1: 40);

// Benar untuk data UTF-8: potong pada batas karakter,
// lalu verifikasi hasilnya tetap valid
if %len(%trim(wNama)) > 40;
   wHasil = potongAman(wNama: 40);
else;
   wHasil = wNama;
endif;
```

172.3 Kontrak pertukaran data yang lengkap

Sebagian besar masalah pertukaran data berasal dari kontrak yang tidak menyebutkan hal-hal yang dianggap jelas oleh kedua pihak, padahal tidak.

Isi minimum sebuah kontrak pertukaran berkas:

Butir	Contoh nilai
Encoding	UTF-8 tanpa BOM
Akhir baris	LF
Pemisah kolom	Koma
Pembungkus teks	Tanda kutip ganda
Perlakuan pemisah di dalam data	Dibungkus dan di-escape
Format tanggal	YYYY-MM-DD
Format angka	Titik desimal, tanpa pemisah ribuan
Perlakuan nilai kosong	String kosong, bukan kata NULL
Baris judul	Ada, satu baris
Baris trailer	Ada, berisi jumlah dan total
Nama berkas	Pola dengan tanggal
Penanda selesai	Berkas kedua berakhiran .done
Jam kirim	Beserta toleransi keterlambatan
Tindakan bila validasi gagal	Tolak seluruh berkas

Butir kedelapan sering menjadi sumber perselisihan. Nilai kosong yang dikirim sebagai kata `NULL` akan tersimpan sebagai teks empat huruf bila penerima tidak menerjemahkannya, dan itu jenis kesalahan yang lolos dari seluruh validasi format.

🏗️ Bab 173. Pendalaman Arsitektur Pesan

173.1 Memilih pola, bukan memilih produk

Pertanyaan yang sering diajukan adalah apakah memakai data queue, MQ, atau REST. Pertanyaan yang lebih berguna mendahuluinya: apakah interaksi ini sinkron atau asinkron, dan apa yang terjadi bila pihak lain tidak tersedia?

Pertanyaan	Bila jawabannya ya	Bila tidak
Pemanggil menunggu hasil untuk melanjutkan?	Sinkron	Asinkron

Pertanyaan	Bila jawabannya ya	Bila tidak
Kehilangan pesan dapat diterima?	Boleh tanpa persistensi	Perlu antrean persisten
Pihak lain sering tidak tersedia?	Wajib antrean	Panggilan langsung cukup
Urutan pesan penting?	Antrean FIFO tunggal	Bebas
Satu pesan dikonsumsi banyak pihak?	Publikasi dan langganan	Antrean titik ke titik
Lintas platform?	MQ atau REST	Data queue cukup

Setelah enam pertanyaan ini dijawab, pilihan produknya biasanya tinggal satu atau dua.

173.2 Pola antrean surat mati

Pesan yang gagal diproses berulang kali tidak boleh terus dicoba selamanya, dan tidak boleh pula dibuang. Antrean surat mati adalah tempat ketiga.

```
CREATE TABLE APPWRK.PESAN_GAGAL (
    ID          BIGINT GENERATED ALWAYS AS IDENTITY,
    WAKTU_MASUK TIMESTAMP   NOT NULL,
    SUMBER      VARCHAR(30) NOT NULL,
    KORELASI_ID VARCHAR(64),
    ISI_PESAN   VARCHAR(4000),
    JML_COBA    SMALLINT    NOT NULL DEFAULT 0,
    KESALAHAN   VARCHAR(500),
    STATUS      CHAR(1)     NOT NULL DEFAULT 'B',
    DITANGANI_OLEH VARCHAR(30),
    WAKTU_SELESAI  TIMESTAMP,
    PRIMARY KEY (ID)
);
```

Alur yang disarankan: coba tiga kali dengan jeda bertingkat, lalu pindahkan ke tabel ini dengan status belum ditangani. Sebuah laporan harian menampilkan isinya, dan setiap baris harus diselesaikan oleh manusia — diperbaiki dan dikirim ulang, atau dinyatakan tidak perlu diproses dengan alasan tertulis.

Yang membuat pola ini bekerja adalah laporan hariannya. Antrean surat mati tanpa seseorang yang meninjaunya setiap hari hanyalah tempat pesan menghilang dengan cara yang lebih rapi.

173.3 Mengukur kesehatan aliran pesan

Tiga metrik yang paling menunjukkan kesehatan integrasi asinkron:

Metrik	Arti	Ambang yang wajar
Kedalaman antrean	Pesan menunggu diproses	Naik terus berarti konsumen tertinggal
Usia pesan tertua	Berapa lama menunggu	Melebihi SLA berarti masalah
Laju masuk dibanding laju keluar	Keseimbangan	Masuk lebih besar berarti akan menumpuk

Metrik kedua adalah yang paling sering diabaikan dan paling informatif. Antrean dengan seribu pesan yang seluruhnya masuk semenit lalu adalah kondisi normal pada jam sibuk. Antrean dengan sepuluh pesan yang tertua berumur empat jam adalah masalah.

```
SELECT DATA_QUEUE_NAME,
       CURRENT_MESSAGES AS kedalaman,
       MAXIMUM_MESSAGES AS kapasitas,
       DECIMAL(CURRENT_MESSAGES * 100.0 /
               NULLIF(MAXIMUM_MESSAGES, 0), 5, 1) AS persen_penuh
  FROM QSYS2.DATA_QUEUE_INFO
 WHERE DATA_QUEUE_LIBRARY = 'APPWRK'
 ORDER BY persen_penuh DESC;
```

Bab 174. Pendalaman Hybrid Cloud

174.1 Apa yang sebenarnya dipindahkan

Pembicaraan tentang hybrid cloud untuk IBM i sering kabur karena tidak jelas apa yang dimaksud dipindahkan. Ada empat kemungkinan yang sangat berbeda.

Yang dipindahkan	Risiko	Manfaat
Lingkungan DR	Rendah	Biaya data center kedua turun
Lingkungan uji dan pengembangan	Rendah	Kapasitas elastis
Backup dan arsip jangka panjang	Rendah sampai sedang	Biaya penyimpanan turun
Produksi	Tinggi	Bergantung kasus

Urutan yang lazim dan masuk akal adalah dari atas ke bawah. Memindahkan DR lebih dulu memberi pengalaman nyata dengan latensi, biaya, dan operasi di lingkungan itu, sebelum keputusan yang lebih besar diambil.

174.2 Pertanyaan yang menentukan kelayakan

Sebelum pembahasan teknis dimulai, ada pertanyaan yang jawabannya bisa menghentikan seluruh rencana:

1. Apakah ketentuan yang berlaku mengizinkan data nasabah berada di luar wilayah tertentu?
2. Apakah penyedia memenuhi persyaratan kepatuhan institusi Anda?
3. Apakah latensi antara lokasi memenuhi kebutuhan replikasi?
4. Bagaimana data keluar bila suatu saat perlu berpindah penyedia?
5. Siapa yang bertanggung jawab bila terjadi insiden keamanan di sisi penyedia?

Pertanyaan keempat sering terlewat pada tahap perencanaan dan menjadi mahal kemudian. Biaya memindahkan data keluar dari sebuah penyedia bisa jauh melebihi biaya memasukkannya, dan itu menciptakan ketergantungan yang tidak terlihat di perhitungan awal.

174.3 Latensi dan pengaruhnya pada replikasi

Latensi antar lokasi menentukan mode replikasi yang mungkin.

Latensi bolak-balik	Replikasi sinkron	Replikasi asinkron
Di bawah 2 ms	Layak	Layak
2 sampai 10 ms	Layak dengan dampak	Layak
10 sampai 50 ms	Tidak disarankan	Layak
Di atas 50 ms	Tidak layak	Layak dengan lag lebih besar

Alasannya bersifat fisika, bukan konfigurasi: replikasi sinkron menunggu konfirmasi dari sisi jauh sebelum

transaksi dinyatakan selesai, sehingga setiap transaksi menanggung latensi itu. Pada beban seribu transaksi per detik, tambahan 10 ms per transaksi mengubah kapasitas sistem secara mendasar.

Konsekuensi praktisnya untuk RPO. RPO nol menuntut replikasi sinkron, dan replikasi sinkron menuntut jarak yang dekat. Institusi yang menetapkan RPO nol dan sekaligus menuntut DR di kota berbeda yang jauh sedang menetapkan dua hal yang tidak bisa dipenuhi bersamaan. Pembahasan ini lebih baik terjadi pada tahap perencanaan daripada pada tahap implementasi.

📦 Bab 175. Pendalaman Migrasi Data

175.1 Profiling yang sistematis

Bab 101 memperkenalkan profiling. Berikut kerangka yang lebih lengkap, disusun agar bisa dijalankan sebagai rangkaian kueri untuk setiap kolom penting.

```
-- Kerangka profil satu kolom, disalin dan disesuaikan per kolom
SELECT 'KD_PRODUK' AS kolom,
       COUNT(*) AS total_baris,
       COUNT(KD_PRODUK) AS terisi,
       COUNT(*) - COUNT(KD_PRODUK) AS jml_null,
       SUM(CASE WHEN TRIM(KD_PRODUK) = '' THEN 1 ELSE 0 END) AS kosong,
       COUNT(DISTINCT KD_PRODUK) AS nilai_unik,
       MIN(LENGTH(TRIM(KD_PRODUK))) AS panjang_min,
       MAX(LENGTH(TRIM(KD_PRODUK))) AS panjang_maks,
       SUM(CASE WHEN KD_PRODUK <> UPPER(KD_PRODUK) THEN 1 ELSE 0 END)
         AS ada_huruf_kecil,
       SUM(CASE WHEN TRANSLATE(KD_PRODUK, '    ',
                  'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
                     <> REPEAT(' ', LENGTH(KD_PRODUK))
                THEN 1 ELSE 0 END) AS ada_karakter_aneh
  FROM SUMBER.REKENING_LAMA;
```

Kolom terakhir menemukan karakter yang tidak diharapkan — spasi di tengah, tanda baca, atau karakter kontrol. Ini yang paling sering merusak migrasi dan paling jarang diperiksa.

175.2 Merancang transformasi yang bisa diaudit

Aturan transformasi yang tertanam di dalam kode sulit ditinjau dan sulit diubah. Simpan sebagai data.

```
CREATE TABLE MIGRASI.ATURAN_KONVERSI (
    KD_ATURAN   VARCHAR(20)  NOT NULL,
    NILAI_ASAL  VARCHAR(50)  NOT NULL,
    NILAI_TUJUAN VARCHAR(50) NOT NULL,
    KETERANGAN  VARCHAR(200),
    DISETUJUI_OLEH VARCHAR(30),
    TANGGAL     DATE,
    PRIMARY KEY (KD_ATURAN, NILAI_ASAL)
);

-- Contoh isi
INSERT INTO MIGRASI.ATURAN_KONVERSI VALUES
  ('K01', 'TAB', 'DP01', 'Tabungan reguler', 'Kepala Operasi', '2026-08-01'),
  ('K01', 'TBJ', 'DP02', 'Tabungan berjangka', 'Kepala Operasi', '2026-08-01'),
  ('K01', 'GIR', 'GR01', 'Giro', 'Kepala Operasi', '2026-08-01');

-- Nilai sumber yang belum punya aturan
SELECT DISTINCT s.KDPRD
  FROM SUMBER.REKENING_LAMA s
 WHERE NOT EXISTS (SELECT 1 FROM MIGRASI.ATURAN_KONVERSI a
                    WHERE a.KD_ATURAN = 'K01'
                      AND a.NILAI_ASAL = s.KDPRD);
```

Kueri terakhir wajib mengembalikan nol baris sebelum migrasi dijalankan. Bila tidak, ada nilai sumber yang akan diterjemahkan menjadi kosong atau ditolak, dan lebih baik mengetahuinya sekarang.

175.3 Uji coba yang benar-benar menguji

Uji coba migrasi yang hanya memakai sepersepuluh data tidak menguji hal-hal yang paling penting: kasus pinggiran ada di data yang jarang, dan kapasitas serta durasi hanya terlihat pada volume penuh.

Uji coba	Cakupan data	Yang diuji
Pertama	1.000 baris terpilih	Logika transformasi
Kedua	Seluruh data ke UAT	Kasus pinggiran, penolakan
Ketiga	Seluruh data, waktu diukur	Durasi, kapasitas, jendela

Uji coba kedua harus mencakup seluruh data, bukan sampel, karena tujuannya menemukan baris yang tidak biasa. Sampel acak justru cenderung melewatkannya.

Cara memilih 1.000 baris untuk uji pertama. Jangan acak. Pilih secara sengaja: baris tertua, baris terbaru, nilai ekstrem pada setiap kolom numerik, baris dengan kolom kosong, baris dengan karakter tidak biasa, dan baris dari setiap kategori. Seribu baris yang dipilih begini menguji lebih banyak daripada sepuluh ribu baris acak.

175.4 Rekonsiliasi berlapis

Rekonsiliasi yang hanya membandingkan jumlah baris tidak cukup. Empat lapis yang saya sarankan:

```
-- Lapis 1: jumlah
SELECT (SELECT COUNT(*) FROM SUMBER.REKENING_LAMA) AS sumber,
       (SELECT COUNT(*) FROM DTALIB.REKENING) AS tujuan,
       (SELECT COUNT(*) FROM MIGRASI.TOLAK) AS ditolak
  FROM SYSIBM.SYSDUMMY1;

-- Lapis 2: total nilai keuangan
SELECT (SELECT SUM(SALDOAKHIR) FROM SUMBER.REKENING_LAMA) AS sumber,
       (SELECT SUM(SALDO) FROM DTALIB.REKENING) AS tujuan
  FROM SYSIBM.SYSDUMMY1;

-- Lapis 3: distribusi per kategori
SELECT KDPRD AS kode, COUNT(*) AS jml, SUM(SALDOAKHIR) AS total
  FROM SUMBER.REKENING_LAMA GROUP BY KDPRD ORDER BY kode;

-- Lapis 4: perbandingan baris demi baris lewat tabel jejak
SELECT COUNT(*) AS jml_tidak_cocok
  FROM SUMBER.REKENING_LAMA s
  JOIN MIGRASI.JEJAK j ON j.KUNCI_SUMBER = s.REKNO
  JOIN DTALIB.REKENING t ON t.NO_REK = j.KUNCI_TUJUAN
 WHERE s.SALDOAKHIR <> t.SALDO;
```

Lapis keempat adalah yang paling menyeluruh dan paling mahal untuk dijalankan pada data besar. Jalankan pada uji coba kedua dan ketiga, dan pada migrasi produksi jalankan sebagai proses latar setelah sistem dibuka — dengan hasilnya diperiksa sebelum akhir hari pertama.

Bab 176. Pendalaman Cutover

176.1 Tiga gerbang keputusan, dijabarkan

Bab 77 menyebut tiga titik keputusan. Berikut kriteria yang layak ditetapkan untuk masing-masing.

Gerbang pertama: mulai atau batalkan. Diambil sebelum perubahan apa pun dilakukan, sehingga pembatalan tidak berbiaya.

Kriteria	Nilai yang dituntut
Backup penuh selesai dan terverifikasi	Ya
Seluruh peran terisi dan hadir	Ya
Uji coba terakhir berhasil	Ya
Pembekuan perubahan berjalan	Ya
Rencana rollback tertulis dan disepakati	Ya
Jendela penuh tersedia	Ya

Gerbang kedua: lanjut ke verifikasi bisnis atau rollback. Diambil setelah eksekusi teknis, sebelum pengguna bisnis dilibatkan.

Kriteria	Nilai yang dituntut
Seluruh angka kontrol cocok	Selisih nol
Tidak ada job gagal	Ya
Seluruh layanan berjalan	Ya
Waktu terpakai masih dalam rencana	Ya

Gerbang ketiga: buka layanan atau rollback. Diambil setelah verifikasi bisnis.

Kriteria	Nilai yang dituntut
Perwakilan bisnis menerima, tertulis	Ya
Satu transaksi berhasil dari tiap kanal	Ya
Tim dukungan siap 72 jam	Ya
Rollback masih mungkin bila dibutuhkan	Ya

176.2 Hypercare

Periode setelah cutover, sering disebut hypercare, adalah bagian yang paling sering disiapkan seadanya.

Yang layak disiapkan:

1. Jadwal jaga yang diperluas selama 72 jam pertama
2. Saluran komunikasi khusus untuk masalah terkait cutover
3. Pemantauan tambahan pada metrik yang paling mungkin terpengaruh
4. Perbandingan harian volume transaksi dengan pekan sebelumnya
5. Rapat singkat harian selama sepekan pertama
6. Kriteria eksplisit untuk menyatakan hypercare berakhir

Butir keenam sering tidak ada, dan akibatnya hypercare berlanjut tanpa batas atau berhenti tanpa keputusan. Kriteria yang wajar: tidak ada insiden tingkat satu atau dua terkait cutover selama lima hari kerja berturut-turut, dan seluruh temuan tingkat tinggi sudah ditutup.

176.3 Metrik ruang kendali

Selama cutover berlangsung, sebuah papan yang menampilkan beberapa angka membuat seluruh tim punya gambaran yang sama tanpa harus bertanya.

Metrik	Diperbarui	Arti
Kemajuan migrasi	Tiap 15 menit	Persentase objek atau baris selesai
Job batch selesai	Tiap 15 menit	Berapa dari total
Status interface	Tiap 30 menit	Berapa yang sudah terhubung
Rekonsiliasi	Setelah tiap tahap	Persentase cocok
Isu terbuka	Waktu nyata	Jumlah dan tingkatnya
Waktu terhadap rencana	Tiap tahap	Lebih cepat atau lebih lambat

Angka terakhir adalah yang paling menentukan keputusan. Sebuah cutover yang tertinggal dua jam pada pertengahan jendela hampir pasti tidak akan selesai tepat waktu, dan mengetahuinya lebih awal memberi ruang untuk memutuskan dengan tenang alih-alih terburu-buru pada menit terakhir.

🗄️ Bagian LII — Pendalaman Db2 for i Lanjutan

Bab 177. Partisi Tabel

177.1 Kapan partisi layak dipertimbangkan

Tabel riwayat transaksi yang tumbuh dua juta baris per hari akan mencapai tujuh ratus juta baris dalam

setahun. Pada ukuran itu, beberapa operasi yang sebelumnya sepele menjadi mahal: penghapusan data lama, reorganisasi, pembangunan ulang index, dan backup.

Partisi memecah satu tabel logis menjadi beberapa bagian fisik yang dikelola terpisah, sementara aplikasi tetap melihatnya sebagai satu tabel.

Manfaat	Penjelasan
Penghapusan data lama menjadi murah	Hapus satu partisi, bukan jutaan baris
Reorganisasi per bagian	Tidak perlu menyentuh seluruh tabel
Backup selektif	Partisi lama bisa dibekukan
Kueri yang terbatas periode lebih cepat	Optimizer melewati partisi tak relevan
Pemeliharaan index lebih ringan	Index per partisi

177.2 Membuat tabel berpartisi

```
CREATE TABLE DTALIB.TRXHIST (
    NO_URUT    BIGINT        NOT NULL,
    NO_REK     CHAR(16)      NOT NULL,
    TGL_BUKU   DATE          NOT NULL,
    KD_TRX     CHAR(4)       NOT NULL,
    NOMINAL    DECIMAL(17,2) NOT NULL,
    KETERANGAN VARCHAR(100)
)
PARTITION BY RANGE (TGL_BUKU) (
    PARTITION P2025Q1 STARTING ('2025-01-01') ENDING ('2025-03-31'),
    PARTITION P2025Q2 STARTING ('2025-04-01') ENDING ('2025-06-30'),
    PARTITION P2025Q3 STARTING ('2025-07-01') ENDING ('2025-09-30'),
    PARTITION P2025Q4 STARTING ('2025-10-01') ENDING ('2025-12-31'),
    PARTITION P2026Q1 STARTING ('2026-01-01') ENDING ('2026-03-31'),
    PARTITION P2026Q2 STARTING ('2026-04-01') ENDING ('2026-06-30'),
    PARTITION P2026Q3 STARTING ('2026-07-01') ENDING ('2026-09-30'),
    PARTITION P2026Q4 STARTING ('2026-10-01') ENDING ('2026-12-31')
);
```

Pemilihan kunci partisi adalah keputusan yang sulit diubah kemudian. Kunci yang tepat adalah kolom yang muncul di hampir setiap kueri sebagai filter, dan yang nilainya maju searah waktu. Untuk tabel transaksi perbankan, tanggal buku hampir selalu pilihan yang benar.

Kunci yang keliru — misalnya kode cabang — menghasilkan partisi yang tidak pernah bisa dibuang, karena tidak ada cabang yang datanya kedaluwarsa sebagai satu kesatuan.

177.3 Mengelola siklus hidup partisi

```
-- Menambah partisi untuk periode berikutnya
ALTER TABLE DTALIB.TRXHIST
  ADD PARTITION P2027Q1
  STARTING ('2027-01-01') ENDING ('2027-03-31');

-- Melepas partisi lama menjadi tabel tersendiri, untuk diarsipkan
ALTER TABLE DTALIB.TRXHIST
  DETACH PARTITION P2025Q1 INTO DTALIB.TRXHIST_2025Q1;

-- Menghapus partisi beserta datanya
ALTER TABLE DTALIB.TRXHIST
  DROP PARTITION P2025Q1;
```

Operasi DETACH jauh lebih baik daripada DROP untuk data yang punya kewajiban retensi. Ia memindahkan partisi menjadi tabel mandiri yang bisa disimpan, dipindahkan ke ASP lain, atau di-backup lalu dihapus — semuanya tanpa menyentuh tabel utama.

Jadwalkan penambahan partisi jauh sebelum dibutuhkan. Sebuah insert yang jatuh di luar seluruh rentang partisi akan gagal. Bila partisi triwulan berikutnya tidak dibuat sebelum triwulan itu dimulai, transaksi pertama pada hari pertama akan ditolak — dan itu jenis kegagalan yang terjadi tepat pada pergantian periode, ketika perhatian sedang tersita hal lain.

```
-- Pemeriksaan yang layak dijadwalkan bulanan
SELECT TABLE_NAME, PARTITION_NAME,
       PARTITION_TYPE, NUMBER_PARTITIONING_KEYS,
       NUMBER_ROWS
  FROM QSYS2.SYSPARTITIONSTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND TABLE_NAME = 'TRXHIST'
 ORDER BY PARTITION_NAME DESC;
```

177.4 Dampak pada kueri

Partisi hanya memberi manfaat kinerja bila kueri menyertakan kunci partisi dalam kondisinya.

```
-- Cepat: optimizer hanya membaca satu partisi
SELECT SUM(NOMINAL) FROM DTALIB.TRXHIST
 WHERE TGL_BUKU BETWEEN '2026-09-01' AND '2026-09-30';

-- Lambat: seluruh partisi dibaca
SELECT SUM(NOMINAL) FROM DTALIB.TRXHIST
 WHERE NO_REK = '1234567890123456';
```

Kueri kedua menunjukkan batas dari pendekatan ini. Pencarian berdasarkan nomor rekening tanpa batasan tanggal harus memindai seluruh partisi. Bila pola akses itu sering terjadi, index pada nomor rekening tetap dibutuhkan, dan partisi tidak menggantikannya.

Nasihat praktis untuk aplikasi perbankan. Layar inquiry mutasi rekening hampir selalu punya rentang tanggal bawaan — tiga bulan terakhir, misalnya. Pastikan rentang itu ikut terkirim ke kueri, bukan hanya dipakai untuk menyaring hasil di sisi tampilan. Perbedaannya sangat besar pada tabel berpartisi.

Bab 178. Materialized Query Table

178.1 Konsep dan kapan dipakai

MQT adalah tabel yang isinya merupakan hasil sebuah kueri, disimpan secara fisik, dan bisa dipakai optimizer secara otomatis untuk menjawab kueri lain yang setara.

Untuk laporan agregat yang dijalankan berulang kali atas data yang sama, ini bisa mengubah kueri berjam-jam menjadi hitungan detik.

```
CREATE TABLE DTALIB.MQT_SALDO_HARIAN AS (
  SELECT TGL_BUKU,
         KD_CABANG,
         KD_PRODUK,
         COUNT(*)      AS JML_REKENING,
         SUM(SALDO)    AS TOTAL_SALDO,
         AVG(SALDO)    AS RATA_SALDO
    FROM DTALIB.REKENING_HARIAN
   GROUP BY TGL_BUKU, KD_CABANG, KD_PRODUK
)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
ENABLE QUERY OPTIMIZATION;

-- Isi pertama kali
REFRESH TABLE DTALIB.MQT_SALDO_HARIAN;
```

178.2 Pilihan pemeliharaan

Pilihan	Perilaku	Cocok untuk
`REFRESH DEFERRED`	Diperbarui saat diminta	Data yang berubah dalam batch
`REFRESH IMMEDIATE`	Diperbarui saat tabel dasar berubah	Data yang harus selalu terkini
`MAINTAINED BY USER`	Anda yang mengisi	Kendali penuh atas waktu
`MAINTAINED BY SYSTEM`	Sistem yang mengisi	Lebih sederhana, lebih mahal

Untuk lingkungan core banking, kombinasi `REFRESH DEFERRED` dengan `MAINTAINED BY USER` hampir selalu pilihan yang tepat: MQT diisi sebagai bagian dari fase EOD, setelah data harian final, dan tidak menambah beban pada jam transaksi.

Peringatan yang penting. MQT yang diaktifkan untuk optimasi kueri akan dipakai optimizer meskipun isinya sudah usang. Bila MQT tidak di-refresh setelah data berubah, kueri akan menjawab dengan angka lama tanpa peringatan apa pun.

Pengamanannya: jadikan refresh MQT sebagai langkah wajib dalam rangkaian EOD, dengan penanda status seperti fase lainnya, dan tambahkan pemeriksaan yang membandingkan tanggal terakhir refresh dengan tanggal buku berjalan.

```
-- Pemeriksaan kesegaran MQT
SELECT TABLE_NAME,
       LAST_REFRESH_TIMESTAMP,
       DAYS(CURRENT DATE) - DAYS(DATE(LAST_REFRESH_TIMESTAMP)) AS umur_hari,
       ENABLE_QUERY_OPTIMIZATION
  FROM QSYS2.SYSTABLES
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND TABLE_TYPE = 'M';
```

178.3 Memverifikasi bahwa MQT benar-benar dipakai

Membuat MQT tidak menjamin optimizer memakainya. Verifikasi dengan pesan debug.

```
CALL QSYS2.OVERRIDE_QAQQINI(1, 'MESSAGES_DEBUG', '*YES');
CALL QSYS2.OVERRIDE_QAQQINI(1, 'MATERIALIZED_QUERY_TABLE_REFRESH_AGE', '*ANY');

-- Jalankan kueri yang seharusnya dijawab oleh MQT,
-- lalu periksa pesan CPI4348 di job log
```

Pesan `CPI4348` menandakan MQT dipakai. Bila tidak muncul, kemungkinan penyebabnya: kueri tidak cukup mirip dengan definisi MQT, MQT tidak diaktifkan untuk optimasi, atau parameter umur refresh membatasi pemakaiannya.

Bab 179. Bekerja dengan Objek Besar

179.1 Kapan LOB dibutuhkan, dan kapan tidak

Kolom BLOB dan CLOB memungkinkan penyimpanan data berukuran besar di dalam basis data: dokumen, gambar, berkas tanda tangan digital.

Pertanyaan yang perlu dijawab sebelum memakainya: apakah data itu memang harus berada di dalam basis data?

Pertimbangan	Di dalam basis data	Di IFS dengan penunjuk
Integritas transaksional	Terjaga	Tidak otomatis
Backup bersama data	Ya	Perlu diatur terpisah
Ukuran tabel	Membesar cepat	Tetap ramping
Kinerja kueri pada tabel itu	Terpengaruh	Tidak terpengaruh
Akses dari aplikasi web	Perlu lapisan	Langsung

Untuk dokumen pendukung nasabah — salinan kartu identitas, misalnya — pendekatan yang lazim dan masuk akal adalah menyimpan berkasnya di IFS dan menyimpan jalurnya di tabel, dengan konsekuensi bahwa integritas antara keduanya harus dijaga oleh aplikasi.

Untuk data yang integritasnya kritikal dan ukurannya sedang — tanda tangan digital sebuah transaksi, misalnya — penyimpanan di dalam basis data lebih tepat karena ia ikut dalam transaksi yang sama.

179.2 Pola penyimpanan hibrida

```
CREATE TABLE DTALIB.DOKUMEN (
    ID_DOK      BIGINT GENERATED ALWAYS AS IDENTITY,
    ID_NASABAH  BIGINT       NOT NULL,
    JENIS_DOK   VARCHAR(20)  NOT NULL,
    NAMA_BERKAS VARCHAR(200) NOT NULL,
    JALUR_IFS   VARCHAR(500) NOT NULL,
    UKURAN_BYTE BIGINT       NOT NULL,
    CHECKSUM    VARCHAR(64)  NOT NULL,
    WAKTU_UNGGAH TIMESTAMP   NOT NULL,
    DIUNGGAH_OLEH VARCHAR(30) NOT NULL,
    STATUS      CHAR(1)      NOT NULL DEFAULT 'A',
    PRIMARY KEY (ID_DOK)
);

CREATE INDEX DTALIB.DOKUMEN_IX1
    ON DTALIB.DOKUMEN (ID_NASABAH, JENIS_DOK);
```

Kolom `CHECKSUM` adalah yang membuat pola ini bisa diaudit. Ia memungkinkan verifikasi berkala bahwa berkas di IFS masih utuh dan belum diubah di luar aplikasi.

```
-- Dokumen yang catatannya ada tetapi berkasnya hilang
SELECT d.ID_DOK, d.NAMA_BERKAS, d.JALUR_IFS, d.WAKTU_UNGGAH
  FROM DTALIB.DOKUMEN d
 WHERE d.STATUS = 'A'
   AND NOT EXISTS (
       SELECT 1 FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
                      START_PATH_NAME => d.JALUR_IFS)) x);
```

Kueri ini layak dijalankan bulanan. Catatan dokumen yang berkasnya hilang adalah masalah kepatuhan yang lebih baik ditemukan sendiri daripada oleh auditor.

179.3 Kinerja dan kapasitas

Tabel dengan kolom LOB tumbuh jauh lebih cepat daripada perkiraan berdasarkan jumlah baris, dan itu memengaruhi banyak hal.

Dampak	Penjelasan
Durasi backup	Naik sebanding ukuran total
Durasi reorganisasi	Naik tajam
Journaling	Isi LOB ikut dijurnal bila dikonfigurasi demikian
Replikasi	Lalu lintas jaringan naik
Kueri yang tidak memakai kolom LOB	Umumnya tidak terpengaruh

Baris ketiga layak diperhatikan. Journaling pada tabel dengan kolom LOB besar bisa menghasilkan journal receiver yang tumbuh sangat cepat. Db2 for i menyediakan pilihan untuk tidak menjurnal isi LOB, dan untuk sebagian kasus itu pilihan yang tepat — dengan konsekuensi bahwa pemulihan ke titik waktu tidak mencakup isi LOB.

Keputusan itu harus diambil sadar dan didokumentasikan, karena akibatnya baru terasa saat pemulihan dibutuhkan.

👥 Bab 180. Penulisan Ulang Kueri oleh Optimizer

180.1 Apa yang dilakukan optimizer sebelum memilih jalur

Sebelum memilih jalur akses, optimizer menulis ulang kueri ke bentuk yang setara secara hasil tetapi lebih mudah dioptimasi. Memahami transformasi yang lazim membantu menjelaskan mengapa kueri tertentu berperilaku tidak seperti yang diharapkan.

Transformasi	Contoh
Predikat didorong ke bawah	Kondisi dipindahkan ke dalam subquery
Subquery diubah menjadi join	`IN (SELECT ...)` menjadi join
View digabung ke kueri induk	Definisi view disisipkan
Predikat transitif ditambahkan	Dari `a=b` dan `b=5` disimpulkan `a=5`
Ekspresi konstan dihitung di muka	`WHERE x = 2+3` menjadi `WHERE x = 5`
`OR` diubah menjadi `UNION`	Bila menguntungkan

180.2 Pola penulisan yang menghalangi optimasi

Beberapa bentuk penulisan mencegah optimizer memakai index, dan menghindarinya adalah cara termurah

memperbaiki kinerja.

```
-- Buruk: fungsi pada kolom mencegah pemakaian index
SELECT * FROM DTALIB.TRXHIST
 WHERE YEAR(TGL_BUKU) = 2026 AND MONTH(TGL_BUKU) = 9;

-- Baik: rentang pada kolom, index dapat dipakai
SELECT * FROM DTALIB.TRXHIST
 WHERE TGL_BUKU BETWEEN '2026-09-01' AND '2026-09-30';
```

```
-- Buruk: konversi tipe implisit pada kolom
SELECT * FROM DTALIB.REKENING WHERE NO_REK = 1234567890123456;

-- Baik: tipe cocok dengan definisi kolom
SELECT * FROM DTALIB.REKENING WHERE NO_REK = '1234567890123456';
```

```
-- Buruk: wildcard di depan
SELECT * FROM DTALIB.CIF_UTAMA WHERE NAMA LIKE '%SANTOSO%';

-- Baik: wildcard hanya di belakang
SELECT * FROM DTALIB.CIF_UTAMA WHERE NAMA_CARI LIKE 'SANTOSO%';
```

```
-- Buruk: OR pada kolom berbeda sering menghalangi index
SELECT * FROM DTALIB.REKENING
 WHERE NO_REK = :p1 OR ID_NASABAH = :p2;

-- Baik: dipecah menjadi dua kueri yang digabung
SELECT * FROM DTALIB.REKENING WHERE NO_REK = :p1
UNION
SELECT * FROM DTALIB.REKENING WHERE ID_NASABAH = :p2;
```

Tentang contoh terakhir. Optimizer modern sering melakukan transformasi ini sendiri, sehingga penulisan manual tidak selalu dibutuhkan. Tetapi ketika kinerja bermasalah dan pesan debug menunjukkan table scan pada kueri dengan `OR`, menulis ulangnya sebagai `UNION` adalah percobaan yang murah dan sering berhasil.

180.3 Statistik dan pengaruhnya pada keputusan

Optimizer mengambil keputusan berdasarkan perkiraan jumlah baris yang akan dihasilkan setiap langkah. Perkiraan itu berasal dari statistik kolom, dan statistik yang usang menghasilkan keputusan yang buruk.

```
-- Statistik dan kapan terakhir diperbarui
SELECT TABLE_NAME, COLUMN_NAMES,
       NUMBER_DISTINCT_VALUES,
       LAST_REFRESH_TIME,
       DAYS(CURRENT DATE) - DAYS(DATE(LAST_REFRESH_TIME)) AS umur_hari,
       STATISTIC_TYPE
  FROM QSYS2.SYSCOLUMNSTAT
 WHERE TABLE_SCHEMA = 'DTALIB'
   AND LAST_REFRESH_TIME < CURRENT TIMESTAMP - 30 DAYS
 ORDER BY umur_hari DESC;
```

Kapan statistik perlu diperbarui secara sengaja:

1. Setelah pemuatan data besar

2. Setelah reorganisasi tabel
3. Setelah migrasi data
4. Setelah perubahan struktur tabel
5. Ketika kinerja kueri berubah tanpa perubahan kode

Butir kelima adalah gejala yang paling sering salah didiagnosis. Sebuah kueri yang tiba-tiba melambat tanpa ada perubahan apa pun pada kodenya sering disebabkan oleh perubahan distribusi data yang membuat statistik lama menyesatkan.

```
-- Memaksa pembaruan statistik
CALL QSYS2.ENSURE_STATISTICS('DTALIB', 'TRXHIST');
```

180.4 Membaca rencana eksekusi

Visual Explain di Access Client Solutions menampilkan rencana eksekusi secara grafis, dan itu cara termudah memahaminya. Untuk analisis terprogram, plan cache menyediakan datanya.

Yang dicari saat membaca rencana:

Yang dilihat	Arti bila buruk
Metode akses tiap tabel	Table scan pada tabel besar
Urutan join	Tabel besar diproses lebih dulu
Metode join	Nested loop pada hasil besar
Perkiraan baris versus aktual	Selisih besar berarti statistik menyesatkan
Operasi sort	Sort pada jutaan baris
Temporary result	File sementara besar

Baris keempat adalah yang paling informatif dan paling sering diabaikan. Ketika optimizer memperkirakan seribu baris dan hasil sebenarnya dua juta, seluruh keputusan setelahnya dibangun di atas perkiraan yang salah — dan perbaikannya bukan mengubah kueri, melainkan memperbaiki statistik.

Bab 181. Kendali Sumber Daya Kueri

181.1 Membatasi dampak kueri yang boros

Di lingkungan yang membolehkan kueri ad hoc untuk pelaporan, satu kueri yang ditulis sembarangan bisa mengganggu seluruh sistem. Db2 for i menyediakan beberapa mekanisme pembatas.

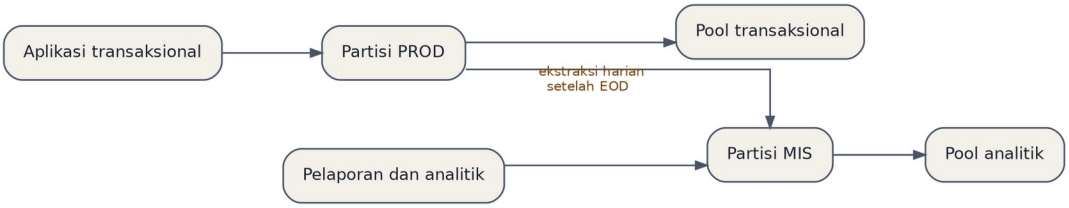
```
-- Batas waktu perkiraan: kueri yang diperkirakan lebih lama ditolak
INSERT INTO QUSRSYS.QAQQINI (QQPARM, QQVAL, QQTEXT)
VALUES ('QUERY_TIME_LIMIT', '300',
        'Tolak kueri yang diperkirakan lebih dari 5 menit');

-- Batas penyimpanan sementara
INSERT INTO QUSRSYS.QAQQINI (QQPARM, QQVAL, QQTEXT)
VALUES ('STORAGE_LIMIT', '5000',
        'Batas 5 GB penyimpanan sementara per kueri');

-- Cegah pembangunan index sementara pada tabel besar
INSERT INTO QUSRSYS.QAQQINI (QQPARM, QQVAL, QQTEXT)
VALUES ('ALLOW_TEMPORARY_INDEXES', '*ONLY_REQUIRED',
        'Hanya bangun index sementara bila benar-benar perlu');
```

Tentang `QUERY_TIME_LIMIT`. Ia bekerja berdasarkan perkiraan optimizer, bukan waktu sebenarnya. Kueri yang diperkirakan cepat tetapi ternyata lambat tidak akan dihentikan olehnya. Ia efektif menangkap kueri yang jelas boros — hasil kali silang, misalnya — dan tidak efektif menangkap kueri yang perkiraannya keliru.

Untuk perlindungan yang lebih menyeluruh, kombinasikan dengan pembatasan pada tingkat job: prioritas rendah dan pool memori terpisah untuk beban pelaporan.

181.2 Memisahkan beban pelaporan

Pendekatan yang paling efektif bukan membatasi kueri, melainkan memindahkannya.



Gambar 113

Pemisahan ini punya biaya: perangkat keras tambahan, proses ekstraksi yang harus dipelihara, dan data yang tidak waktu nyata. Tetapi ia menghilangkan seluruh kelas masalah di mana laporan mengganggu transaksi, dan di lingkungan perbankan itu hampir selalu sepadan.

Bila pemisahan penuh belum memungkinkan, langkah antara yang berguna: batasi jam akses ODBC untuk pelaporan lewat exit program, sehingga kueri berat hanya bisa berjalan di luar jam layanan.

Bagian LIII — Pendalaman Identitas dan Akses

Bab 182. Mengelola Identitas dalam Skala Besar

182.1 Masalah yang muncul pada skala

Sebuah bank dengan tiga ratus cabang punya ribuan profil pengguna. Pada skala itu, pendekatan yang bekerja untuk lima puluh pengguna mulai gagal.

Masalah	Gejala
Wewenang diberikan per individu	Tidak ada yang tahu siapa boleh apa
Profil tidak dicabut saat orang pindah	Wewenang menumpuk sepanjang karier
Profil bersama untuk satu fungsi	Jejak audit tidak menunjuk orang
Penamaan tidak konsisten	Sulit mengaitkan profil ke orang
Pencabutan tidak terhubung ke SDM	Orang keluar, profil tetap aktif

Masalah kedua punya nama di literatur kontrol akses: penumpukan hak. Seorang staf yang berpindah dari operasi ke pengembangan, lalu ke dukungan aplikasi, mengumpulkan wewenang dari ketiga peran karena tidak ada yang mencabut yang lama.

182.2 Model berbasis peran

Jawabannya adalah memberikan wewenang kepada peran, bukan kepada orang, lalu menempatkan orang ke dalam peran.

```
CREATE TABLE SECLIB.PERAN (
    KD_PERAN    VARCHAR(20)  NOT NULL,
    NAMA_PERAN  VARCHAR(80)  NOT NULL,
    GROUP_PRF   VARCHAR(10)  NOT NULL,
    PEMILIK     VARCHAR(50)  NOT NULL,
    KETERANGAN  VARCHAR(300),
    PRIMARY KEY (KD_PERAN)
);

CREATE TABLE SECLIB.PENUGASAN (
    ID          BIGINT GENERATED ALWAYS AS IDENTITY,
    USERID      VARCHAR(10)  NOT NULL,
    KD_PERAN    VARCHAR(20)  NOT NULL,
    MULAI       DATE         NOT NULL,
    SELESAI     DATE,
    TIKET       VARCHAR(20)  NOT NULL,
    DISETUJUI   VARCHAR(50)  NOT NULL,
    PRIMARY KEY (ID)
);

CREATE INDEX SECLIB.PENUGASAN_IX1 ON SECLIB.PENUGASAN (USERID, SELESAI);
```

Kolom `PEMILIK` pada tabel peran adalah unsur yang membuat model ini berkelanjutan. Setiap peran punya pemilik dari sisi bisnis yang bertanggung jawab meninjau siapa saja yang ada di dalamnya. Tanpa pemilik, tinjauan akses menjadi tugas tim TI yang tidak punya dasar untuk menilai apakah seseorang memang membutuhkan peran itu.

182.3 Rekonsiliasi antara niat dan kenyataan

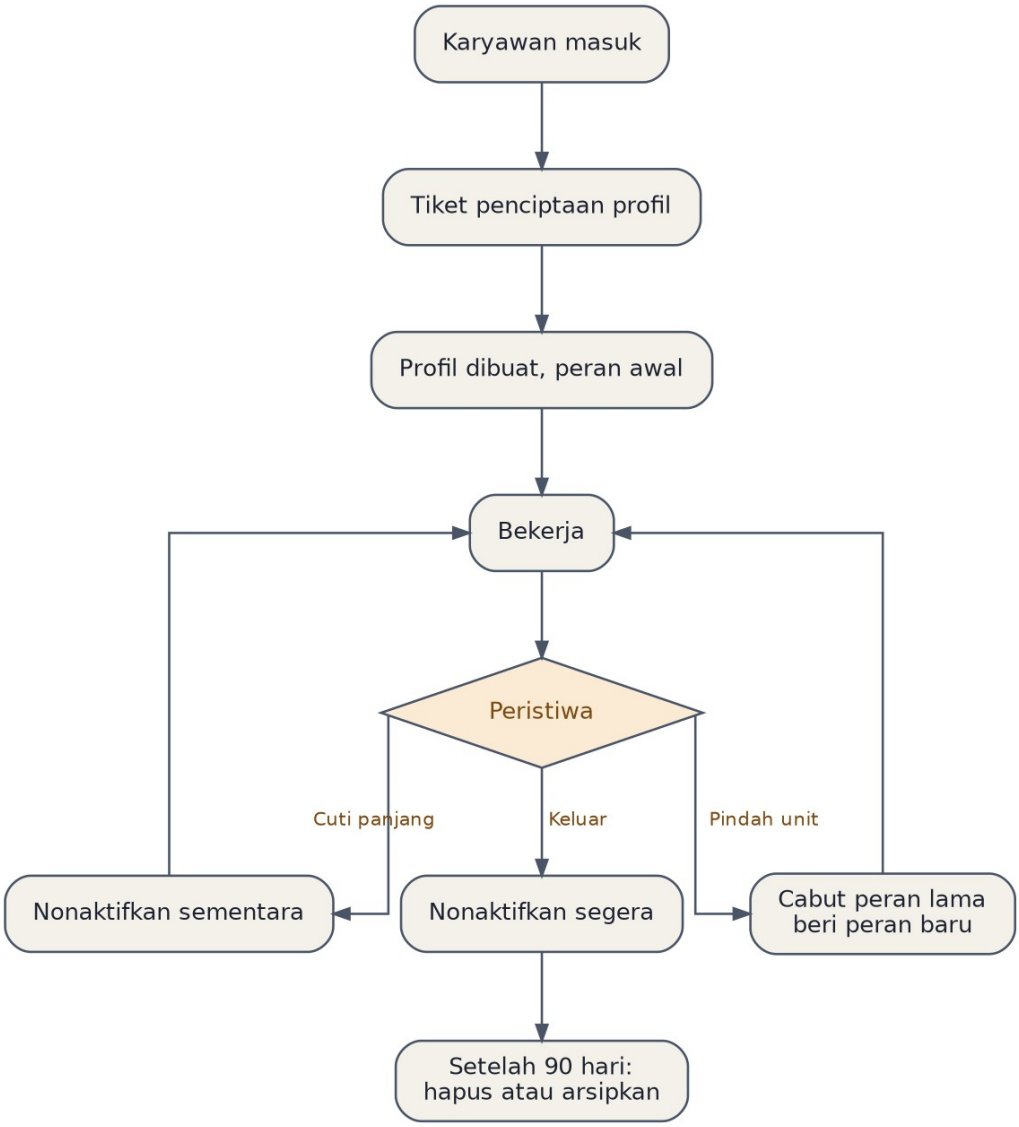
Tabel di atas menyatakan siapa seharusnya punya apa. Sistem menyatakan siapa sebenarnya punya apa. Keduanya harus dibandingkan secara berkala.

```
-- Profil yang ada di sistem tetapi tidak punya penugasan aktif
SELECT u.AUTHORIZATION_NAME, u.STATUS, u.GROUP_PROFILE_NAME,
       u.PREVIOUS_SIGNON, u.TEXT_DESCRIPTION
  FROM QSYS2.USER_INFO u
 WHERE u.STATUS = '*ENABLED'
   AND u.AUTHORIZATION_NAME NOT LIKE 'Q%'
   AND NOT EXISTS (
       SELECT 1 FROM SECLIB.PENUGASAN p
        WHERE p.USERID = u.AUTHORIZATION_NAME
          AND p.MULAI <= CURRENT DATE
          AND (p.SELESAI IS NULL OR p.SELESAI >= CURRENT DATE))
 ORDER BY u.PREVIOUS_SIGNON;

-- Penugasan yang tercatat tetapi group profile-nya tidak sesuai
SELECT p.USERID, p.KD_PERAN, r.GROUP_PRF AS seharusnya,
       u.GROUP_PROFILE_NAME AS kenyataan
  FROM SECLIB.PENUGASAN p
  JOIN SECLIB.PERAN r ON r.KD_PERAN = p.KD_PERAN
  JOIN QSYS2.USER_INFO u ON u.AUTHORIZATION_NAME = p.USERID
 WHERE (p.SELESAI IS NULL OR p.SELESAI >= CURRENT DATE)
   AND u.GROUP_PROFILE_NAME <> r.GROUP_PRF;
```

Kedua kueri ini layak dijadwalkan mingguan. Kueri pertama menemukan profil yatim — aktif tanpa dasar; kueri kedua menemukan penyimpangan antara yang disetujui dan yang diterapkan.

182.4 Siklus hidup profil yang terhubung ke SDM

Idealnya, penciptaan dan pencabutan profil dipicu oleh peristiwa kepegawaian, bukan oleh permintaan terpisah.



Gambar 114

Kotak bertanda pindah unit adalah yang paling sering gagal dalam praktik. Pemberian peran baru selalu terjadi karena orang itu membutuhkannya untuk bekerja; pencabutan peran lama sering tidak terjadi karena tidak ada yang merasakan akibatnya.

Cara yang efektif menutupnya: buat penugasan peran punya masa berlaku, dan wajibkan perpanjangan berkala. Peran yang tidak diperpanjang akan berakhir sendiri. Ini memindahkan beban dari mengingat mencabut menjadi mengingat memperpanjang, dan yang kedua jauh lebih mungkin terjadi karena yang

bersangkutan akan mengeluh bila aksesnya hilang.

182.5 Profil bersama dan mengapa ia bermasalah

Profil yang dipakai bersama oleh beberapa orang — misalnya satu profil untuk seluruh teller di satu cabang — masih ditemukan di banyak institusi.

Alasannya biasanya praktis: lisensi, kemudahan, atau warisan dari sistem lama. Tetapi konsekuensinya serius.

Konsekuensi	Penjelasan
Jejak audit tidak menunjuk orang	Transaksi tercatat atas nama profil
Pencabutan tidak selektif	Mencabut satu orang berarti mencabut semua
Kata sandi tidak bisa dijaga rahasia	Diketahui banyak orang
Temuan audit hampir pasti	Melanggar prinsip akuntabilitas individu

Bila profil bersama tidak bisa dihapus dalam waktu dekat, mitigasi yang bisa diterapkan: catat identitas pengguna sebenarnya pada tingkat aplikasi, pastikan catatan itu masuk ke tabel transaksi, dan buat laporan yang memetakan transaksi ke orang. Ini tidak menyelesaikan masalah, tetapi ia menyediakan jejak yang bisa dipertanggungjawabkan sambil menunggu perbaikan struktural.

Bab 183. Single Sign-On dan Kerberos

183.1 Mengapa ini dipertimbangkan

Di lingkungan dengan banyak sistem, pengguna memelihara kata sandi terpisah untuk masing-masing. Akibatnya bisa diduga: kata sandi yang sama dipakai di mana-mana, ditulis di tempat yang tidak aman, atau di-reset berkali-kali.

Single sign-on memindahkan autentikasi ke satu tempat. Di lingkungan yang memakai direktori terpusat, IBM i dapat ikut serta lewat Kerberos dan Enterprise Identity Mapping.

183.2 Komponen yang terlibat

Komponen	Peran
Kerberos KDC	Menerbitkan tiket autentikasi
Network Authentication Service	Dukungan Kerberos di IBM i
EIM	Memetakan identitas antar sistem
Direktori LDAP	Menyimpan pemetaan EIM
Profil IBM i	Tetap ada, tetapi tanpa kata sandi

Yang perlu dipahami: profil IBM i tetap dibutuhkan. Kerberos mengurus pembuktian identitas; wewenang objek tetap ditentukan oleh profil dan group profile. SSO tidak menggantikan model keamanan IBM i, ia hanya mengganti cara membuktikan siapa Anda.

183.3 Pertimbangan sebelum menerapkan

Penerapan SSO adalah proyek yang lebih besar daripada yang biasanya diperkirakan. Beberapa hal yang perlu dipastikan lebih dulu:

1. Waktu sistem tersinkronisasi dengan ketat, karena Kerberos sensitif terhadap selisih waktu
2. Nama host dan DNS terdaftar konsisten di seluruh sistem
3. Ada jalur masuk cadangan bila KDC tidak tersedia
4. Aplikasi yang memakai koneksi ODBC mendukung autentikasi Kerberos
5. Prosedur untuk akun layanan yang tidak bisa memakai SSO

Butir ketiga sering diabaikan dan paling berisiko. Bila seluruh autentikasi bergantung pada KDC dan KDC tidak tersedia, tidak ada seorang pun yang bisa masuk ke sistem — termasuk untuk memperbaikinya. Pertahankan sedikitnya satu profil dengan kata sandi yang disimpan dalam brankas, sebagai jalur darurat.

Saran tentang urutan penerapan. Mulai dari kelompok pengguna yang paling sederhana — staf kantor pusat dengan satu aplikasi, misalnya — bukan dari teller cabang yang memakai beberapa sistem sekaligus. Pengalaman dari kelompok pertama akan mengungkap hal-hal yang tidak muncul dalam perencanaan.

Bab 184. Manajemen Kata Sandi dan Kredensial

184.1 Kebijakan kata sandi yang masuk akal

System value yang mengendalikan kebijakan kata sandi cukup banyak, dan menyetelnya terlalu ketat sering menghasilkan perilaku yang justru kurang aman.

System value	Fungsi	Pertimbangan
`QPWDLVL`	Tingkat kompleksitas yang didukung	Level 3 atau 4 untuk kata sandi panjang
`QPWDMINLEN`	Panjang minimum	Panjang lebih penting daripada kerumitan
`QPWDMAXLEN`	Panjang maksimum	Setinggi mungkin pada level yang dipakai
`QPWDEXPITV`	Masa berlaku	Terlalu pendek mendorong pola yang mudah ditebak
`QPWDRQDDIF`	Berapa kata sandi lama tidak boleh diulang	Minimal 10
`QPWDRULES`	Aturan komposisi	Gantikan system value lama yang terpisah
`QMAXSIGN`	Percobaan gagal sebelum dikunci	3 sampai 5
`QMAXSGNACN`	Tindakan saat batas tercapai	Nonaktifkan profil

Tentang masa berlaku. Pemaksaan penggantian kata sandi setiap tiga puluh hari mendorong orang memakai pola berurutan yang mudah ditebak. Panduan keamanan modern cenderung menganjurkan masa berlaku lebih panjang dengan kata sandi yang lebih panjang, ditambah pemantauan atas indikasi kompromi. Sesuaikan dengan ketentuan yang berlaku bagi institusi Anda, karena sebagian regulator masih menetapkan angka tertentu.

184.2 Kredensial untuk akun layanan

Akun yang dipakai aplikasi, bukan manusia, punya kebutuhan berbeda dan sering ditangani dengan buruk.

Masalah yang lazim:

Masalah	Akibat
Kata sandi ditulis di kode sumber	Tersebar ke repositori dan backup
Kata sandi tidak pernah diganti	Berlaku bertahun-tahun
Satu akun untuk banyak aplikasi	Pencabutan memutus semuanya
Wewenang berlebih karena praktis	Melanggar prinsip hak minimum
Tidak ada yang tahu siapa pemiliknya	Tidak bisa ditinjau

Pendekatan yang lebih baik:

```
CREATE TABLE SECLIB.AKUN_LAYANAN (
    USERID       VARCHAR(10)  NOT NULL,
    APLIKASI     VARCHAR(50)  NOT NULL,
    PEMILIK      VARCHAR(50)  NOT NULL,
    TUJUAN       VARCHAR(300) NOT NULL,
    WEWENANG     VARCHAR(300) NOT NULL,
    ROTASI_HARI  SMALLINT     NOT NULL DEFAULT 180,
    ROTASI_TERAKHIR DATE,
    PRIMARY KEY (USERID)
);

-- Akun layanan yang melewati jadwal rotasi
SELECT USERID, APLIKASI, PEMILIK, ROTASI_TERAKHIR,
       DAYS(CURRENT DATE) - DAYS(ROTASI_TERAKHIR) AS umur_hari,
       ROTASI_HARI
  FROM SECLIB.AKUN_LAYANAN
 WHERE ROTASI_TERAKHIR IS NULL
    OR DAYS(CURRENT DATE) - DAYS(ROTASI_TERAKHIR) > ROTASI_HARI
 ORDER BY umur_hari DESC;
```

Kolom `TUJUAN` dan `PEMILIK` adalah yang membuat akun layanan bisa ditinjau. Sebuah akun tanpa keduanya tidak bisa dinilai apakah masih dibutuhkan, dan akibatnya ia akan terus ada selamanya.

184.3 Menghilangkan kredensial dari kode

Pola yang bisa diterapkan tanpa perangkat tambahan:

1. Kredensial disimpan di objek yang wewenangnya ketat, bukan di kode
2. Program membaca kredensial saat dijalankan, bukan menyimpannya
3. Yang membaca adalah program yang mengadopsi wewenang, bukan pemanggilnya
4. Setiap pembacaan tercatat

```
/* Objek penyimpan dengan wewenang sangat terbatas */
CRTDTAARA DTAARA(SECLIB/KRED001) TYPE(*CHAR) LEN(128)
CHGOBJOWN OBJ(SECLIB/KRED001) OBJTYPE(*DTAARA) NEWOWN(SECOWNER)
GRTOBJAUT OBJ(SECLIB/KRED001) OBJTYPE(*DTAARA) +
          USER(*PUBLIC) AUT(*EXCLUDE)

/* Program pembaca mengadopsi wewenang pemiliknya */
CHGPGM PGM(SECLIB/AMBILKRED) USRPRF(*OWNER) USEADPAUT(*NO)
```

Ini bukan pengganti dari perangkat pengelola rahasia yang sesungguhnya, dan bila institusi Anda punya perangkat itu, pakailah. Tetapi bagi institusi yang belum punya, pola ini jauh lebih baik daripada kata sandi yang tertulis di kode sumber.

Bab 185. Tinjauan Akses Berkala

185.1 Bentuk tinjauan yang bekerja

Tinjauan akses yang dilakukan dengan mengirimkan daftar panjang kepada manajer dan meminta persetujuan hampir selalu menghasilkan persetujuan tanpa pembacaan. Beberapa hal membuatnya lebih efektif.

Praktik	Alasan
Kirim per peran, bukan per orang	Pemilik peran tahu konteksnya
Sertakan data pemakaian nyata	Peran yang tidak dipakai mudah dicabut
Minta keputusan eksplisit per baris	Bukan persetujuan menyeluruh
Batasi panjang daftar per tinjauan	Daftar panjang tidak dibaca
Terapkan hasilnya dengan cepat	Tinjauan yang tidak berdampak akan diabaikan

Butir kedua adalah yang paling mengubah hasil. Ketika seorang pemilik peran melihat bahwa seseorang tidak pernah memakai wewenang itu selama sembilan puluh hari, keputusan mencabutnya menjadi mudah.

```
-- Pemakaian nyata per pengguna, dari audit journal
WITH pakai AS (
  SELECT USER_NAME, COUNT(*) AS jml,
         MAX(ENTRY_TIMESTAMP) AS terakhir
    FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
           STARTING_TIMESTAMP => CURRENT TIMESTAMP - 90 DAYS))
   GROUP BY USER_NAME
)
SELECT p.USERID, p.KD_PERAN,
       COALESCE(k.jml, 0) AS aktivitas_90hari,
       k.terakhir,
       CASE WHEN COALESCE(k.jml, 0) = 0 THEN 'CABUT'
            WHEN k.jml < 10 THEN 'TINJAU'
            ELSE 'PERTAHANKAN' END AS saran
  FROM SECLIB.PENUGASAN p
  LEFT JOIN pakai k ON k.USER_NAME = p.USERID
 WHERE p.SELESAI IS NULL OR p.SELESAI >= CURRENT DATE
 ORDER BY aktivitas_90hari;
```

Kolom saran membuat tinjauan menjadi verifikasi atas rekomendasi, bukan penilaian dari nol. Itu perbedaan yang besar dalam hal berapa banyak baris yang benar-benar dipikirkan.

185.2 Menangani hasil tinjauan

Tinjauan menghasilkan tiga jenis keputusan, dan masing-masing perlu penanganan berbeda.

Pertahankan. Tidak ada tindakan, tetapi catat bahwa tinjauan sudah dilakukan beserta tanggalnya. Bukti bahwa tinjauan berjalan adalah hal yang diminta auditor.

Cabut. Terapkan dalam waktu yang disepakati, biasanya lima hari kerja. Pantau kegagalan wewenang selama dua pekan setelahnya, karena kadang ada ketergantungan yang tidak terlihat di atas kertas.

Tinjau lebih lanjut. Ini kategori yang paling sering menjadi lubang. Tetapkan tenggat dan pemilik; tanpa keduanya, baris ini akan muncul lagi pada tinjauan berikutnya dengan status yang sama.

185.3 Menjadikan tinjauan tidak menyakitkan

Tinjauan akses punya reputasi buruk karena biasanya dilakukan sebagai kegiatan besar sekali setahun yang mengganggu semua orang.

Alternatif yang lebih baik: bagi menjadi tinjauan kecil yang berulang. Satu peran per pekan, bergilir, sehingga setiap peran ditinjau dua sampai empat kali setahun dan tidak ada momen yang terasa berat.

Pendekatan ini juga menghasilkan data yang lebih baik: peran yang baru saja ditinjau tiga bulan lalu punya daftar perubahan yang pendek dan mudah dinilai, sementara daftar setahun penuh terlalu banyak untuk dicerna.

📖 Lampiran

📖 Lampiran A. Perintah CL Esensial

Perintah	Fungsi	Catatan
`WRKACTJOB`	Job aktif	F10 untuk reset statistik
`WRKSYSSTS`	Status sistem, pool, faulting	F10 reset sebelum mengukur
`WRKDSKSTS`	Status disk	Perhatikan % busy dan % used
`WRKSBS`	Subsystem aktif	
`WRKJOBQ`	Job queue	Cek yang tertahan
`WRKOUTQ`	Output queue	
`DSPJOBLOG`	Job log	Pakai `OUTPUT(*PRINT)` untuk arsip
`WRKOBJLCK`	Siapa mengunci objek	Wajib saat LCKW
`DSPMSG QSYSOPR`	Pesan operator	Periksa tiap pagi
`WRKPTFGRP`	Level PTF group	
`DSPSYSVAL`	Nilai system value	
`WRKUSRPRF`	Profil pengguna	
`DSPOBJAUT`	Wewenang objek	
`WRKJRNA`	Atribut journal	Cek ukuran receiver
`DSPJRN`	Isi journal	Untuk investigasi perubahan data
`STRDBG` / `ENDDBG`	Sesi debug	Selalu `UPDPROD(*NO)` di PROD
`STRSRVJOB` / `ENDSRVJOB`	Layani job lain	Wajib dipasangkan
`SBMJOB`	Kirim job batch	
`WRKSPLF`	Spooled file	
`RGZPFM`	Reorganisasi tabel	`ALWCANCEL(*YES)`
`APYJRNCHG`	Terapkan perubahan journal	Perhatikan `TOTIME`
`RCLSTG`	Klaim ruang hilang	Butuh restricted state

📖 Lampiran B. System Values Kritikal

System value	Nilai untuk perbankan	Butuh IPL?
`QSECURITY`	40 atau 50	Ya
`QMAXSIGN`	3	Tidak
`QMAXSGNACN`	3	Tidak
`QINACTITV`	15	Tidak
`QINACTMSGQ`	`*DSCJOB`	Tidak
`QPWDLVL`	3 atau 4	Ya
`QPWDEXPITV`	90	Tidak
`QPWDRQDDIF`	5	Tidak
`QAUDCTL`	`*AUDLVL *OBJAUD`	Tidak
`QAUDLVL`	`*AUTFAIL *SECURITY *CREATE *DELETE *SAVRST`	Tidak
`QALWOBJRST`	`*NONE`	Tidak
`QCRTAUT`	`*EXCLUDE`	Tidak
`QLMTSECOFR`	1	Tidak
`QRMTSIGN`	`*VERIFY` atau `*REJECT`	Tidak
`QDSPSGNINF`	1	Tidak

📕 Lampiran C. Templat Runbook

```
RUNBOOK-[nomor]: [judul singkat]

PEMICU
  Message ID   :
  Gejala       :
  Terdeteksi   : [watch / monitoring / laporan pengguna]

DAMPAK
  Layanan      :
  Terdampak    :
  Tingkat      : [kritis / tinggi / sedang / rendah]

PRASYARAT
  Wewenang     :
  Akses        :
  Alat         :

DIAGNOSIS
  1. Perintah  :
     Harapan   :
     Jika beda :
  2. ...

PERBAIKAN
  1. Perintah  :
     PERINGATAN:
  2. ...

VERIFIKASI
  1.
  2.
```

```
ESKALASI
  Tingkat 1    : [nama, nomor, batas waktu]
  Tingkat 2    :
  Vendor       : [nomor kontrak, nomor kontak]

RIWAYAT
  [tanggal] [nama] [perubahan]
```

📖 Lampiran D. Checklist Harian System Engineer

Pagi, sebelum jam layanan

- ☐ Tanggal buku aplikasi sama dengan tanggal sistem
- ☐ Seluruh fase EOD berstatus selesai
- ☐ Tidak ada job berstatus MSGW
- ☐ Tidak ada job queue tertahan
- ☐ Pemakaian ASP di bawah ambang perhatian
- ☐ Backup semalam selesai dan terverifikasi
- ☐ Pesan QSYSOPR sudah ditinjau
- ☐ Replikasi ke DR tidak tertinggal
- ☐ Seluruh subsystem produksi aktif

Sore, sebelum cut-off

- ☐ Profil memori malam terjadwal
- ☐ Tidak ada sesi interaktif menahan record
- ☐ Media backup tersedia
- ☐ Tidak ada change yang belum selesai

Mingguan

- ☐ Tinjau tren kapasitas
- ☐ Tinjau durasi tiap fase EOD dibanding baseline
- ☐ Periksa PTF group yang tertinggal
- ☐ Audit profil dengan wewenang khusus
- ☐ Bersihkan spooled file melewati retensi

Bulanan

- ☐ Ambil baseline konfigurasi
- ☐ Uji restore satu objek acak
- ☐ Tinjau journal receiver dan arsipkan
- ☐ Perbarui proyeksi kapasitas

📖 Lampiran E. Checklist EOD

Sebelum EOD

- ☐ Cut-off terkonfirmasi, kanal ditutup

- ☐ Tidak ada sesi interaktif tersisa
- ☐ Checkpoint A selesai dan terverifikasi
- ☐ Media backup memadai
- ☐ Tabel `EOD_KONTROL` diinisialisasi
- ☐ Personel jaga terkonfirmasi

Selama EOD

- ☐ Pantau progres tiap fase
- ☐ Bandingkan durasi dengan baseline
- ☐ Pantau job MSGW dan LCKW
- ☐ Pantau pemakaian ASP

Sesudah EOD

- ☐ Seluruh fase berstatus selesai
- ☐ GL seimbang
- ☐ Date roll berhasil
- ☐ Checkpoint B selesai
- ☐ Laporan regulator terbentuk
- ☐ Ekstraksi ke MIS terkirim
- ☐ Profil memori siang aktif
- ☐ Kanal dibuka

📖 Lampiran F. Glosarium

Istilah	Arti
ASP	Auxiliary Storage Pool, kelompok disk
iASP	Independent ASP, bisa dipindah antar sistem
BRMS	Backup Recovery and Media Services
CPW	Commercial Processing Workload, satuan kapasitas
DDS	Data Description Specification, pendefinisian file gaya lama
EOD	End of Day, proses tutup hari
IFS	Integrated File System
ILE	Integrated Language Environment
IWS	Integrated Web Services
LPAR	Logical Partition
PASE	Portable Application Solutions Environment, runtime AIX
PTF	Program Temporary Fix
RCAC	Row and Column Access Control
SIBS	Silverlake Axis Integrated Banking Solution
SLIC	System Licensed Internal Code
TIMI	Technology Independent Machine Interface
TR	Technology Refresh

Istilah	Arti
VIOS	Virtual I/O Server

Daftar Pustaka

Standar dan kerangka kerja

- INCOSE. *INCOSE Systems Engineering Handbook*, 5th ed. Hoboken: Wiley, 2023. ISBN 978-1119814290
- ISO/IEC/IEEE 15288:2023, *Systems and Software Engineering — System Life Cycle Processes*
- ISO/IEC/IEEE 42010:2022, *Software, Systems and Enterprise — Architecture Description*
- ISO/IEC 27001:2022, *Information Security Management Systems*
- ISO 22301:2019, *Business Continuity Management Systems*
- AXELOS. *ITIL Foundation: ITIL 4 Edition*. London: TSO, 2019
- ISACA. *COBIT 2019 Framework: Governance and Management Objectives*. 2018
- NIST SP 800-53 Rev. 5, *Security and Privacy Controls for Information Systems and Organizations*. 2020
- Basel Committee on Banking Supervision. *Principles for Operational Resilience*. 2021

🏗️ Arsitektur platform

- Soltis, Frank G. *Inside the AS/400*, 2nd ed. Loveland: Duke Press, 1997
- Soltis, Frank G. *Fortress Rochester: The Inside Story of the IBM iSeries*. Loveland: 29th Street Press, 2001

💻 Pemrograman

- Meyers, Bryan dan Buck, Jim. *Programming in ILE RPG*, 5th ed. MC Press, 2015. ISBN 978-1583473795
- Buck, Jim; Meyers, Bryan; Riehl, Dan. *Control Language Programming for IBM i*. MC Press
- Forsythe, Kevin. *SQL for IBM i: A Database Modernization Guide*. MC Press

🔒 Keamanan

- Woodbury, Carol. *IBM i Security Administration and Compliance*, edisi terbaru. MC Press
- IBM Redbook, *Security Guide for IBM i* (seri SG24)

Dokumentasi IBM

- IBM. *IBM Documentation — IBM i 7.6*. https://www.ibm.com/docs/en/i
- IBM. *IBM i 7.6 Announcement Letter*, April 2025
- IBM. *ILE Concepts* (SC41-5606)
- IBM. *Db2 for i SQL Reference* dan *IBM i Services*
- IBM Redbooks (seri SG24): *Technical Overview with Technology Refresh Updates*, *Backup and Recovery using BRMS*, *IBM PowerHA SystemMirror for i*, *Modernizing IBM i Applications*

Nomor seri Redbook berubah tiap rilis — verifikasi di redbooks.ibm.com sebelum dicetak.

🏦 Core banking

- Silverlake Axis Ltd. *Annual Report* dan pengumuman korporat, tersedia di SGX dan https://www.silverlakeaxis.com
- Silverlake Axis Ltd. Siaran pers implementasi SIBS Versi 12 di Indonesia, November 2023
- Gartner Peer Insights. *Core Banking Systems — Silverlake Axis*
- Jack Henry & Associates. *SilverLake System* documentation — disertakan hanya sebagai pembanding, untuk mencegah kerancuan penamaan

Regulasi Indonesia

- Peraturan OJK tentang Penyelenggaraan Teknologi Informasi oleh Bank Umum — verifikasi nomor dan status terkini sebelum dijadikan rujukan formal
- Undang-Undang No. 27 Tahun 2022 tentang Pelindungan Data Pribadi
- Ketentuan Bank Indonesia terkait BI-FAST, SKNBI, dan BI-RTGS

Berkala dan komunitas

- MC Press Online — https://www.mcpressonline.com
- IT Jungle — https://www.itjungle.com
- COMMON dan COMMON Europe — prosiding konferensi tahunan
- RPGPGM.com — blog teknis IBM i

Catatan penyuntingan

Naskah ini adalah draf kerja, bukan versi siap cetak. Sebelum dipublikasikan, tiga hal perlu diselesaikan:

1. **Tinjauan legal dan compliance** atas seluruh contoh di Bagian VI, meskipun sudah teranonimkan.
2. **Verifikasi ulang** setiap nomor Redbook, nomor peraturan OJK, dan detail bibliografi terhadap sumber aslinya.
3. **Uji teknis** setiap potongan kode di lingkungan non-produksi; contoh di sini ditulis untuk kejelasan pedagogis dan mungkin perlu penyesuaian dengan konvensi penamaan institusi Anda.

⚡ Bagian LIV — Pendalaman Analisis Kinerja

Bab 186. Collection Services Secara Mendalam

186.1 Apa yang sebenarnya dikumpulkan

Collection Services mengumpulkan data kinerja secara berkala dan menyimpannya dalam objek koleksi. Data itu kemudian diubah menjadi berkas basis data yang bisa dianalisis.

Yang dikumpulkan dikelompokkan dalam kategori, dan tidak semuanya aktif secara bawaan.

Kategori

Isi

Beban pengumpulan

*SYSLVL

Ringkasan tingkat sistem

Sangat ringan

*JOBMI

Data job tingkat mesin

Ringan

*JOBOS

Data job tingkat sistem operasi

Sedang

*POOL

Memory pool

Sangat ringan

*DISK

Aktivitas disk

Ringan

*COMM

Komunikasi

Ringan

*LCLRSP

Waktu respons lokal

Sedang

*SQL

Aktivitas SQL

Sedang sampai berat

```
/* Memulai dengan profil standar */
STRPFRCOL COLSVCPFN(*STANDARDP) INTERVAL(5) +
          LIB(QMPGDATA) DFTCOLLVL(*CATEGORY)

/* Melihat konfigurasi yang berlaku */
WRKPFRCOL
DSPPFRCOL
```

Tentang interval pengumpulan. Interval lima menit adalah kompromi yang wajar untuk pemantauan rutin. Untuk penyelidikan masalah tertentu, interval satu menit memberi resolusi yang jauh lebih berguna, dengan biaya ruang penyimpanan yang lebih besar.

Pola yang saya sarankan: jalankan pengumpulan rutin pada interval lima menit sepanjang waktu, dan naikkan ke satu menit secara sementara ketika sedang menyelidiki masalah.

186.2 Mengubah koleksi menjadi data yang bisa dianalisis

```
/* Membuat berkas basis data dari
objek koleksi */
CRTPFRDTA
FROMMGTCOL(QMPGDATA/Q262610000) +
          TOLIB(PERFDTA) +
          CATEGORY(*ALL)

/* Melihat koleksi yang tersedia
*/
WRKOBJ OBJ(QMPGDATA/*ALL)
OBJTYPE(*MGTCOL)
```

Setelah diubah, datanya tersedia sebagai tabel yang bisa dikueri. Beberapa tabel yang paling berguna:

Tabel

Isi

QAPMSYSTEM

Ringkasan sistem per interval

QAPMJOBMI

Data job per interval

QAPMJOBOS

Data job tingkat sistem operasi

QAPMPOOL

Memory pool per interval

QAPMDISK

Aktivitas disk per unit

QAPMETH

Interface Ethernet

QAPMJOBWT

Rincian tunggu per job

186.3 Kueri analisis yang berguna

```
-- Pemakaian CPU per interval sepanjang hari
SELECT DTETIM AS interval_waktu,
       DECIMAL(TOTCPU / 1000.0, 9, 2) AS cpu_detik,
       DECIMAL(SYSCPU / 1000.0, 9, 2) AS cpu_sistem,
       JOBACT AS job_aktif
  FROM PERFDTA.QAPMSYSTEM
 ORDER BY DTETIM;

-- Job dengan pemakaian CPU tertinggi sepanjang periode
SELECT JBNAME, JBUSER, JBNBR,
       SUM(JBCPU) / 1000.0 AS total_cpu_detik,
       SUM(JBDBR + JBDBW) AS io_database,
       SUM(JBNDBR + JBNDBW) AS io_nondatabase
  FROM PERFDTA.QAPMJOBMI
 GROUP BY JBNAME, JBUSER, JBNBR
 ORDER BY total_cpu_detik DESC
 FETCH FIRST 25 ROWS ONLY;

-- Faulting per pool sepanjang waktu
SELECT DTETIM, PONBR AS pool,
       PODBFLT AS fault_database,
       PONDBFLT AS fault_nondatabase,
       POSIZE / 1024 AS ukuran_mb,
       POACTLVL AS activity_level
  FROM PERFDTA.QAPMPOOL
 WHERE PODBFLT + PONDBFLT > 100
 ORDER BY DTETIM, PONBR;
```

186.4 Menyusun laporan kinerja periodik

Laporan bulanan yang berguna memuat perbandingan, bukan sekadar angka.

```sql
-- Perbandingan bulan ini dengan bulan lalu pada jam yang sama
WITH bulanan AS (
  SELECT VARCHAR_FORMAT(DTETIM, 'YYYY-MM') AS bulan,
         HOUR(DTETIM) AS jam,
         AVG(TOTCPU / 1000.0) AS rata_cpu,
         MAX(TOTCPU / 1000.0) AS puncak_cpu
    FROM PERFDTA.QAPMSYSTEM
   GROUP BY VARCHAR_FORMAT(DTETIM, 'YYYY-MM'), HOUR(DTETIM)
)
SELECT a.jam,
       DECIMAL(a.rata_cpu, 9, 2) AS cpu_bulan_ini,
       DECIMAL(b.rata_cpu, 9, 2) AS cpu_bulan_lalu,
       DECIMAL((a.rata_cpu - b.rata_cpu) * 100.0
               / NULLIF(b.rata_cpu, 0), 7, 1) AS persen_perubahan
  FROM bulanan a
  JOIN bulanan b ON b.jam = a.jam
 WHERE a.bulan = VARCHAR_FORMAT(CURRENT DATE, 'YYYY-MM')
   AND b.bulan = VARCHAR_FORMAT(CURRENT DATE - 1 MONTH, 'YYYY-MM')
 ORDER BY a.jam;
```

Perbandingan per jam jauh lebih informatif daripada rata-rata harian. Kenaikan beban yang terjadi hanya pada jam sepuluh sampai dua belas menunjuk ke perubahan pola pemakaian; kenaikan merata sepanjang hari menunjuk ke pertumbuhan volume.

Bab 187. Wait Accounting Terapan

187.1 Mengapa kerangka ini paling berguna

Pertanyaan yang biasa diajukan ketika sebuah proses lambat adalah berapa CPU yang dipakainya. Pertanyaan yang lebih berguna adalah ke mana waktunya pergi.

Sebuah job yang berjalan sepuluh menit mungkin hanya memakai CPU selama satu menit. Sembilan menit sisanya dihabiskan menunggu sesuatu, dan penyebab kelambatan ada di sana.

```
flowchart TD
    A[Waktu job berjalan] --> B[CPU aktif]
    A --> C[Menunggu]
    C --> D[Menunggu disk]
    C --> E[Menunggu kunci record]
    C --> F[Menunggu kunci objek]
    C --> G[Menunggu seize]
    C --> H[Menunggu antrean]
    C --> I[Menunggu jurnal]
    C --> J[Menunggu CPU tersedia]
    C --> K[Menunggu sistem lain]
```

187.2 Menafsirkan setiap jenis tunggu

Jenis tunggu

Kemungkinan penyebab

Langkah pertama

Disk page fault

Pool terlalu kecil

Tambah memori pool

Disk operation

I/O berat, index kurang

Periksa jalur akses

Record lock

Kontensi antar job

Tinjau urutan proses

Object lock

Alokasi eksklusif

Cari job penahan

Seize

Kontensi internal

Sering akibat kontensi lain

Journal

Receiver di ASP sibuk

Pindahkan ke ASP terpisah

Ineligible

Activity level terlalu rendah

Naikkan ACTLVL

Socket

Menunggu sistem lain

Periksa mitra dan jaringan

Data queue

Menunggu sinyal

Periksa pengirim

Pola yang paling sering ditemukan di lingkungan core banking:

Tunggu jurnal yang tinggi pada fase posting menunjuk ke journal receiver yang berada di ASP yang sama dengan data. Memindahkannya ke ASP terpisah sering memberi perbaikan yang besar dengan usaha yang kecil.

Tunggu kunci record yang tinggi pada satu tabel tertentu menunjuk ke rekening penampungan atau tabel parameter yang menjadi titik panas. Pola penyelesaiannya ada di Bab 59.

Tunggu ineligible yang tidak nol berarti job siap berjalan tetapi tidak diizinkan karena activity level penuh. Ini masalah konfigurasi, bukan kapasitas, dan perbaikannya murah.

187.3 Studi kasus penerapan

Sebuah fase posting berjalan tiga jam. Pemeriksaan awal menunjukkan CPU sistem hanya empat puluh persen, sehingga kesimpulan pertama adalah sistem tidak kekurangan daya.

Analisis tunggu menunjukkan komposisi berikut:

Jenis

Waktu

Porsi

CPU aktif

34 menit

19%

Tunggu jurnal

71 menit

39%

Tunggu disk

48 menit

27%

Tunggu kunci record

19 menit

11%

Lainnya

8 menit

4%

Komposisi ini langsung mengarahkan tindakan. Tunggu jurnal yang mendominasi menunjuk ke konfigurasi journal, bukan ke kode program. Setelah journal receiver dipindahkan ke ASP terpisah dengan unit disk sendiri, durasi fase turun menjadi satu jam empat puluh menit.

Yang ingin saya tekankan dari kasus ini: tanpa analisis tunggu, upaya perbaikan hampir pasti diarahkan ke program — meninjau kode, menambah index, atau memecah pemrosesan. Semuanya masuk akal dan semuanya akan memberi perbaikan kecil, sementara penyebab utamanya berada di tempat lain sama sekali.

Bab 188. Job Watcher dan Analisis Mendalam

188.1 Kapan Job Watcher dibutuhkan

Collection Services mengumpulkan data setiap beberapa menit. Untuk masalah yang berlangsung dalam hitungan detik, resolusi itu terlalu kasar.

Job Watcher mengumpulkan data dalam interval sub-detik, termasuk call stack setiap job pada setiap pengambilan. Ini memungkinkan menjawab pertanyaan yang tidak bisa dijawab Collection Services: prosedur apa yang sedang dijalankan job saat ia menunggu.

```
/* Menjalankan Job Watcher untuk periode terbatas */
STRJWMON DFN(QSYS/QJWDFT) +
         COL(JWCOL01) +
         COLLIB(PERFDTA) +
         MAXDTA(500) +
         ITV(1)

/* Menghentikan */
ENDJWMON COL(JWCOL01)
```

Peringatan tentang beban. Job Watcher menghasilkan data dalam jumlah sangat besar dan menambah beban pada sistem. Ia bukan alat yang dijalankan terus-menerus. Pola pemakaian yang tepat: aktifkan selama lima sampai lima belas menit ketika masalah sedang terjadi, lalu hentikan dan analisis.

188.2 Prinsip penafsiran hasil

Prinsip penafsirannya sederhana: prosedur yang paling sering muncul dalam sampel adalah tempat job menghabiskan waktunya. Ini teknik pengambilan sampel statistik, dan ia sangat efektif menemukan titik panas yang tidak terduga.

Yang sering mengejutkan adalah bahwa titik panas jarang berada di tempat yang diduga. Pada program yang dicurigai lambat karena perhitungannya rumit, sampel sering menunjukkan bahwa sebagian besar waktu justru dihabiskan pada operasi baca file yang sederhana tetapi dijalankan jutaan kali.

188.3 Disk Watcher

Untuk masalah yang diduga berasal dari penyimpanan, Disk Watcher memberi resolusi yang tidak tersedia di tempat lain.

```
STRDW DFN(QSYS/QDWDFT) +
      COL(DWCOL01) +
      COLLIB(PERFDTA) +
      ITV(1)

ENDDW COL(DWCOL01)
-- Ketidakseimbangan beban antar unit disk
SELECT UNIT_NUMBER,
       DECIMAL(AVG(ELAPSED_PERCENT_BUSY), 5, 1) AS rata_sibuk,
       MAX(ELAPSED_PERCENT_BUSY) AS puncak_sibuk,
       SUM(ELAPSED_IO_REQUESTS) AS total_io,
       DECIMAL(AVG(PERCENT_USED), 5, 1) AS rata_terpakai
  FROM QSYS2.SYSDISKSTAT
 GROUP BY UNIT_NUMBER
 ORDER BY rata_sibuk DESC;
```

Bila satu atau dua unit jauh lebih sibuk daripada yang lain, penyebabnya biasanya salah satu dari tiga hal: data belum diseimbangkan setelah penambahan disk, objek besar terkonsentrasi di unit tertentu, atau journal receiver berada di unit yang juga menampung data.

⚡ Bab 189. Menyusun Studi Kinerja yang Meyakinkan

189.1 Struktur laporan yang bekerja

Laporan studi kinerja yang hanya menyajikan angka jarang menghasilkan tindakan. Struktur berikut lebih

mungkin diterima.

```
LAPORAN STUDI KINERJA

1. PERNYATAAN MASALAH
   Dinyatakan dengan angka dan dampak bisnis.
   Buruk : "EOD lambat"
   Baik  : "Fase posting naik dari 47 menit menjadi 2 jam 12 menit
            dalam 4 bulan. Pada laju ini, batch window terlampaui
            pada bulan Februari."

2. METODE
   Data apa yang dikumpulkan, dari kapan sampai kapan,
   dengan alat apa.

3. TEMUAN
   Angka, disajikan sebagai perbandingan.

4. HIPOTESIS YANG DIUJI
   Termasuk yang ditolak, beserta alasannya.

5. AKAR MASALAH
   Dengan bukti yang mendukungnya.

6. REKOMENDASI
   Berurutan: usaha, dampak yang diharapkan, risiko.

7. VERIFIKASI YANG DIRENCANAKAN
   Bagaimana kita akan tahu perbaikan berhasil.
```

Bagian keempat adalah yang membedakan laporan yang meyakinkan dari yang tidak. Menyebutkan hipotesis yang ditolak menunjukkan bahwa penyelidikan dilakukan secara menyeluruh, dan mencegah pertanyaan “sudah dicoba ini belum?” pada saat presentasi.

189.2 Aturan satu perubahan

Ketika beberapa perbaikan diterapkan bersamaan dan kinerja membaik, Anda tidak tahu mana yang bekerja. Bila kinerja memburuk, Anda tidak tahu mana yang harus dibatalkan.

```
flowchart TD
    A[Ukur baseline] --> B[Terapkan satu perubahan]
    B --> C[Ukur lagi]
    C --> D{Membaik?}
    D -->|Ya| E[Pertahankan, catat besarnya]
    D -->|Tidak| F[Batalkan]
    E --> G{Cukup?}
    F --> G
    G -->|Belum| B
    G -->|Ya| H[Dokumentasikan]
```

Godaan untuk melanggar aturan ini paling besar ketika tekanan waktu tinggi. Dan justru dalam kondisi itu pelanggarannya paling mahal, karena Anda kehilangan kemampuan membalikkan perubahan yang keliru secara selektif.

189.3 Kesalahan analisis yang sering terjadi

Kesalahan

Mengapa keliru

Menyalahkan job dengan CPU tertinggi

Job itu mungkin bekerja efisien

Memakai rata-rata untuk merencanakan kapasitas

Sistem harus menangani puncak

Mengukur setelah perubahan tanpa baseline

Tidak ada pembanding

Menyimpulkan sebab dari korelasi waktu

Dua hal bisa punya penyebab ketiga

Mengukur pada hari tidak normal

Hasilnya tidak mewakili

Mengabaikan perubahan volume

Sistem lebih lambat karena beban naik

Menganggap CPU rendah berarti tidak ada masalah

Masalah bisa di tunggu

Baris terakhir adalah kesalahan yang paling sering saya temui, dan ia adalah alasan mengapa kerangka wait accounting layak menjadi cara berpikir bawaan.

189.4 Menyampaikan hasil kepada manajemen

Satu halaman yang berisi hal berikut biasanya lebih efektif daripada laporan lengkap:

Satu kalimat masalah, dengan angka

Satu grafik yang menunjukkan tren

Satu kalimat akar masalah

Tiga pilihan dengan biaya dan dampaknya

Rekomendasi yang jelas

Kapan keputusan dibutuhkan

Butir keenam sering dilupakan dan paling menentukan. Sebuah rekomendasi tanpa tenggat keputusan akan mengendap. Sebuah rekomendasi yang menyebutkan bahwa keputusan dibutuhkan sebelum akhir kuartal karena lead time pengadaan enam bulan akan dibahas.

Bagian LV — Pendalaman Work Management

Bab 190. Merancang Subsystem dari Awal

190.1 Mengapa tidak memakai QBATCH saja

Subsystem bawaan cukup untuk sistem kecil. Pada lingkungan core banking, memisahkan beban ke subsystem tersendiri memberi beberapa hal yang tidak bisa dicapai dengan cara lain.

Manfaat

Penjelasan

Pool memori terpisah

Beban batch tidak menggerus pool interaktif

Batas job serentak

Mencegah satu proses membanjiri sistem

Prioritas berbeda

EOD bisa diprioritaskan di atas laporan

Penghentian selektif

Hentikan batch tanpa menghentikan layanan

Pemantauan yang jelas

Status per subsystem langsung bermakna

Manfaat keempat sering baru terasa nilainya saat insiden. Kemampuan menghentikan seluruh pekerjaan batch dengan satu perintah, tanpa menyentuh sesi teller yang sedang melayani nasabah, adalah hal yang sulit dilakukan bila semuanya berjalan di satu subsystem.

190.2 Rancangan yang lazim untuk core banking

Subsystem

Beban

Pool

Batas job

QINTER

Sesi 5250

Interaktif

Sesuai lisensi

QBATCH

Batch umum

Bersama

Sedang

EODSBS

End of Day

Khusus, besar

4 sampai 8

RPTSBS

Pelaporan

Khusus, sedang

2

INTSBS

Interface

Khusus, kecil

4

QSERVER

Server file

Bersama

Bawaan

QUSRWRK

Server pengguna

Bersama

Bawaan

190.3 Membangun subsystem lengkap

```
/* 1. Buat deskripsi subsystem dengan pool khusus */
CRTSBSD SBSD(QGPL/EODSBS) +
        POOLS((1 *BASE) (2 *SHRPOOL3)) +
        TEXT('Subsystem End of Day')

/* 2. Buat class dengan prioritas dan time slice yang sesuai */
CRTCLS CLS(QGPL/EODCLS) +
       RUNPTY(30) +
       TIMESLICE(5000) +
       DFTWAIT(60) +
       TEXT('Class untuk job EOD')

/* 3. Buat job queue */
CRTJOBQ JOBQ(QGPL/EODJOBQ) +
        TEXT('Antrean job EOD')

/* 4. Kaitkan job queue ke subsystem, batasi job serentak */
ADDJOBQE SBSD(QGPL/EODSBS) +
         JOBQ(QGPL/EODJOBQ) +
```

```
         MAXACT(4) +
         SEQNBR(10)

/* 5. Routing entry: arahkan ke class yang benar */
ADDRTGE SBSD(QGPL/EODSBS) +
        SEQNBR(10) +
        CMPVAL('EODJOB') +
        PGM(QSYS/QCMD) +
        CLS(QGPL/EODCLS) +
        POOLID(2)

/* 6. Routing entry cadangan untuk data tidak dikenali */
ADDRTGE SBSD(QGPL/EODSBS) +
        SEQNBR(9999) +
        CMPVAL(*ANY) +
        PGM(QSYS/QCMD) +
        CLS(QGPL/EODCLS) +
        POOLID(2)

/* 7. Job description yang mengarahkan ke routing data */
CRTJOBD JOBD(QGPL/EODJOBD) +
        JOBQ(QGPL/EODJOBQ) +
        RTGDTA('EODJOB') +
        JOBPTY(4) OUTPTY(5) +
        LOG(4 00 *SECLVL) +
        LOGCLPGM(*YES) +
        INLLIBL(APPPGM DTALIB APPWRK) +
        TEXT('Job description untuk EOD')
```

Tentang langkah keenam. Routing entry cadangan dengan CMPVAL(*ANY) mencegah job tertahan tanpa penjelasan ketika routing data-nya tidak cocok dengan entri mana pun. Tanpa itu, job akan berakhir dengan pesan yang membingungkan, dan penyebabnya sulit ditemukan oleh orang yang belum pernah mengalaminya.

Tentang LOGCLPGM(*YES). Ia membuat setiap perintah CL tercatat di job log. Ini menambah ukuran job log, tetapi untuk proses EOD nilainya sangat besar: ketika sesuatu gagal, Anda bisa melihat persis perintah apa yang dijalankan dan dengan parameter apa.

190.4 Menyetel pool memori

```
/* Melihat pool yang ada */
WRKSHRPOOL

/* Menyetel pool khusus EOD
*/
CHGSHRPOOL POOL(*SHRPOOL3) +
           SIZE(8192000) +
           ACTLVL(12) +
           PAGING(*FIXED)
```

Parameter

Arti

Pertimbangan

SIZE

Ukuran pool dalam KB

Cukup agar faulting rendah

ACTLVL

Thread aktif serentak

Terlalu rendah menyebabkan ineligible

PAGING

Metode paging

*CALC untuk penyetelan otomatis

Tentang PAGING(*FIXED) versus *CALC. Nilai *CALC membiarkan sistem menyesuaikan sendiri, dan untuk sebagian besar beban itu pilihan yang baik. Untuk beban EOD yang polanya sangat berbeda dari beban harian, *FIXED dengan nilai yang ditentukan dari pengukuran kadang memberi hasil lebih baik — tetapi hanya bila Anda benar-benar mengukurnya, bukan menebak.

190.5 Memantau kesehatan subsystem

```
-- Status seluruh subsystem beserta beban
SELECT SUBSYSTEM_DESCRIPTION, STATUS,
       CURRENT_ACTIVE_JOBS, MAXIMUM_ACTIVE_JOBS,
       SUBSYSTEM_MONITOR_JOB
  FROM QSYS2.SUBSYSTEM_INFO
 WHERE STATUS = 'ACTIVE'
 ORDER BY CURRENT_ACTIVE_JOBS DESC;

-- Job queue yang menumpuk
SELECT JOB_QUEUE_LIBRARY, JOB_QUEUE_NAME,
       NUMBER_OF_JOBS, JOB_QUEUE_STATUS,
       SUBSYSTEM_NAME, MAXIMUM_ACTIVE_JOBS
  FROM QSYS2.JOB_QUEUE_INFO
 WHERE NUMBER_OF_JOBS > 0
 ORDER BY NUMBER_OF_JOBS DESC;

-- Pool dengan faulting tinggi
SELECT POOL_NAME, SUBSYSTEM_NAME,
       CURRENT_SIZE / 1024 AS mb,
       MAXIMUM_ACTIVE_THREADS,
       DATABASE_FAULTS, NON_DATABASE_FAULTS,
       ACTIVE_TO_INELIGIBLE
  FROM TABLE(QSYS2.MEMORY_POOL())
 WHERE DATABASE_FAULTS + NON_DATABASE_FAULTS > 50
    OR ACTIVE_TO_INELIGIBLE > 0;
```

Kolom ACTIVE_TO_INELIGIBLE yang tidak nol adalah temuan yang layak ditindaklanjuti segera. Ia berarti ada job yang siap bekerja tetapi harus menunggu giliran, dan perbaikannya biasanya hanya menaikkan ACTLVL — perubahan yang murah dan berdampak langsung.

Bab 191. Prestart Job dan Server Job

191.1 Mengapa prestart job penting untuk kanal digital

Setiap koneksi ODBC atau JDBC baru menuntut pembuatan job di IBM i. Pembuatan job adalah operasi yang relatif mahal, dan pada kanal dengan ribuan koneksi per menit biayanya menjadi dominan.

Prestart job menyiapkan job terlebih dahulu, sehingga koneksi baru langsung memakai job yang sudah siap.

```
/* Melihat konfigurasi
prestart job untuk server
database */
DSPSBSD SBSD(QSYS/QUSRWRK)

/* Menyetel prestart job
entry */
CHGPJE SBSD(QSYS/QUSRWRK) +
       PGM(QSYS/QZDASOINIT)
+
       INLJOBS(50) +
       THRESHOLD(20) +
       ADLJOBS(10) +
       MAXJOBS(500) +
       MAXUSE(200) +
       WAIT(*YES) +
       CLS(QGPL/ODBCCLS)
```

Parameter

Arti

Pertimbangan

INLJOBS

Job disiapkan saat subsystem mulai

Sesuai beban dasar

THRESHOLD

Batas bawah sebelum menambah

Sekitar 40% dari INLJOBS

ADLJOBS

Berapa ditambahkan sekaligus

Jangan terlalu kecil

MAXJOBS

Batas atas

Cegah kehabisan sumber daya

MAXUSE

Berapa kali job dipakai ulang

Lihat catatan di bawah

WAIT

Tunggu bila tidak ada yang siap

*YES lebih aman

Tentang MAXUSE. Nilai ini menentukan berapa kali sebuah job dipakai ulang sebelum diakhiri dan diganti. Nilai terlalu tinggi berarti job bertahan lama dan bisa menumpuk sumber daya yang tidak dilepas. Nilai terlalu rendah membuat manfaat prestart hilang karena job terus dibuat ulang.

Nilai antara seratus dan lima ratus wajar untuk sebagian besar beban. Bila Anda melihat job ODBC dengan pemakaian penyimpanan sementara yang terus naik, menurunkan MAXUSE adalah mitigasi yang efektif sambil menunggu perbaikan pada aplikasi.

191.2 Memantau prestart job

```
-- Jumlah dan status job ODBC
SELECT JOB_NAME, AUTHORIZATION_NAME, JOB_STATUS,
       ELAPSED_CPU_PERCENTAGE, TEMPORARY_STORAGE,
       ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         SUBSYSTEM_LIST_FILTER => 'QUSRWRK',
         DETAILED_INFO => 'ALL'))
 WHERE JOB_NAME LIKE '%QZDASOINIT%'
 ORDER BY TEMPORARY_STORAGE DESC;

-- Ringkasan: berapa siap, berapa terpakai
SELECT COUNT(*) AS total,
       SUM(CASE WHEN JOB_STATUS = 'PSRW' THEN 1 ELSE 0 END) AS menunggu,
       SUM(CASE WHEN JOB_STATUS <> 'PSRW' THEN 1 ELSE 0 END) AS terpakai
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(SUBSYSTEM_LIST_FILTER => 'QUSRWRK'))
 WHERE JOB_NAME LIKE '%QZDASOINIT%';
```

Bila jumlah yang menunggu sering mendekati nol, INLJOBS terlalu rendah dan koneksi baru mengalami penundaan. Bila jumlah yang menunggu selalu tinggi, sumber daya terbuang meskipun dampaknya kecil.

191.3 Memisahkan beban ODBC

Secara bawaan seluruh koneksi ODBC berjalan di QUSRWRK dengan konfigurasi yang sama. Untuk lingkungan dengan beban beragam, memisahkannya memberi kendali yang lebih baik.

Pendekatan yang bisa dipakai: buat subsystem tersendiri untuk beban pelaporan, dan arahkan koneksi ke sana lewat exit program yang mengubah atribut job berdasarkan pengguna atau alamat asal.

Dengan begitu, kueri pelaporan berjalan di pool terpisah dengan prioritas lebih rendah, dan tidak bisa mengganggu koneksi transaksional meskipun memakan sumber daya besar.

Bab 192. Penjadwalan Lanjutan

192.1 Batas dari penjadwal bawaan

ADDJOBSCDE cukup untuk pekerjaan yang berjalan pada waktu tetap tanpa ketergantungan. Ia tidak memadai ketika:

Sebuah job harus menunggu job lain selesai

Urutan pelaksanaan bergantung pada hasil

Kegagalan satu langkah harus menghentikan rangkaian

Jadwal berbeda pada hari kerja dan hari libur

Perlu percobaan ulang otomatis dengan batas

Untuk kebutuhan itu, ada dua jalan: produk penjadwal pihak ketiga, atau membangun pengendali sendiri.

192.2 Pola pengendali rangkaian sederhana

```
CREATE TABLE APPWRK.JADWAL_RANTAI (
    KD_RANTAI   VARCHAR(20) NOT NULL,
    URUTAN      SMALLINT    NOT NULL,
    KD_LANGKAH  VARCHAR(20) NOT NULL,
    PROGRAM     VARCHAR(10) NOT NULL,
    LIBRARY     VARCHAR(10) NOT NULL,
    PARAMETER   VARCHAR(200),
    WAJIB       CHAR(1)     NOT NULL DEFAULT 'Y',
    MAKS_COBA   SMALLINT    NOT NULL DEFAULT 1,
    BATAS_MENIT SMALLINT    NOT NULL DEFAULT 60,
    AKTIF       CHAR(1)     NOT NULL DEFAULT 'Y',
    PRIMARY KEY (KD_RANTAI, URUTAN)
);

CREATE TABLE APPWRK.RIWAYAT_RANTAI (
    ID          BIGINT GENERATED ALWAYS AS IDENTITY,
    KD_RANTAI   VARCHAR(20) NOT NULL,
    TGL_JALAN   DATE        NOT NULL,
    URUTAN      SMALLINT    NOT NULL,
    KD_LANGKAH  VARCHAR(20) NOT NULL,
    MULAI       TIMESTAMP,
    SELESAI     TIMESTAMP,
    STATUS      CHAR(1)     NOT NULL,
    JML_COBA    SMALLINT    DEFAULT 0,
    PESAN       VARCHAR(500),
    PRIMARY KEY (ID)
);
```

Pengendalinya membaca tabel jadwal, menjalankan langkah berurutan, mencatat hasil, dan berhenti bila langkah wajib gagal. Kolom MAKS_COBA dan BATAS_MENIT memberi perilaku yang tidak tersedia pada penjadwal bawaan.

Keuntungan pendekatan berbasis tabel. Menambah langkah baru tidak menuntut perubahan kode. Urutan bisa diubah dengan mengubah data. Dan yang paling berharga: riwayat pelaksanaan tersimpan dan bisa dianalisis, sehingga pertanyaan seperti berapa lama langkah ini biasanya berjalan punya jawaban.

192.3 Menangani hari libur

Jadwal yang mengabaikan hari libur akan menjalankan proses pada hari ketika seharusnya tidak, atau melewatkannya pada hari ketika seharusnya berjalan.

```
CREATE TABLE APPPRM.KALENDER (
    TANGGAL     DATE        NOT NULL,
    JENIS_HARI  CHAR(1)     NOT NULL,
    KETERANGAN  VARCHAR(100),
    HARI_KERJA_KE SMALLINT,
    PRIMARY KEY (TANGGAL)
);

-- JENIS_HARI: K kerja, L libur nasional, A akhir pekan, C cuti bersama

-- Hari kerja berikutnya setelah tanggal tertentu
SELECT MIN(TANGGAL) AS hari_kerja_berikutnya
  FROM APPPRM.KALENDER
 WHERE TANGGAL > CURRENT DATE
   AND JENIS_HARI = 'K';

-- Apakah hari ini hari kerja terakhir bulan ini
SELECT CASE WHEN CURRENT DATE = (
         SELECT MAX(TANGGAL) FROM APPPRM.KALENDER
          WHERE JENIS_HARI = 'K'
            AND MONTH(TANGGAL) = MONTH(CURRENT DATE)
            AND YEAR(TANGGAL) = YEAR(CURRENT DATE))
       THEN 'YA' ELSE 'TIDAK' END AS akhir_bulan_kerja
  FROM SYSIBM.SYSDUMMY1;
```

Pelajaran dari Bab 154. Tabel kalender adalah objek yang banyak program bergantung padanya, dan perubahan padanya bisa berakibat jauh. Perlakukan ia sebagai objek yang perubahannya melalui manajemen perubahan, sama seperti kode.

Simpan juga daftar program yang membacanya, sehingga setiap perubahan bisa dinilai dampaknya.

```
-- Program yang membaca tabel kalender
SELECT WHPNAM AS program, WHLNAM AS library, WHUSAG
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'KALENDER'
 ORDER BY program;
```

192.4 Menjaga jadwal tetap terdokumentasi

```
-- Seluruh jadwal aktif dengan keterangannya
SELECT SCHEDULED_JOB_NAME,
       STATUS,
       FREQUENCY,
       SCHEDULE_DATE,
       SCHEDULE_TIME,
       SCHEDULE_DAYS,
       JOB_QUEUE_LIBRARY || '/' || JOB_QUEUE_NAME AS antrean,
       COMMAND_STRING,
       TEXT_DESCRIPTION
  FROM QSYS2.SCHEDULED_JOB_INFO
 WHERE STATUS = '*SCDE'
 ORDER BY SCHEDULE_TIME;
```

Jalankan kueri ini setiap bulan dan simpan hasilnya. Perbandingan antar bulan akan menunjukkan jadwal

yang ditambahkan atau diubah, dan setiap perubahan seharusnya punya tiket yang menjelaskannya.

Kolom TEXT_DESCRIPTION yang kosong adalah temuan tersendiri: sebuah jadwal tanpa keterangan adalah jadwal yang tidak ada yang tahu tujuannya, dan tidak ada yang berani menghapusnya.

Bagian LVI — Pendalaman Ketersediaan Tinggi

🏗️ Bab 193. Arsitektur PowerHA Secara Rinci

193.1 Lapisan yang menyusunnya

PowerHA membangun ketersediaan tinggi di atas beberapa lapisan yang masing-masing punya peran berbeda. Memahami pembagian ini penting karena ketika terjadi masalah, gejalanya muncul di satu lapisan sementara penyebabnya berada di lapisan lain.

Lapisan

Peran

Objek

Cluster

Keanggotaan node dan komunikasi

Cluster, node

Cluster resource group

Kelompok sumber daya yang berpindah bersama

CRG

Recovery domain

Node mana berperan apa

Bagian dari CRG

Perangkat

Penyimpanan yang berpindah

iASP, device CRG

Administrative domain

Objek sistem yang disinkronkan

Cluster admin domain

Aplikasi

Yang dijalankan setelah berpindah

Exit program CRG

193.2 Membangun cluster

```
/* Aktifkan dukungan cluster pada kedua node */
CHGTCPSVR SVRSPCVAL(*INETD) AUTOSTART(*YES)
STRTCPSVR SERVER(*INETD)

/* Buat cluster dengan dua node */
CRTCLU CLUSTER(CLSTR1) +
       NODE((NODEPROD ('10.20.30.50')) +
            (NODEDR   ('10.20.31.50'))) +
       START(*YES) +
       VERSION(*CUR)

/* Periksa status */
DSPCLUINF CLUSTER(CLSTR1)
WRKCLU
-- Status cluster lewat SQL
SELECT NODE_ID, NODE_STATUS, CLUSTER_VERSION,
       DEVICE_DOMAIN, CURRENT_CLUSTER_VERSION
  FROM QSYS2.CLUSTER_INFO;
```

Tentang jalur komunikasi cluster. Node berkomunikasi lewat alamat yang didaftarkan saat pembuatan. Bila hanya ada satu jalur dan jalur itu putus, kedua node akan saling menganggap yang lain mati — kondisi yang berpotensi menjadi split brain.

Praktik yang disarankan: daftarkan dua alamat per node, melalui jalur fisik yang berbeda. Dengan begitu, putusnya satu jalur tidak menghasilkan kesimpulan yang keliru.

193.3 Cluster resource group

```
/* CRG untuk perangkat, yaitu iASP yang berpindah */
CRTCRG CLUSTER(CLSTR1) +
       CRG(CRGDTA) +
       CRGTYPE(*DEV) +
       EXITPGM(APPPGM/CRGEXIT) +
       USRPRF(CRGOWNER) +
       RCYDMN((NODEPROD *PRIMARY) (NODEDR *BACKUP 1)) +
       CFGOBJ((IASP01 *DEVD *ONLINE)) +
       TEXT('CRG untuk iASP data produksi')

/* Mulai CRG */
STRCRG CLUSTER(CLSTR1) CRG(CRGDTA)
```

Parameter EXITPGM menunjuk program yang dipanggil pada setiap peristiwa cluster. Di sinilah logika khusus aplikasi ditempatkan: menghentikan subsystem sebelum berpindah, memulai kembali setelahnya, memberi tahu pihak lain, dan mencatat kronologi.

193.4 Menulis exit program CRG

```
PGM PARM(&INFO)
   DCL VAR(&INFO)     TYPE(*CHAR) LEN(256)
   DCL VAR(&AKSI)     TYPE(*CHAR) LEN(10)
   DCL VAR(&PERAN)    TYPE(*CHAR) LEN(10)
   DCL VAR(&HASIL)    TYPE(*CHAR) LEN(1)

   /* Kode aksi ada di posisi tertentu dalam struktur info */
   CHGVAR VAR(&AKSI) VALUE(%SST(&INFO 17 10))

   SELECT
     /* Sebelum peran berpindah dari node ini */
     WHEN COND(&AKSI *EQ 'SWITCHOVER') THEN(DO)
        CALL PGM(UTILLIB/UTLLOG) PARM('CRGEXIT' 'I' +
                'Switchover dimulai, menghentikan aplikasi')
        ENDSBS SBS(EODSBS) OPTION(*CNTRLD) DELAY(300)
        MONMSG MSGID(CPF0000)
        ENDTCPSVR SERVER(*HTTP)
        MONMSG MSGID(CPF0000)
     ENDDO

     /* Setelah node ini menjadi primer */
     WHEN COND(&AKSI *EQ 'START') THEN(DO)
        CALL PGM(UTILLIB/UTLLOG) PARM('CRGEXIT' 'I' +
                'Node menjadi primer, memulai aplikasi')
        STRSBS SBSD(QGPL/EODSBS)
        MONMSG MSGID(CPF0000)
        STRTCPSVR SERVER(*HTTP) HTTPSVR(WEBPROD)
        MONMSG MSGID(CPF0000)
        CALL PGM(APPPGM/VERIFPOST)
     ENDDO

     WHEN COND(&AKSI *EQ 'FAILOVER') THEN(DO)
        CALL PGM(UTILLIB/UTLLOG) PARM('CRGEXIT' 'W' +
                'FAILOVER terdeteksi')
        CALL PGM(UTILLIB/UTLALERT) PARM('KRITIS' +
                'Failover cluster terjadi')
     ENDDO
   ENDSELECT
ENDPGM
```

Prinsip penting dalam menulis exit program CRG. Ia harus selesai dengan cepat dan tidak boleh menggantung. Cluster menunggu exit program selesai sebelum melanjutkan, dan exit program yang tidak kembali akan menghentikan seluruh proses perpindahan di tengah jalan.

Setiap operasi di dalamnya perlu batas waktu, dan operasi yang berpotensi lama sebaiknya dijalankan sebagai job terpisah yang tidak ditunggu.

193.5 Administrative domain

Objek sistem di luar iASP tidak ikut berpindah. Administrative domain menyinkronkannya antar node.

```
/* Buat administrative domain */
CRTCAD CLUSTER(CLSTR1) +
       ADMDMN(ADMDMN1) +
       DMNNODL(NODEPROD NODEDR) +
       SYNCOPT(*LASTCHG)

STRCAD CLUSTER(CLSTR1)
ADMDMN(ADMDMN1)

/* Tambahkan objek yang perlu
disinkronkan */
ADDCADMRE CLUSTER(CLSTR1)
ADMDMN(ADMDMN1) +
          MRE((*USRPRF BUDI001))

ADDCADMRE CLUSTER(CLSTR1)
ADMDMN(ADMDMN1) +
          MRE((*SYSVAL QSECURITY))

/* Lihat apa yang sudah termonitor
*/
WRKCADMRE CLUSTER(CLSTR1)
ADMDMN(ADMDMN1)
```

Jenis objek yang layak dimasukkan:

Jenis

Alasan

*USRPRF

Profil harus ada di kedua sisi

*SYSVAL

Perilaku sistem harus sama

*JOBD

Job description dipakai aplikasi

*SBSD

Deskripsi subsystem

*CLS

Class untuk prioritas

*AUTL

Authorization list

*ENVVAR

Variabel lingkungan

*NETA

Atribut jaringan

Yang tetap tidak tercakup dan harus diurus terpisah: entri job scheduler, exit program yang terdaftar, sertifikat di DCM, konfigurasi TCP/IP, dan objek di QGPL. Daftar lengkapnya ada di Bab 55.3, dan memeriksanya secara berkala adalah bagian dari pemeliharaan rutin.

```
-- Membandingkan objek antar node, dijalankan di kedua sisi
SELECT OBJNAME, OBJTYPE, OBJOWNER, OBJDEFINER,
       CHANGE_TIMESTAMP
  FROM TABLE(QSYS2.OBJECT_STATISTICS('QGPL', '*ALL'))
 ORDER BY OBJNAME;
```

🗄️ Bab 194. Db2 Mirror

194.1 Perbedaan mendasar dari PowerHA

Db2 Mirror menyediakan replikasi sinkron aktif-aktif pada tingkat basis data, berbeda dari PowerHA yang memindahkan penyimpanan antar node.

Aspek

PowerHA

Db2 Mirror

Mode

Aktif-pasif

Aktif-aktif

Replikasi

Penyimpanan

Basis data, sinkron

RPO

Nol pada switchover terencana

Nol

RTO

Menit

Mendekati nol

Jarak antar node

Fleksibel

Sangat dekat, latensi rendah

Beban baca

Satu node

Bisa dibagi

Kerumitan

Sedang

Tinggi

Biaya

Sedang

Tinggi

Keunggulan terbesarnya adalah kemampuan kedua node menerima transaksi bersamaan, sehingga kegagalan satu node tidak menuntut perpindahan apa pun. Keterbatasan terbesarnya adalah persyaratan latensi yang sangat ketat antar node.

194.2 Kapan Db2 Mirror tepat

Pertimbangan yang menentukan:

Apakah RTO mendekati nol benar-benar dibutuhkan, atau beberapa menit dapat diterima

Apakah kedua lokasi cukup dekat untuk replikasi sinkron

Apakah aplikasi siap berjalan aktif-aktif

Apakah tim punya kapasitas mengelola kerumitan tambahan

Apakah anggarannya tersedia

Butir ketiga sering diremehkan. Aplikasi yang menulis ke tabel yang sama dari dua node bersamaan bisa menghasilkan konflik yang tidak muncul pada konfigurasi aktif-pasif. Sebagian aplikasi core banking perlu penyesuaian sebelum bisa berjalan aktif-aktif dengan aman.

Saran yang realistis. Untuk sebagian besar bank menengah di Indonesia, PowerHA dengan replikasi ke lokasi jauh sudah memenuhi kebutuhan kesinambungan usaha, dan Db2 Mirror menambah kerumitan yang belum sebanding manfaatnya. Ia menjadi tepat ketika ada kebutuhan bisnis spesifik yang tidak bisa dipenuhi dengan RTO dalam hitungan menit.

Bab 195. Menguji Ketersediaan Secara Berkala

195.1 Tingkatan pengujian

Uji DR tahunan yang besar tidak menggantikan pengujian kecil yang sering. Keduanya menjawab pertanyaan berbeda.

Tingkat

Frekuensi

Yang diuji

Gangguan

1

Bulanan

Status cluster, lag replikasi

Tidak ada

2

Triwulanan

Restore satu library

Tidak ada

3

Semesteran

Simulasi lisan skenario

Tidak ada

4

Tahunan

Switchover terencana penuh

Terjadwal

5

Dua tahunan

Failover tanpa pemberitahuan

Nyata

Tingkat lima jarang dilakukan dan paling berharga. Ia menguji apa yang benar-benar terjadi ketika tidak ada persiapan — dan hasilnya hampir selalu berbeda dari uji terencana.

Bila institusi Anda belum siap untuk tingkat lima, langkah antara yang berguna: lakukan uji terencana tetapi tanpa memberi tahu tim tanggal persisnya dalam rentang satu pekan.

195.2 Pemeriksaan bulanan yang murah

```
-- Kesehatan cluster dan CRG
SELECT CLUSTER_NAME, NODE_ID, NODE_STATUS,
       CURRENT_CLUSTER_VERSION
  FROM QSYS2.CLUSTER_INFO;

SELECT CLUSTER_RESOURCE_GROUP, CRG_STATUS,
       PRIMARY_NODE, CRG_TYPE
  FROM QSYS2.CLUSTER_RESOURCE_GROUP_INFO;

-- Objek yang termonitor administrative domain
SELECT MONITORED_RESOURCE, RESOURCE_TYPE,
       GLOBAL_STATUS, LOCAL_STATUS
  FROM QSYS2.CLUSTER_ADMIN_DOMAIN_MRE_INFO
 WHERE GLOBAL_STATUS <> 'CONSISTENT';
```

Kueri ketiga adalah yang paling sering menemukan masalah. Objek yang tidak konsisten antar node berarti sesuatu berubah di satu sisi dan tidak tersinkronkan — dan itu akan menjadi masalah pada saat failover.

195.3 Menyusun laporan kesiapan

Laporan triwulanan yang layak disampaikan ke manajemen risiko memuat:

Butir

Isi

Status cluster

Sehat atau tidak, dengan tanggal pemeriksaan

Lag replikasi

Rata-rata dan puncak sepanjang triwulan

Uji yang dilakukan

Tingkat, tanggal, hasil

RTO aktual terakhir

Dari uji terakhir

Temuan terbuka

Jumlah dan tingkat

Objek di luar cakupan

Yang belum tersinkronkan

Perubahan sejak triwulan lalu

Konfigurasi, prosedur, personel

Baris terakhir sering diabaikan dan penting. Perubahan personel memengaruhi kesiapan DR sama nyatanya dengan perubahan konfigurasi: prosedur yang hanya dipahami oleh orang yang baru pindah adalah prosedur yang belum tentu bisa dijalankan.

195.4 Menjaga dokumen tetap benar

Dokumen DR punya kecenderungan menjadi usang lebih cepat daripada dokumen lain, karena ia jarang dipakai sehingga jarang diperiksa.

Cara menjaganya tetap benar:

Setiap perubahan konfigurasi yang menyentuh DR memicu peninjauan dokumen

Setiap uji DR mencatat ketidakcocokan antara dokumen dan kenyataan

Versi cetak diperbarui setiap November

Salinan disimpan di luar sistem dan di luar lokasi

Dokumen diberi nomor versi dan tanggal, sehingga salinan lama dikenali

Butir kedua adalah mekanisme yang paling efektif, karena uji DR adalah satu-satunya saat dokumen benar-benar dibaca dan dijalankan. Jadikan pencatatan ketidakcocokan sebagai bagian resmi dari prosedur uji, dengan satu orang yang ditugaskan khusus untuk itu.

💻 Bagian LVII — Pendalaman RPG Lanjutan

🏦 Bab 196. Prosedur dan Modularitas

196.1 Dari program monolitik ke prosedur

Program RPG warisan cenderung berbentuk satu berkas panjang dengan ribuan baris, subrutin yang berbagi seluruh variabel, dan alur yang sulit ditelusuri. Pemecahan menjadi prosedur memberi beberapa hal sekaligus.

Manfaat

Penjelasan

Ruang lingkup variabel terbatas

Variabel lokal tidak bisa diubah dari luar

Parameter eksplisit

Ketergantungan terlihat di tanda tangan

Bisa diuji terpisah

Prosedur dipanggil dari program uji

Bisa dipakai ulang

Lewat service program

Lebih mudah dibaca

Setiap prosedur punya satu tujuan

```
**free
ctl-opt nomain option(*srcstmt: *nodebugio);

/copy APPSRC/QRPGLEREF,HITUNGH

dcl-proc hitungBungaHarian export;
  dcl-pi *n packed(17:2);
    pSaldo  packed(17:2) const;
    pRate   packed(9:6)  const;
    pHari   packed(3:0)  const;
  end-pi;

  dcl-s wHasil packed(17:2);

  if pHari = 0;
     kirimEscape('Jumlah hari tidak boleh nol');
  endif;

  if pRate < 0;
     kirimEscape('Rate tidak boleh negatif');
  endif;

  wHasil = %dech(pSaldo * pRate / pHari: 17: 2);
  return wHasil;
end-proc;
```

Tentang ctl-opt nomain. Ia menandakan modul ini tidak punya prosedur utama dan hanya berisi prosedur yang diekspor. Modul semacam ini di-bind ke dalam service program, bukan dijalankan sendiri.

Tentang validasi di dalam prosedur. Memeriksa parameter di awal prosedur, bukan mengandalkan pemanggil, adalah praktik yang jauh lebih aman. Pemanggil bisa banyak dan ditulis orang berbeda; prosedur hanya satu.

196.2 Prototipe dan berkas salinan

Prototipe mendeklarasikan bagaimana sebuah prosedur dipanggil. Ia harus identik antara modul yang mendefinisikan dan yang memanggil, dan cara menjamin itu adalah menyimpannya di satu berkas salinan bersama.

```
**free
// APPSRC/QRPGLEREF,HITUNGH

dcl-pr hitungBungaHarian packed(17:2);
  pSaldo  packed(17:2) const;
  pRate   packed(9:6)  const;
  pHari   packed(3:0)  const;
end-pr;

dcl-pr hitungDenda packed(17:2);
  pPokok     packed(17:2) const;
  pHariTelat packed(5:0)  const;
  pTarif     packed(9:6)  const;
end-pr;

dcl-pr validasiRekening ind;
  pNoRek char(16) const;
  pAlasan varchar(100);
end-pr;
```

Kesalahan yang paling merusak adalah prototipe yang berbeda antara pendefinisi dan pemanggil. Kompilasi berhasil, tetapi saat dijalankan parameter dibaca dari posisi memori yang salah. Gejalanya nilai acak tanpa pesan kesalahan.

Menyimpan prototipe di berkas salinan bersama menghilangkan kelas kesalahan ini sepenuhnya, dan biayanya hanya disiplin untuk tidak menyalin prototipe secara manual.

196.3 Merancang tanda tangan prosedur

Praktik

Alasan

Pakai const untuk parameter masukan

Mencegah perubahan tak sengaja

Kembalikan nilai, bukan ubah parameter

Lebih jelas dibaca

Batasi jumlah parameter

Lebih dari lima sulit dipakai

Pakai data structure untuk parameter banyak

Satu parameter, isi terstruktur

Beri nama yang menyatakan tujuan

hitungBunga bukan proc1

Konsisten dalam urutan

Masukan dulu, keluaran kemudian

```
**free
// Terlalu banyak parameter, sulit dipakai dan mudah tertukar
dcl-pr posting1 ind;
  p1 char(16) const;
  p2 packed(17:2) const;
  p3 char(4) const;
  p4 date const;
  p5 char(40) const;
  p6 char(10) const;
  p7 packed(3:0) const;
end-pr;

// Lebih baik: struktur yang bernama
dcl-ds transaksiT qualified template;
  noRek    char(16);
  nominal  packed(17:2);
  kdTrx    char(4);
  tglBuku  date;
  noRef    char(40);
  petugas  char(10);
  kdCabang packed(3:0);
end-ds;

dcl-pr posting ind;
  pTrx likeds(transaksiT) const;
  pAlasan varchar(100);
end-pr;
```

Bab 197. Service Program dan Pengelolaan Versi

197.1 Signature dan mengapa ia penting

Setiap service program punya signature yang mewakili daftar prosedur yang diekspor beserta urutannya. Program yang di-bind menyimpan signature itu, dan saat dijalankan sistem memeriksa kecocokannya.

Bila signature berubah, seluruh program pemakai harus di-bind ulang. Ini perlindungan yang berguna, tetapi juga sumber gangguan bila tidak dikelola.

```
/* Binder language: APPSRC/QSRVSRC,APPSRV */
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('APPSRV_V3')
  EXPORT SYMBOL('HITUNGBUNGAHARIAN')
  EXPORT SYMBOL('HITUNGDENDA')
  EXPORT SYMBOL('VALIDASIREKENING')
  EXPORT SYMBOL('POSTING')
ENDPGMEXP

STRPGMEXP PGMLVL(*PRV) SIGNATURE('APPSRV_V2')
  EXPORT SYMBOL('HITUNGBUNGAHARIAN')
  EXPORT SYMBOL('HITUNGDENDA')
  EXPORT SYMBOL('VALIDASIREKENING')
ENDPGMEXP

STRPGMEXP PGMLVL(*PRV) SIGNATURE('APPSRV_V1')
  EXPORT SYMBOL('HITUNGBUNGAHARIAN')
  EXPORT SYMBOL('HITUNGDENDA')
ENDPGMEXP
```

Aturan yang harus dipatuhi agar program lama tetap berjalan:

Prosedur baru ditambahkan di akhir daftar, tidak di tengah

Prosedur lama tidak pernah dihapus dari daftar

Urutan prosedur lama tidak pernah diubah

Setiap versi lama dipertahankan sebagai blok *PRV

Melanggar salah satunya membuat seluruh program yang di-bind ke versi lama gagal saat dijalankan, dengan pesan yang menyebut signature tidak cocok.

197.2 Menemukan program yang terpengaruh

```
-- Program yang di-bind ke service program tertentu
SELECT PROGRAM_LIBRARY, PROGRAM_NAME,
       BOUND_SERVICE_PROGRAM_LIBRARY, BOUND_SERVICE_PROGRAM,
       SERVICE_PROGRAM_SIGNATURE
  FROM QSYS2.BOUND_SRVPGM_INFO
 WHERE BOUND_SERVICE_PROGRAM = 'APPSRV'
 ORDER BY PROGRAM_LIBRARY, PROGRAM_NAME;
```

Jalankan kueri ini sebelum mengubah service program. Hasilnya adalah daftar program yang perlu diuji, dan bila perlu di-bind ulang.

197.3 Menambah prosedur tanpa mengganggu

Langkah yang aman:

Tambahkan prosedur baru di akhir blok *CURRENT

Naikkan nama signature

Salin blok lama menjadi *PRV dengan signature lamanya

Kompilasi ulang service program

Verifikasi program lama masih berjalan

Bind ulang program yang perlu memakai prosedur baru

```
CRTSRVPGM SRVPGM(APPPGM/APPSRV) +
          MODULE(APPPGM/HITUNG APPPGM/POSTING APPPGM/VALIDASI) +
          SRCSTMF('/src/binder/appsrv.bnd') +
          EXPORT(*SRCFILE) +
          ACTGRP(*CALLER) +
          REPLACE(*YES)
```

Tentang ACTGRP(*CALLER). Nilai ini membuat service program berjalan di activation group pemanggilnya, sehingga commitment control dan override berlaku dalam cakupan yang sama. Untuk service program yang dipakai dalam rangkaian transaksional, ini hampir selalu nilai yang benar.

197.4 Memantau kesehatan binding

```
-- Program dengan signature tidak cocok, terdeteksi sebelum dijalankan
SELECT p.PROGRAM_LIBRARY, p.PROGRAM_NAME,
       p.BOUND_SERVICE_PROGRAM,
       p.SERVICE_PROGRAM_SIGNATURE AS signature_program,
       s.SIGNATURE AS signature_srvpgm
  FROM QSYS2.BOUND_SRVPGM_INFO p
  JOIN QSYS2.SERVICE_PROGRAM_INFO s
    ON s.PROGRAM_NAME = p.BOUND_SERVICE_PROGRAM
   AND s.PROGRAM_LIBRARY = p.BOUND_SERVICE_PROGRAM_LIBRARY
 WHERE p.SERVICE_PROGRAM_SIGNATURE <> s.SIGNATURE;
```

Kueri ini menemukan masalah sebelum pengguna menemukannya. Jalankan setelah setiap promosi ke produksi.

Bab 198. Penanganan Kesalahan yang Matang

198.1 Tiga lapis penanganan

```
flowchart TD
    A[Operasi gagal] --> B{Ditangani di tingkat operasi?}
    B -->|Ya| C[Indikator e dan status]
    B -->|Tidak| D{Ditangani di tingkat blok?}
    D -->|Ya| E[MONITOR dan ON-ERROR]
    D -->|Tidak| F{Ditangani di tingkat prosedur?}
    F -->|Ya| G[Prosedur penangan]
    F -->|Tidak| H[Diteruskan ke pemanggil]
    H --> I[Job berakhir dengan escape]
```

Prinsip yang saya sarankan: tangani di tingkat serendah mungkin yang masih bermakna. Kesalahan yang bisa dipulihkan ditangani di tempat terjadinya; kesalahan yang tidak bisa dipulihkan diteruskan ke atas agar tidak disembunyikan.

198.2 Pola yang matang

```
**free
dcl-proc prosesRekening;
  dcl-pi *n ind;
    pNoRek  char(16) const;
    pAlasan varchar(200);
  end-pi;

  dcl-s wSqlState char(5);
```

```
  monitor;
     // Operasi yang mungkin gagal
     chain(e) pNoRek REKENING;

     if %error();
        select;
          when %status() = 1218;
            pAlasan = 'Record sedang dipakai job lain';
            return *off;
          when %status() = 1221;
            pAlasan = 'Record belum dibaca sebelum diperbarui';
            kirimEscape(pAlasan);
          other;
            pAlasan = 'Kesalahan baca, status ' + %char(%status());
            kirimEscape(pAlasan);
        endsl;
     endif;

     if not %found(REKENING);
        pAlasan = 'Rekening tidak ditemukan';
        return *off;
     endif;

     exec sql
       UPDATE DTALIB.REKENING
          SET SALDO = SALDO + :wJumlah
        WHERE NO_REK = :pNoRek;

     if sqlcode < 0;
        exec sql GET DIAGNOSTICS CONDITION 1
          :wSqlState = RETURNED_SQLSTATE;
        pAlasan = 'SQL gagal, state ' + wSqlState;
        kirimEscape(pAlasan);
     endif;

     return *on;

  on-error *all;
     unlock(e) REKENING;
     pAlasan = 'Kesalahan tak terduga di prosesRekening';
     kirimEscape(pAlasan);
  endmon;
end-proc;
```

Tiga hal yang membuat pola ini matang.

Pertama, status kesalahan dibedakan. 1218 berarti record terkunci — kondisi sementara yang layak dilaporkan sebagai penolakan. 1221 berarti kesalahan logika program — kondisi yang tidak boleh dilanjutkan.

Kedua, unlock dipanggil di blok on-error, sehingga kunci dilepas pada jalur mana pun.

Ketiga, GET DIAGNOSTICS mengambil informasi kesalahan SQL yang lebih rinci daripada sekadar SQLCODE.

198.3 Pesan yang berguna

Pesan kesalahan yang baik menjawab tiga pertanyaan: apa yang gagal, dalam konteks apa, dan apa yang harus dilakukan.

Buruk

Baik

Error occurred

Gagal memperbarui saldo rekening 1234…5678: record terkunci job 123456/EODUSER/EODPOST

Invalid data

Kode produk DP99 pada rekening 1234…5678 tidak terdaftar di APPPRM.PRODUK

Program failed

HITBUNGA gagal di baris 142: rate untuk produk DP01 bernilai nol pada 2026-09-18

Perbedaannya bukan panjangnya, melainkan bahwa pesan yang baik memuat identitas data yang bermasalah. Pesan tanpa identitas memaksa orang mencari dari awal.

```
**free
dcl-proc kirimEscape;
  dcl-pi *n;
    pPesan varchar(500) const;
  end-pi;

  dcl-pr sndpm extpgm('QMHSNDPM');
    idPesan   char(7)      const;
    berkasPsn char(20)     const;
    dataPesan char(500)    const;
    panjang   int(10)      const;
    jenisPesan char(10)    const;
    antrianPgm char(10)    const;
    counter   int(10)      const;
    kunciPesan char(4);
    kesalahan char(8);
  end-pr;

  dcl-s wKunci char(4);
  dcl-s wErr   char(8);

  sndpm('CPF9898': 'QCPFMSG   *LIBL     ':
        pPesan: %len(pPesan): '*ESCAPE': '*PGMBDY': 1:
        wKunci: wErr);
end-proc;
```

198.4 Mencatat sebelum melempar

Pesan escape akan menghentikan program, dan bila tidak ada yang menangkapnya, isinya hanya tersimpan di job log. Untuk kesalahan yang penting, catat juga ke tabel sebelum melempar.

```
**free
dcl-proc kirimEscapeDicatat;
  dcl-pi *n;
    pProgram varchar(10)  const;
    pKonteks varchar(100) const;
    pPesan   varchar(400) const;
  end-pi;

  exec sql
    INSERT INTO APPWRK.LOG_KESALAHAN
      (WAKTU, PROGRAM, KONTEKS, PESAN, JOB_NAME)
    VALUES (CURRENT TIMESTAMP, :pProgram, :pKonteks, :pPesan,
            (SELECT JOB_NAME FROM TABLE(QSYS2.JOB_INFO(
                    JOB_STATUS_FILTER => '*ACTIVE',
                    JOB_USER_FILTER => '*CURRENT'))
              FETCH FIRST 1 ROW ONLY));

  exec sql COMMIT;

  kirimEscape(pKonteks + ': ' + pPesan);
end-proc;
```

Perhatikan COMMIT sebelum melempar. Tanpa itu, pencatatan akan ikut di-rollback ketika transaksi dibatalkan, dan jejak kesalahan hilang persis pada saat ia paling dibutuhkan.

Ini pengecualian yang disengaja terhadap prinsip bahwa pencatatan mengikuti transaksi. Untuk log kesalahan, bertahan lebih penting daripada konsisten dengan transaksi yang gagal.

🔒 Bagian LVIII — Tata Kelola dan Audit Lanjutan

Bab 199. Membangun Kerangka Kendali

199.1 Dari daftar periksa ke kendali

Banyak institusi memiliki daftar periksa kepatuhan yang panjang tetapi tidak punya kendali yang sesungguhnya. Perbedaannya penting dan sering kabur.

Sebuah daftar periksa menyatakan apa yang seharusnya. Sebuah

	kendali adalah mekanisme yang memastikan hal itu terjadi, dan yang meninggalkan bukti bahwa ia bekerja.
Aspek	
Daftar periksa	
Kendali	
Bentuk	
Pernyataan	
Mekanisme	
Pelaksanaan	
Bergantung orang	
Otomatis atau terjadwal	
Bukti	
Tanda tangan	
Catatan sistem	
Kegagalan	
Tidak terdeteksi	
Menghasilkan peringatan	
Pengujian	
Jarang	
Berkala	

Contoh konkret. Daftar periksa menyatakan bahwa seluruh tabel produksi harus dijurnal. Kendali adalah kueri terjadwal yang mendeteksi tabel tanpa journaling, menghasilkan tiket, dan menyimpan riwayat hasilnya.

Auditor yang berpengalaman akan menanyakan yang kedua. Menunjukkan daftar periksa yang ditandatangani tidak membuktikan apa pun tentang keadaan sistem.

199.2 Merancang kendali yang bisa diuji

Setiap kendali yang baik punya lima unsur.

Unsur

Pertanyaan yang dijawab

Tujuan

Risiko apa yang dikurangi

Mekanisme

Bagaimana ia bekerja

Frekuensi

Seberapa sering dijalankan

Bukti

Apa yang tersimpan sebagai jejak

Kegagalan

Apa yang terjadi bila kendali tidak berjalan

Unsur kelima paling sering hilang. Sebuah kendali yang berhenti berjalan tanpa ada yang menyadarinya memberi rasa aman palsu — persis seperti pemantauan yang mati diam-diam di Bab 113.5.

```
CREATE TABLE AUDITLIB.KENDALI (
    KD_KENDALI   VARCHAR(20)  NOT NULL,
    NAMA         VARCHAR(150) NOT NULL,
    RISIKO       VARCHAR(300) NOT NULL,
    FREKUENSI    VARCHAR(20)  NOT NULL,
    PEMILIK      VARCHAR(50)  NOT NULL,
    OTOMATIS     CHAR(1)      NOT NULL,
    KUERI        VARCHAR(2000),
    AMBANG_GAGAL INTEGER      DEFAULT 0,
    AKTIF        CHAR(1)      NOT NULL DEFAULT 'Y',
    PRIMARY KEY (KD_KENDALI)
);

CREATE TABLE AUDITLIB.HASIL_KENDALI (
```

```sql
    ID           BIGINT GENERATED ALWAYS AS IDENTITY,
    KD_KENDALI   VARCHAR(20) NOT NULL,
    WAKTU        TIMESTAMP   NOT NULL,
    HASIL        CHAR(1)     NOT NULL,
    JML_TEMUAN   INTEGER     NOT NULL DEFAULT 0,
    RINCIAN      VARCHAR(2000),
    DITINJAU_OLEH VARCHAR(50),
    WAKTU_TINJAU TIMESTAMP,
    PRIMARY KEY (ID)
);

CREATE INDEX AUDITLIB.HASIL_IX1
    ON AUDITLIB.HASIL_KENDALI (KD_KENDALI, WAKTU);
```

199.3 Katalog kendali yang layak ada

Kode

Kendali

Frekuensi

K-01

Seluruh tabel produksi dijurnal

Mingguan

K-02

Backup harian lengkap tanpa CPF3730

Harian

K-03

Uji restore berhasil

Triwulanan

K-04

Tidak ada profil *ALLOBJ di luar daftar

Mingguan

K-05

Audit journal aktif dan mencatat

Harian

K-06

Perubahan produksi punya tiket

Mingguan

K-07

Sertifikat tidak kedaluwarsa dalam 60 hari

Mingguan

K-08

Lag replikasi DR dalam batas

Harian

K-09

EOD selesai dan seimbang

Harian

K-10

Tidak ada objek tanpa pemilik yang jelas

Bulanan

K-11

Pemisahan tugas pengembang dan produksi

Bulanan

K-12

Akun layanan dirotasi sesuai jadwal

Bulanan

K-13

Tinjauan akses dilakukan

Triwulanan

K-14

Kapasitas dalam ambang aman

Mingguan

K-15

Interface berjalan sesuai jadwal

Harian

199.4 Menjalankan dan mencatat

```
-- Menjalankan kendali K-01 dan mencatat hasilnya
INSERT INTO AUDITLIB.HASIL_KENDALI
  (KD_KENDALI, WAKTU, HASIL, JML_TEMUAN, RINCIAN)
WITH temuan AS (
  SELECT o.OBJNAME
    FROM TABLE(QSYS2.OBJECT_STATISTICS('DTALIB','*FILE')) o
   WHERE o.OBJATTRIBUTE IN ('PF','PF38')
     AND NOT EXISTS (SELECT 1 FROM QSYS2.JOURNALED_OBJECTS j
                     WHERE j.JOURNALED_OBJECT = o.OBJNAME
                       AND j.JOURNALED_OBJECT_LIBRARY = 'DTALIB')
)
SELECT 'K-01', CURRENT TIMESTAMP,
       CASE WHEN COUNT(*) = 0 THEN 'L' ELSE 'G' END,
       COUNT(*),
       COALESCE(LISTAGG(OBJNAME, ', '), 'Tidak ada temuan')
  FROM temuan;
-- Kendali yang seharusnya berjalan tetapi tidak
SELECT k.KD_KENDALI, k.NAMA, k.FREKUENSI, k.PEMILIK,
       MAX(h.WAKTU) AS terakhir_dijalankan,
       DAYS(CURRENT DATE) - DAYS(DATE(MAX(h.WAKTU))) AS umur_hari
  FROM AUDITLIB.KENDALI k
  LEFT JOIN AUDITLIB.HASIL_KENDALI h ON h.KD_KENDALI = k.KD_KENDALI
 WHERE k.AKTIF = 'Y'
 GROUP BY k.KD_KENDALI, k.NAMA, k.FREKUENSI, k.PEMILIK
HAVING MAX(h.WAKTU) IS NULL
    OR (k.FREKUENSI = 'HARIAN'
        AND MAX(h.WAKTU) < CURRENT TIMESTAMP - 2 DAYS)
    OR (k.FREKUENSI = 'MINGGUAN'
        AND MAX(h.WAKTU) < CURRENT TIMESTAMP - 10 DAYS)
    OR (k.FREKUENSI = 'BULANAN'
        AND MAX(h.WAKTU) < CURRENT TIMESTAMP - 40 DAYS)
 ORDER BY umur_hari DESC;
```

Kueri kedua adalah kendali atas kendali, dan ia yang membedakan kerangka yang hidup dari kerangka yang hanya ada di atas kertas.

🔒 Bab 200. Menghadapi Audit

200.1 Persiapan yang mengubah pengalaman

Audit yang dipersiapkan dengan baik berlangsung berbeda dari audit yang dihadapi mendadak, dan

perbedaannya bukan pada hasil akhirnya saja tetapi pada beban kerjanya.

Persiapan yang efektif dilakukan pada tiga tingkat waktu:

Waktu

Kegiatan

Sepanjang tahun

Menjalankan kendali dan menyimpan buktinya

Tiga bulan sebelum

Menjalankan audit mandiri dan memperbaiki temuan

Dua pekan sebelum

Menyiapkan berkas bukti dan personel

Audit mandiri tiga bulan sebelumnya adalah langkah yang paling berdampak. Temuan yang Anda temukan sendiri bisa diperbaiki; temuan yang ditemukan auditor menjadi catatan resmi dengan seluruh beban administratifnya.

200.2 Menjawab pertanyaan dengan baik

Praktik

Alasan

Jawab yang ditanya, tidak lebih

Informasi tambahan menimbulkan pertanyaan baru

Tunjukkan bukti, bukan penjelasan

Bukti tidak bisa diperdebatkan

Akui bila tidak tahu

Menebak yang keliru merusak kredibilitas

Catat pertanyaan yang sulit

Untuk perbaikan tahun depan

Jangan berargumen tentang temuan yang valid

Energi lebih baik untuk perbaikan

Butir pertama sering dilanggar karena niat baik. Seorang engineer yang bersemangat menjelaskan seluruh arsitektur ketika ditanya satu hal spesifik membuka jalur pertanyaan yang tidak perlu.

Catatan tentang butir ketiga. Mengatakan tidak tahu, lalu menyediakan jawaban dalam waktu yang dijanjikan, jauh lebih baik daripada menjawab spekulatif. Auditor mencatat jawaban Anda, dan jawaban yang kemudian terbukti keliru memunculkan pertanyaan tentang keandalan seluruh keterangan Anda.

200.3 Bukti yang kuat

Tidak semua bukti sama kuatnya di mata auditor.

Kekuatan

Bentuk bukti

Terkuat

Catatan sistem yang tidak bisa diubah, misalnya audit journal

Kuat

Keluaran kueri dengan stempel waktu, tersimpan berkala

Sedang

Laporan yang dihasilkan sistem

Lemah

Dokumen yang disusun manual

Terlemah

Pernyataan lisan

Konsekuensi praktisnya: bangun kendali yang meninggalkan jejak pada tingkat yang setinggi mungkin. Sebuah

kendali yang hasilnya tersimpan otomatis di tabel dengan stempel waktu jauh lebih mudah dipertanggungjawabkan daripada kendali yang buktinya berupa lembar tanda tangan.

200.4 Menindaklanjuti temuan

```
CREATE TABLE AUDITLIB.TEMUAN (
    NO_TEMUAN    VARCHAR(20)  NOT NULL,
    SUMBER       VARCHAR(50)  NOT NULL,
    TANGGAL      DATE         NOT NULL,
    TINGKAT      CHAR(1)      NOT NULL,
    URAIAN       VARCHAR(1000) NOT NULL,
    AKAR_MASALAH VARCHAR(1000),
    RENCANA      VARCHAR(1000),
    PENANGGUNG   VARCHAR(50)  NOT NULL,
    TENGGAT      DATE         NOT NULL,
    STATUS       CHAR(1)      NOT NULL DEFAULT 'B',
    TGL_SELESAI  DATE,
    BUKTI        VARCHAR(500),
    PRIMARY KEY (NO_TEMUAN)
);

-- Temuan yang melewati tenggat
SELECT NO_TEMUAN, TINGKAT, URAIAN, PENANGGUNG, TENGGAT,
       DAYS(CURRENT DATE) - DAYS(TENGGAT) AS hari_terlambat
  FROM AUDITLIB.TEMUAN
 WHERE STATUS <> 'S'
   AND TENGGAT < CURRENT DATE
 ORDER BY TINGKAT, hari_terlambat DESC;

-- Temuan berulang dari tahun ke tahun
SELECT URAIAN, COUNT(*) AS jml_muncul,
       MIN(TANGGAL) AS pertama, MAX(TANGGAL) AS terakhir
  FROM AUDITLIB.TEMUAN
 GROUP BY URAIAN
HAVING COUNT(*) > 1
 ORDER BY jml_muncul DESC;
```

Kueri kedua menghasilkan daftar yang layak dibawa ke manajemen. Temuan yang muncul kembali setelah dinyatakan selesai menunjukkan bahwa perbaikannya tidak bertahan, dan itu masalah yang berbeda dari temuan aslinya.

200.5 Membangun hubungan dengan auditor

Ini bagian yang jarang dibahas dan cukup menentukan.

Auditor internal bukan lawan. Mereka menjalankan fungsi yang sama-sama bertujuan menjaga institusi, dan sebagian besar temuan mereka valid. Hubungan yang baik menghasilkan beberapa hal yang bermanfaat:

Anda mengetahui area fokus audit lebih awal

Temuan didiskusikan sebelum menjadi resmi

Konteks teknis dipahami, sehingga temuan lebih tepat

Rekomendasi Anda bisa didukung oleh temuan mereka

Butir keempat layak disadari. Sebuah permintaan anggaran yang didukung temuan audit punya bobot politis yang tidak dimiliki permintaan biasa. Ini bukan manipulasi; ini pemakaian mekanisme yang memang ada

untuk tujuan yang sama.

Bab 201. Manajemen Risiko Teknologi

201.1 Menyusun daftar risiko yang berguna

Daftar risiko yang panjang dan generik tidak berguna. Daftar yang berguna spesifik, terukur, dan punya pemilik.

```
CREATE TABLE AUDITLIB.RISIKO (
    KD_RISIKO    VARCHAR(20)  NOT NULL,
    URAIAN       VARCHAR(500) NOT NULL,
    KATEGORI     VARCHAR(30)  NOT NULL,
    KEMUNGKINAN  SMALLINT     NOT NULL,
    DAMPAK       SMALLINT     NOT NULL,
    SKOR         SMALLINT     GENERATED ALWAYS AS (KEMUNGKINAN * DAMPAK),
    MITIGASI     VARCHAR(1000),
    KENDALI      VARCHAR(200),
    PEMILIK      VARCHAR(50)  NOT NULL,
    TINJAU_BERIKUT DATE       NOT NULL,
    PRIMARY KEY (KD_RISIKO)
);
```

Kolom KENDALI yang menunjuk ke tabel kendali di Bab 199 adalah yang membuat daftar ini hidup. Sebuah risiko tanpa kendali yang memitigasinya hanyalah kekhawatiran yang tertulis.

201.2 Risiko khas lingkungan IBM i

Risiko

Kemungkinan

Dampak

Mitigasi

Kehilangan ahli kunci

Tinggi

Tinggi

Dokumentasi, pasangan, rotasi

Vendor aplikasi berhenti mendukung

Sedang

Sangat tinggi

Escrow, kemampuan internal

Rilis OS tidak lagi didukung

Pasti

Tinggi

Rencana upgrade berkala

Pemulihan gagal saat dibutuhkan

Sedang

Sangat tinggi

Uji restore berkala

EOD gagal melewati batch window

Sedang

Tinggi

Pemantauan tren, perbaikan

Kapasitas habis mendadak

Rendah

Tinggi

Pemantauan dan proyeksi

Akses tidak sah dari dalam

Rendah

Sangat tinggi

RCAC, audit, exit program

Perubahan tanpa kendali

Sedang

Tinggi

Baseline dan pembanding

Kode tanpa sumber

Pasti di sebagian sistem

Tinggi

Inventarisasi dan penulisan ulang

Baris pertama layak mendapat perhatian khusus di konteks Indonesia. Jumlah praktisi IBM i yang berpengalaman terbatas, dan permintaan cukup tinggi, sehingga kemungkinan kehilangan orang kunci lebih tinggi daripada di platform lain.

201.3 Menyampaikan risiko kepada manajemen

Bahasa yang bekerja bukan bahasa teknis, melainkan bahasa akibat.

Bahasa teknis

Bahasa akibat

Tabel X tidak dijurnal

Bila terjadi kerusakan data pada modul ini, kita tidak bisa memulihkan ke titik waktu; pemulihan terakhir dari backup semalam

Tidak ada uji restore

Kita tidak tahu apakah bisa pulih dari bencana; RTO yang tertulis belum pernah dibuktikan

Satu orang memahami EOD

Bila orang ini tidak tersedia, penanganan kegagalan EOD akan memakan waktu berjam-jam lebih lama

Kode tanpa sumber

Program ini tidak bisa diperbaiki bila rusak, dan akan berhenti berfungsi pada upgrade rilis berikutnya

Penerjemahan ini bukan penyederhanaan yang mengurangi ketepatan. Ia justru lebih tepat, karena menyatakan risiko dalam kerangka yang bisa dinilai dan diputuskan oleh orang yang punya wewenang memutuskan.

201.4 Meninjau daftar risiko

Daftar risiko yang tidak pernah berubah adalah daftar yang tidak dipakai. Peninjauan yang baik mengubah

sesuatu setiap kali.

Pertanyaan yang dipakai saat meninjau:

Apakah risiko ini masih relevan

Apakah kemungkinannya berubah sejak terakhir ditinjau

Apakah mitigasinya berjalan

Apakah ada risiko baru yang muncul dari perubahan sistem

Apakah ada risiko yang bisa ditutup

Butir kelima jarang dijawab dan penting. Risiko yang sudah dimitigasi sepenuhnya layak ditutup, dan menutupnya memberi ruang perhatian untuk yang lain. Daftar risiko yang hanya bertambah tanpa pernah berkurang akan menjadi terlalu panjang untuk dibaca.

📕 Bagian LIX — Katalog Runbook Tambahan

Lima belas runbook berikut melengkapi tiga puluh yang ada di Bagian IX dan XXXIX, menjadikan katalognya empat puluh lima. Disusun dengan struktur yang sama agar bisa dijalankan tanpa membaca bab lain.

⚡ Bab 202. Runbook Kinerja

RB-031: Fase EOD melewati baseline dua kali lipat

Pemicu. Tabel kontrol menunjukkan durasi sebuah fase lebih dari dua kali rata-rata tiga puluh hari terakhir, sementara fase itu masih berjalan.

Dampak. Batch window berisiko terlampaui.

Diagnosis cepat, lima menit pertama

```
-- 1. Fase apa, sudah berjalan berapa lama
SELECT NM_FASE, JAM_MULAI,
       TIMESTAMPDIFF(4, CHAR(CURRENT
TIMESTAMP - JAM_MULAI)) AS menit_berjalan
  FROM APPWRK.EOD_KONTROL
 WHERE TGL_BUKU = CURRENT DATE AND STATUS =
'R';

-- 2. Status job dan ke mana waktunya pergi
SELECT JOB_NAME, JOB_STATUS, FUNCTION,
```

```
       ELAPSED_CPU_PERCENTAGE,
ELAPSED_TOTAL_DISK_IO_COUNT,
       TEMPORARY_STORAGE
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(
         SUBSYSTEM_LIST_FILTER => 'EODSBS',
         DETAILED_INFO => 'ALL'));

-- 3. Ada yang menahan?
SELECT JOB_NAME, LOCK_STATE, TABLE_SCHEMA,
TABLE_NAME
  FROM QSYS2.RECORD_LOCK_INFO;
```

Pohon keputusan

Temuan	Tindakan
Status LCKW	Cari penahan, lepaskan bila sesi tertinggal
Status MSGW	Baca dan jawab pesan
CPU tinggi, I/O rendah	Kemungkinan perulangan; siapkan opsi hentikan
I/O tinggi, CPU rendah	Kemungkinan index hilang; periksa jalur akses
Penyimpanan sementara naik cepat	Kueri boros; siapkan opsi hentikan
Semua normal, hanya volume besar	Tunggu, hitung ulang perkiraan selesai

Keputusan pada batas waktu. Bila perkiraan selesai melewati jam buka layanan, eskalasi ke manajer dengan tiga angka: berapa persen selesai, perkiraan selesai, dan pilihan yang tersedia. Jangan menunggu sampai batas tercapai untuk mengeskalasi.

RB-032: Sistem melambat menyeluruh pada jam layanan

Pemicu. Keluhan dari banyak cabang sekaligus, bukan satu.

Diagnosis berurutan

```
-- Beban sistem
SELECT AVERAGE_CPU_UTILIZATION, SYSTEM_ASP_USED,
       CURRENT_TEMPORARY_STORAGE, ACTIVE_JOBS_IN_SYSTEM
  FROM TABLE(QSYS2.SYSTEM_STATUS());

-- Faulting pool
SELECT POOL_NAME, CURRENT_SIZE/1024 AS mb,
       DATABASE_FAULTS, NON_DATABASE_FAULTS, ACTIVE_TO_INELIGIBLE
  FROM TABLE(QSYS2.MEMORY_POOL());

-- Job paling boros
SELECT JOB_NAME, AUTHORIZATION_NAME, ELAPSED_CPU_PERCENTAGE,
       TEMPORARY_STORAGE, FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY ELAPSED_CPU_PERCENTAGE DESC
 FETCH FIRST 10 ROWS ONLY;
```

Penyebab yang paling sering, berurutan menurut kelaziman

Satu kueri ad hoc yang boros — turunkan prioritasnya

Job batch berjalan di jam layanan — tunda ke malam

Faulting tinggi karena pool kurang — pindahkan memori

Disk mendekati penuh sehingga alokasi melambat

Reorganisasi atau pembangunan index berjalan tak terjadwal

Tindakan sementara yang aman

```
/* Turunkan prioritas job yang mengganggu */
CHGJOB JOB(nomor/user/nama) RUNPTY(60)

/* Tahan job queue batch agar tidak ada job baru masuk */
HLDJOBQ JOBQ(QGPL/RPTJOBQ)
```

RB-033: Pemakaian penyimpanan sementara melonjak

Pemicu. ASP naik cepat tanpa penambahan data bisnis.

Diagnosis

```
SELECT JOB_NAME, AUTHORIZATION_NAME,
       TEMPORARY_STORAGE / 1024 AS mb,
       FUNCTION, ELAPSED_TIME
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 ORDER BY TEMPORARY_STORAGE DESC
 FETCH FIRST 15 ROWS ONLY;
```

Perbaikan. Satu job yang menahan ratusan GB hampir selalu kueri dengan join yang keliru. Akhiri job itu, penyimpanan kembali seketika.

PERINGATAN. Verifikasi job itu bukan bagian dari EOD sebelum mengakhirinya. Mengakhiri fase posting di tengah jauh lebih mahal daripada menunggu ASP naik beberapa persen.

Pencegahan. QUERY_TIME_LIMIT dan STORAGE_LIMIT di QAQQINI untuk lingkungan pelaporan.

📕 Bab 203. Runbook Integrasi

RB-034: Berkas interface masuk tetapi tidak diproses

Pemicu. Mitra melaporkan berkas sudah dikirim, tetapi data tidak muncul.

Diagnosis

```sql
-- Berkas yang ada di direktori masuk
SELECT PATH_NAME, DATA_SIZE, CREATE_TIMESTAMP,
       ACCESS_TIMESTAMP
  FROM TABLE(QSYS2.IFS_OBJECT_STATISTICS(
         START_PATH_NAME => '/interface/in',
         SUBTREE_DIRECTORIES => 'NO'))
 ORDER BY CREATE_TIMESTAMP DESC;

-- Riwayat pemrosesan
SELECT NAMA_BERKAS, WAKTU_TERIMA, WAKTU_PROSES, STATUS, PESAN
  FROM APPWRK.INTERFACE_LOG
 WHERE TGL = CURRENT DATE
 ORDER BY WAKTU_TERIMA DESC;
```

Kemungkinan penyebab

Penyebab

Tanda

Berkas penanda selesai tidak ada

Berkas data ada, .done tidak

Nama berkas tidak sesuai pola

Ada di direktori tapi diabaikan

Job pembaca tidak berjalan

Tidak ada job di subsystem interface

Wewenang berkas tidak cukup

Job tidak bisa membaca

Berkas masih ditulis mitra

Ukuran masih bertambah

Penyebab pertama paling sering. Pola penanda selesai ada untuk mencegah pembacaan berkas yang belum lengkap, dan mitra yang lupa mengirimkannya menghasilkan gejala persis ini.

RB-035: Interface mengirim data kosong

Pemicu. Berkas terbentuk dan terproses, tetapi jumlah barisnya nol atau jauh di bawah biasanya.

Ini lebih berbahaya daripada kegagalan, karena seluruh proses melaporkan sukses.

Diagnosis

```
-- Bandingkan volume dengan hari-hari sebelumnya
SELECT TGL, NAMA_BERKAS, JML_BARIS,
       AVG(JML_BARIS) OVER (PARTITION BY NAMA_BERKAS
                            ORDER BY TGL
                            ROWS BETWEEN 30 PRECEDING AND 1 PRECEDING)
         AS rata_30hari
  FROM APPWRK.INTERFACE_LOG
 WHERE NAMA_BERKAS = 'MUTASI'
 ORDER BY TGL DESC
 FETCH FIRST 10 ROWS ONLY;
```

Tindakan. Jangan proses lebih lanjut. Hubungi mitra untuk konfirmasi apakah volume rendah itu wajar. Data kosong yang diproses sebagai data sah akan menghasilkan laporan yang salah tanpa jejak kesalahan.

Pencegahan. Tambahkan validasi volume minimum ke proses pemuatan: berkas dengan jumlah baris di bawah persentase tertentu dari rata-rata ditolak dan memicu peringatan.

RB-036: API mitra tidak merespons

Pemicu. Transaksi menggantung, atau kode kesalahan timeout meningkat.

Diagnosis

```
-- Kesehatan per mitra satu jam terakhir
SELECT KD_MITRA,
       COUNT(*) AS total,
       SUM(CASE WHEN STATUS = 'S' THEN 1 ELSE 0 END) AS berhasil,
       SUM(CASE WHEN STATUS = 'K' THEN 1 ELSE 0 END) AS menggantung,
       DECIMAL(AVG(DURASI_MS), 9, 1) AS rata_ms,
       MAX(DURASI_MS) AS terlama
  FROM DTALIB.PESAN_KORELASI
 WHERE WAKTU_KIRIM > CURRENT TIMESTAMP - 1 HOUR
 GROUP BY KD_MITRA;
# Uji dari sisi sistem
ping mitra.example.co.id
openssl s_client -connect mitra.example.co.id:443 </dev/null 2>&1 | head -20
```

Tindakan berurutan

Verifikasi masalah di sisi mitra, bukan jaringan kita

Aktifkan circuit breaker agar transaksi baru ditolak cepat, bukan menggantung

Hubungi mitra lewat jalur eskalasi yang disepakati

Catat transaksi yang menggantung untuk rekonsiliasi

Setelah pulih, proses ulang dari antrean dengan idempotency key

Yang paling penting adalah langkah kedua. Transaksi yang menggantung menunggu timeout menahan sumber daya dan memperburuk keadaan. Menolak cepat lebih baik daripada menunggu lama.

📕 Bab 204. Runbook Data

RB-037: Selisih ditemukan pada rekonsiliasi harian

Pemicu. Pemeriksaan pagi menemukan total buku besar tidak sama dengan total subsidiary.

Diagnosis berlapis

```
-- 1. Besar selisih
SELECT (SELECT SUM(SALDO) FROM DTALIB.REKENING WHERE STATUS='A') AS subsidiary,
       (SELECT SUM(SALDO_AKHIR) FROM DTALIB.GLSALDO
         WHERE KD_GL IN (SELECT KD_GL FROM APPPRM.PETA_GL
                         WHERE JENIS='DPK')) AS buku_besar
  FROM SYSIBM.SYSDUMMY1;

-- 2. Akun mana
SELECT p.KD_GL, SUM(r.SALDO) AS subsidiary, g.SALDO_AKHIR AS gl,
       SUM(r.SALDO) - g.SALDO_AKHIR AS selisih
  FROM DTALIB.REKENING r
  JOIN APPPRM.PETA_GL p ON p.KD_PRODUK = r.KD_PRODUK
  JOIN DTALIB.GLSALDO g ON g.KD_GL = p.KD_GL
 WHERE r.STATUS = 'A'
 GROUP BY p.KD_GL, g.SALDO_AKHIR
HAVING SUM(r.SALDO) <> g.SALDO_AKHIR;

-- 3. Transaksi yang tidak berpasangan
SELECT NO_REF, SUM(DEBET) AS d, SUM(KREDIT) AS k,
       SUM(DEBET) - SUM(KREDIT) AS selisih, MIN(PGM_PENULIS) AS program
  FROM DTALIB.GLTRX
 WHERE TGL_BUKU = CURRENT DATE - 1 DAY
 GROUP BY NO_REF
HAVING SUM(DEBET) <> SUM(KREDIT);
```

JANGAN LAKUKAN sebelum penyebab diketahui: penyesuaian manual, date roll berikutnya, atau pembukaan layanan dengan angka yang belum cocok.

Eskalasi. Selisih buku besar melibatkan akuntansi dan operasi, bukan hanya TI. Libatkan mereka sejak awal.

RB-038: Data nasabah tampak rusak setelah integrasi baru

Pemicu. Keluhan nama atau alamat tampil dengan karakter aneh, biasanya dari kanal baru.

Diagnosis

```
-- Apakah data di tabel benar
SELECT ID_NASABAH, NAMA, HEX(NAMA) AS heksa
  FROM DTALIB.CIF_UTAMA WHERE ID_NASABAH = :p;

-- CCSID kolom
SELECT COLUMN_NAME, CCSID, DATA_TYPE
  FROM QSYS2.SYSCOLUMNS
 WHERE TABLE_SCHEMA='DTALIB' AND TABLE_NAME='CIF_UTAMA'
   AND DATA_TYPE IN ('CHAR','VARCHAR');
```

Bila heksa benar tetapi tampilan salah, masalahnya di jalur keluar, bukan di data. Periksa CCSID kolom lebih

dulu — nilai 65535 adalah tersangka utama.

Perbaikan. Mengubah CCSID kolom menuntut pengujian seluruh program pemakainya. Lakukan di UAT lebih dulu, tidak langsung di produksi.

RB-039: Tabel tumbuh jauh di atas proyeksi

Pemicu. Pemeriksaan kapasitas menemukan satu tabel tumbuh jauh lebih cepat dari perkiraan.

Diagnosis

```
SELECT TABLE_NAME,
NUMBER_ROWS,
NUMBER_DELETED_ROWS,
       DATA_SIZE / 1048576
AS mb,

DECIMAL(NUMBER_DELETED_ROWS
* 100.0 /

NULLIF(NUMBER_ROWS +
NUMBER_DELETED_ROWS, 0), 5,
1) AS persen_hapus
  FROM QSYS2.SYSTABLESTAT
 WHERE TABLE_SCHEMA =
'DTALIB'
 ORDER BY DATA_SIZE DESC
 FETCH FIRST 15 ROWS ONLY;
```

Tiga kemungkinan

Kemungkinan

Tanda

Tindakan

Record terhapus menumpuk

persen_hapus tinggi

RGZPFM

Volume bisnis memang naik

Sejalan dengan transaksi

Perbarui proyeksi

Proses menulis duplikat

Jumlah baris tidak wajar

Selidiki program penulis

Kemungkinan ketiga paling berbahaya dan paling jarang diperiksa. Sebuah proses yang menulis record

berulang karena penanda idempotensi hilang akan menumbuhkan tabel dengan cepat, dan gejalanya terlihat sebagai masalah kapasitas.

📕 Bab 205. Runbook Operasional

RB-040: Job scheduler tidak menjalankan jadwal

Pemicu. Pekerjaan terjadwal tidak berjalan tanpa pesan kesalahan.

Diagnosis

```
SELECT SCHEDULED_JOB_NAME, STATUS, FREQUENCY,
       SCHEDULE_TIME, SCHEDULE_DAYS,
       NEXT_SUBMISSION_DATE, LAST_SUBMISSION_DATE,
       LAST_SUBMISSION_STATUS
  FROM QSYS2.SCHEDULED_JOB_INFO
 WHERE SCHEDULED_JOB_NAME = 'EODSTART';
```

Temuan

Penyebab

STATUS bukan *SCDE

Jadwal ditahan

NEXT_SUBMISSION_DATE lampau

Jadwal tidak maju, periksa frekuensi

Job queue ditahan

Job dikirim tapi tidak diambil

Subsystem tidak aktif

Tidak ada yang mengambil job

Profil pengirim dinonaktifkan

Pengiriman gagal senyap

Baris terakhir adalah penyebab yang paling membingungkan. Jadwal tetap terdaftar dan tampak normal, tetapi pengirimannya gagal karena profil yang dipakai sudah tidak aktif.

RB-041: Output queue penuh dan cetak berhenti

Pemicu. Laporan tidak tercetak, spooled file menumpuk.

Diagnosis

```
SELECT OUTPUT_QUEUE_LIBRARY, OUTPUT_QUEUE_NAME,
       COUNT(*) AS jml, SUM(SIZE)/1048576 AS mb,
       MIN(CREATE_TIMESTAMP) AS tertua
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
 GROUP BY OUTPUT_QUEUE_LIBRARY, OUTPUT_QUEUE_NAME
HAVING COUNT(*) > 5000
 ORDER BY jml DESC;
```

Perbaikan

```
/* Periksa writer */
WRKWTR WTR(*ALL)
STRPRTWTR DEV(PRT01) OUTQ(APPPGM/CETAK)

/* Bersihkan yang melewati retensi, setelah verifikasi */
DLTSPLF FILE(*SELECT) SELECT(*ALL *ALL *ALL *ALL)
```

PERINGATAN. Verifikasi kebijakan retensi sebelum menghapus. Laporan regulator yang terhapus tidak bisa dipulihkan, dan itu temuan audit serius.

RB-042: Sesi 5250 tidak bisa dibuat

Pemicu. Pengguna tidak bisa masuk, pesan menyebut batas terlampaui.

Diagnosis

```
-- Jumlah sesi interaktif aktif
SELECT COUNT(*) AS sesi_aktif
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_TYPE_FILTER => '*INTERACT'));

-- Sesi menganggur lama
SELECT JOB_NAME, AUTHORIZATION_NAME, JOB_STATUS,
       ELAPSED_TIME, FUNCTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_TYPE_FILTER => '*INTERACT'))
 WHERE JOB_STATUS = 'DSPW'
 ORDER BY ELAPSED_TIME DESC;
```

Perbaikan. Akhiri sesi yang menganggur sangat lama, setelah memberi peringatan. Jangka panjang: setel QINACTITV agar sesi menganggur diakhiri otomatis.

RB-043: Perubahan diterapkan tanpa tiket

Pemicu. Pembanding baseline menemukan perubahan yang tidak punya tiket.

Diagnosis

```sql
-- Objek yang berubah dalam 7 hari terakhir
SELECT OBJNAME, OBJTYPE, OBJOWNER, CHANGE_TIMESTAMP, OBJTEXT
  FROM TABLE(QSYS2.OBJECT_STATISTICS('APPPGM','*ALL'))
 WHERE CHANGE_TIMESTAMP > CURRENT TIMESTAMP - 7 DAYS
 ORDER BY CHANGE_TIMESTAMP DESC;

-- Siapa yang mengubah, dari audit journal
SELECT ENTRY_TIMESTAMP, USER_NAME, JOB_NAME,
       OBJECT_NAME, OBJECT_TYPE, ENTRY_TYPE
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'CO,DO,OM,ZC',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 7 DAYS))
 WHERE OBJECT_LIBRARY = 'APPPGM'
 ORDER BY ENTRY_TIMESTAMP;
```

Tindakan. Ini bukan sekadar temuan teknis. Perubahan produksi tanpa tiket adalah pelanggaran tata kelola yang perlu ditindaklanjuti sesuai prosedur institusi, bukan diselesaikan diam-diam antar teknisi.

Pencegahan. Baseline bulanan dengan pembanding otomatis, dan pembatasan wewenang ubah objek di library produksi.

RB-044: Profil dengan wewenang khusus muncul tiba-tiba

Pemicu. Pemeriksaan mingguan menemukan profil *ALLOBJ yang tidak ada di daftar yang disetujui.

Diagnosis

```sql
-- Kapan dan oleh siapa wewenang diberikan
SELECT ENTRY_TIMESTAMP, USER_NAME AS pemberi,
       OBJECT_NAME AS profil, ENTRY_TYPE, ENTRY_DATA
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('QSYS','QAUDJRN',
         JOURNAL_ENTRY_TYPES => 'CP',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 30 DAYS))
 WHERE OBJECT_NAME = 'PROFIL_BARU'
 ORDER BY ENTRY_TIMESTAMP;
```

Tindakan. Perlakukan sebagai potensi insiden keamanan sampai terbukti sebaliknya. Ikuti Bab 99: jangan mengubah apa pun sebelum eskalasi, kecuali ada indikasi penyalahgunaan aktif.

RB-045: Sistem tidak merespons sama sekali

Pemicu. Tidak bisa masuk lewat jalur mana pun, sesi yang ada tidak merespons.

Ini kondisi paling jarang dan paling menekan.

Urutan tindakan

Verifikasi bukan masalah jaringan: coba dari lokasi berbeda

Akses konsol lewat HMC atau konsol fisik

Periksa SRC di panel atau HMC

Bila sistem masih hidup tetapi tidak merespons, periksa apakah restricted state

Bila ada job yang menghabiskan seluruh sumber daya, akhiri dari konsol

Bila tidak ada jalan, IPL paksa adalah pilihan terakhir

PERINGATAN KRITIS. Sebelum IPL paksa, catat SRC dan pastikan ia tidak sedang bergerak. IPL paksa saat sistem sedang melakukan pemulihan penyimpanan akan memulai proses dari awal dan memperpanjang pemadaman berjam-jam.

Setelah pulih. Kumpulkan seluruh bukti sebelum apa pun dibersihkan: history log, SRC, job log terakhir, dan waktu persis setiap peristiwa. Analisis akar masalah untuk insiden jenis ini hampir selalu menuntut keterlibatan IBM, dan mereka akan meminta data itu.

📕 Bab 206. Indeks Runbook Lengkap

Kode

Judul

Kategori

RB-001

Disk ASP melewati ambang kritis

Kapasitas

RB-002

Job MSGW di tengah EOD

EOD

RB-003

Record lock menahan job EOD

EOD

RB-004

EOD gagal di fase akrual

EOD

RB-005

GL tidak seimbang

Data

RB-006

Replikasi DR tertinggal

HA/DR

RB-007

Kinerja menurun mendadak

Kinerja

RB-008

Backup semalam gagal

Backup

RB-009

Level check massal

Aplikasi

RB-010

Profil terkunci massal

Keamanan

RB-011

IPL tidak selesai

Sistem

RB-012

Job tabel penuh

Sistem

RB-013

Akses ODBC mencurigakan

Keamanan

RB-014

Tabel riwayat tumbuh tak terkendali

Data

RB-015

Kehilangan partisi produksi

HA/DR

RB-016

Tabel produksi tidak dijurnal

Data

RB-017

Constraint check pending

Data

RB-018

Index rusak

Data

RB-019

Kueri ad hoc melumpuhkan sistem

Kinerja

RB-020

Program berbeda dari sumbernya

Aplikasi

RB-021

Program hilang dari produksi

Aplikasi

RB-022

Laporan regulator tidak terbentuk

Kepatuhan

RB-023

Transaksi ganda

Data

RB-024

Sertifikat kedaluwarsa

Keamanan

RB-025

Waktu sistem tidak sinkron

Sistem

RB-026

Perangkat tape tidak terbaca

Backup

RB-027

Partisi kehabisan memori

Kinerja

RB-028

Multipath hilang satu sisi

Infrastruktur

RB-029

Profil wewenang berlebih

Keamanan

RB-030

Audit journal berhenti mencatat

Keamanan

RB-031

Fase EOD melewati baseline

EOD

RB-032

Sistem melambat menyeluruh

Kinerja

RB-033

Penyimpanan sementara melonjak

Kapasitas

RB-034

Berkas interface tidak diproses

Integrasi

RB-035

Interface mengirim data kosong

Integrasi

RB-036

API mitra tidak merespons

Integrasi

RB-037

Selisih pada rekonsiliasi harian

Data

RB-038

Data nasabah tampak rusak

Data

RB-039

Tabel tumbuh di atas proyeksi

Kapasitas

RB-040

Job scheduler tidak berjalan

Operasional

RB-041

Output queue penuh

Operasional

RB-042

Sesi 5250 tidak bisa dibuat

Operasional

RB-043

Perubahan tanpa tiket

Tata kelola

RB-044

Wewenang khusus muncul tiba-tiba

Keamanan

RB-045

Sistem tidak merespons

Sistem

Cetak halaman ini dan tempel di ruang operasi. Pada pukul tiga pagi, mengetahui bahwa runbook-nya ada dan nomornya berapa sudah menghemat separuh waktu penanganan.

💻 Bagian LX — Pendalaman DDS dan Layar 5250

Bagian XXVI memperkenalkan DDS pada tingkat yang cukup untuk membaca sumber yang sudah ada. Bagian ini melanjutkannya ke tingkat yang dibutuhkan ketika Anda harus mengubah layar produksi, mendiagnosis perilaku yang aneh, atau memelihara aplikasi yang tidak punya dokumentasi.

Ada alasan mengapa ini masih relevan. Sebagian besar transaksi teller di bank yang memakai IBM i tetap berjalan lewat layar 5250, dan akan tetap begitu untuk waktu yang lama. Layar itu cepat, tidak menuntut pelatihan ulang, dan sudah teruji puluhan tahun. Seorang System Engineer yang tidak bisa membaca DDS akan tersandung setiap kali masalah menyentuh lapisan tampilan.

Bab 207. Anatomi Display File

207.1 Tingkatan dalam sebuah display file

Sebuah display file disusun dalam tiga tingkat, dan memahami tingkat mana yang sedang Anda lihat menentukan apakah perubahan Anda berlaku untuk satu field atau seluruh layar.

Tingkat

Ditandai

Cakupan

File

Kolom 17 kosong, di awal sumber

Seluruh record format

Record

R di kolom 17

Satu layar atau satu bagian layar

Field

Nama di kolom 19 sampai 28

Satu field

```
     A*%%TS  SD  20260918  103015  ROMI        REL-V7R6M0
     A                                       DSPSIZ(24 80 *DS3)
     A                                       PRINT
     A                                       CA03(03 'Keluar')
     A                                       CA12(12 'Batal')
     A          R HEADER
     A                                  1  2'CIF001'
     A                                  1 28'PEMELIHARAAN DATA NASABAH'
     A                                      DSPATR(HI)
     A                                  1 70DATE
     A                                      EDTCDE(Y)
     A          R DETAIL
     A                                  5  2'Nomor CIF . . . :'
     A            IDCIF         10A  O  5 22
     A                                  7  2'Nama lengkap  . :'
     A            NAMA          60A  B  7 22
     A                                      CHECK(LC)
     A  50                                  DSPATR(RI PC)
     A                                  9  2'Tanggal lahir . :'
     A            TGLLHR          L  B  9 22
     A                                      DATFMT(*ISO)
     A  51                                  DSPATR(RI PC)
     A                                 11  2'Kode cabang . . :'
     A            KDCAB          3S 0B 11 22
     A                                      VALUES(001 002 003 004)
     A  52                                  DSPATR(RI PC)
     A          R FOOTER
     A                                 23  2'F3=Keluar  F12=Batal'
     A                                      COLOR(BLU)
     A            PESAN         78A  O  24  2
     A                                      DSPATR(HI)
```

207.2 Membaca kolom-kolom yang menentukan

Format DDS berbasis posisi kolom, dan itu sumber kebingungan terbesar bagi orang yang baru membacanya.

Kolom

Isi

Contoh

6

Tipe sumber, selalu A

A

7 sampai 16

Indikator kondisi

A 50

17

R untuk record format

R

19 sampai 28

Nama field atau record

NAMA

30 sampai 34

Panjang

60

35

Tipe data

A, S, P, L, T

36

Jumlah desimal

0, 2

38

Penggunaan

I, O, B, H

39 sampai 41

Baris di layar

7

42 sampai 44

Kolom di layar

22

45 ke kanan

Kata kunci

DSPATR(HI)

Kolom 38 adalah yang paling sering ditanyakan. Nilai I berarti input saja, O keluaran saja, B keduanya, dan H tersembunyi. Field H tidak muncul di layar tetapi nilainya dikirim bolak-balik antara program dan workstation — yang berarti ia tidak boleh dipakai untuk menyimpan data rahasia, karena isinya ada di aliran data terminal.

207.3 Indikator dan mengapa ia membingungkan

Indikator adalah mekanisme utama komunikasi antara program dan layar, dan ia menjadi sumber kesalahan yang khas karena satu nomor bisa dipakai untuk banyak hal di tempat berbeda.

Pemakaian

Arah

Contoh

Kondisi tampilan

Program ke layar

A 50 di depan DSPATR(RI)

Tombol fungsi

Layar ke program

CA03(03)

Respons

Layar ke program

CF04(04)

Kendali subfile

Dua arah

SFLDSP(51)

Praktik yang mengurangi kebingungan secara drastis: beri nama indikator lewat INDARA dan struktur data indikator di RPG, bukan memakai nomor telanjang.

```
**free
dcl-f LAYARCIF workstn indds(indikator);

dcl-ds indikator qualified;
  keluar          ind pos(3);
  batal           ind pos(12);
  namaSalah       ind pos(50);
  tanggalSalah    ind pos(51);
  cabangSalah     ind pos(52);
  tampilSubfile   ind pos(60);
  akhirSubfile    ind pos(61);
end-ds;
```

Dengan itu, kode menjadi terbaca:

```
**free
if indikator.namaSalah;
   pesan = 'Nama tidak boleh kosong';
endif;
```

alih-alih:

```
**free
if *in50;
   pesan = 'Nama tidak boleh kosong';
endif;
```

Perbedaannya besar ketika orang lain membaca kode Anda enam bulan kemudian.

207.4 Kata kunci yang paling sering dipakai

Kata kunci

Fungsi

DSPATR(HI)

Tampilkan lebih terang

DSPATR(RI)

Balik warna, menandai kesalahan

DSPATR(PC)

Tempatkan kursor di field ini

DSPATR(UL)

Garis bawah

DSPATR(ND)

Tidak ditampilkan, untuk kata sandi

COLOR(RED)

Warna teks

CHECK(LC)

Izinkan huruf kecil

CHECK(ME)

Field wajib diisi

CHECK(MF)

Field harus diisi penuh

VALUES(...)

Batasi nilai yang diterima

RANGE(...)

Batasi rentang

COMP(...)

Bandingkan dengan nilai

EDTCDE(...)

Format tampilan angka

EDTWRD(...)

Format kustom

ERRMSG(...)

Pesan kesalahan di baris bawah

OVERLAY

Jangan hapus layar sebelumnya

PUTOVR

Tulis ulang field tertentu saja

CLRL(*NO)

Jangan bersihkan baris

Tentang DSPATR(ND). Ia menyembunyikan tampilan, bukan datanya. Kata sandi yang diketik di field ND tetap terkirim dalam aliran data dan bisa terbaca oleh alat yang menangkap lalu lintas terminal. Bila Anda punya layar login 5250 yang memakai ND di atas koneksi Telnet tanpa TLS, itu temuan keamanan yang nyata, bukan teoretis.

Bab 208. Subfile Secara Mendalam

208.1 Dua record format yang selalu berpasangan

Subfile adalah cara DDS menampilkan daftar. Ia selalu terdiri dari dua record format: satu untuk barisnya, satu untuk kendalinya.

```
     A          R SFLDTL                     SFL
     A            SELEKSI        1A  B  8  3
     A            SNOREK        16A  O  8  6
     A            SNAMA         30A  O  8 25
     A            SSALDO        15Y 20  8 58
     A                                     EDTCDE(J)
     A          R SFLCTL                   SFLCTL(SFLDTL)
     A                                     SFLSIZ(0015)
     A                                     SFLPAG(0014)
     A  60                                 SFLDSP
     A  61                                 SFLDSPCTL
     A  62                                 SFLCLR
     A  63                                 SFLEND(*MORE)
     A                                     OVERLAY
     A                                 6  3'Pil'
     A                                 6  8'Nomor rekening'
     A                                 6 27'Nama'
     A                                 6 60'Saldo'
     A                                     DSPATR(HI UL)
```

208.2 Empat indikator kendali yang harus dipahami

Indikator

Kata kunci

Arti bila aktif

60

SFLDSP

Tampilkan isi subfile

61

SFLDSPCTL

Tampilkan record kendali, yaitu judul kolom

62

SFLCLR

Kosongkan subfile

63

SFLEND

Tampilkan penanda akhir daftar

Urutan pemakaiannya menentukan perilaku yang benar:

```
**free
// 1. Kosongkan subfile lebih dulu
indikator.tampilSubfile = *off;
indikator.tampilKendali = *off;
indikator.kosongkan     = *on;
write SFLCTL;
indikator.kosongkan     = *off;

// 2. Isi subfile
nomorRelatif = 0;
exec sql DECLARE c1 CURSOR FOR
  SELECT NO_REK, NAMA, SALDO FROM DTALIB.REKENING
   WHERE ID_NASABAH = :wIdNasabah ORDER BY NO_REK;
exec sql OPEN c1;

dow nomorRelatif < 500;
   exec sql FETCH c1 INTO :sNoRek, :sNama, :sSaldo;
   if sqlcode <> 0;
      leave;
   endif;
   seleksi = ' ';
   nomorRelatif += 1;
   write SFLDTL;
enddo;
exec sql CLOSE c1;

// 3. Tentukan indikator tampilan
indikator.tampilKendali = *on;
indikator.tampilSubfile = (nomorRelatif > 0);
indikator.akhirSubfile  = *on;

// 4. Tampilkan
write FOOTER;
exfmt SFLCTL;
```

Kesalahan yang paling sering pada langkah ini: lupa mematikan SFLCLR setelah menulis record kendali.

Akibatnya subfile dikosongkan lagi setiap kali ditulis, dan layar tampil kosong meskipun data ada. Gejala ini membingungkan karena program tidak melaporkan kesalahan apa pun.

208.3 Subfile besar dan pemuatan bertahap

Ada tiga pola pemuatan subfile, dan pemilihannya berdampak nyata pada kinerja.

Pola

Cara kerja

Cocok untuk

Muat semua

Isi seluruh record sekaligus

Daftar kecil, di bawah 500 baris

Muat satu halaman

Isi hanya yang tampil, muat lagi saat scroll

Daftar besar

Muat sampai batas

Isi sampai jumlah tertentu, beri peringatan

Kompromi yang praktis

Untuk daftar mutasi rekening yang bisa berisi puluhan ribu baris, pola pertama akan membuat pengguna menunggu lama sebelum layar muncul, dan memakan penyimpanan sementara yang besar.

Pola yang saya sarankan untuk lingkungan perbankan: muat sampai batas, dengan batas yang wajar dan pesan yang jujur.

```
**free
if nomorRelatif >= MAKS_BARIS;
   pesan = 'Ditampilkan ' + %char(MAKS_BARIS) +
           ' baris pertama. Persempit rentang tanggal.';
endif;
```

Ini lebih baik daripada memuat semuanya, dan lebih baik daripada memotong diam-diam. Pengguna yang melihat pesan itu tahu bahwa ada data lain dan tahu apa yang harus dilakukan.

208.4 Membaca pilihan dari subfile

Setelah pengguna menandai baris, program membaca kembali record yang berubah.

```
**free
// READC hanya mengembalikan record yang diubah pengguna
readc SFLDTL;
dow not %eof(LAYARCIF);
   select;
     when seleksi = '1';
       prosesLihat(sNoRek);
     when seleksi = '2';
       prosesUbah(sNoRek);
     when seleksi = '4';
       prosesHapus(sNoRek);
     other;
       pesan = 'Pilihan ' + seleksi + ' tidak dikenali';
   endsl;

   // Kembalikan penanda agar tidak terbaca lagi
   seleksi = ' ';
   update SFLDTL;

   readc SFLDTL;
enddo;
```

Tentang update SFLDTL setelah memproses. Tanpa itu, penanda pilihan tetap ada di subfile, dan pada penekanan Enter berikutnya baris yang sama akan diproses lagi. Pada layar hapus rekening, itu bukan gangguan kecil.

208.5 Diagnosis masalah subfile yang lazim

Gejala

Penyebab paling sering

Layar kosong padahal data ada

SFLDSP tidak aktif, atau SFLCLR masih aktif

Judul kolom hilang

SFLDSPCTL tidak aktif

Baris lama muncul lagi

Subfile tidak dikosongkan sebelum diisi

Penanda akhir tidak muncul

SFLEND tidak aktif

Baris terproses dua kali

Penanda pilihan tidak dikembalikan

Kesalahan saat menulis baris

SFLSIZ terlampaui tanpa SFLSIZ lebih besar dari SFLPAG

Kursor melompat

SFLRCDNBR tidak disetel

Baris terakhir dalam tabel menjelaskan perilaku yang sering dikeluhkan pengguna: setelah menekan Enter, kursor kembali ke baris pertama alih-alih tetap di tempat. Solusinya adalah field SFLRCDNBR yang menyimpan nomor relatif baris yang ingin ditampilkan.

Bab 209. Mengubah Layar Produksi dengan Aman

209.1 Mengapa perubahan DDS lebih berisiko daripada tampaknya

Mengubah display file terlihat seperti perubahan kosmetik, tetapi ia menyentuh hal yang tidak selalu terlihat.

Perubahan

Risiko tersembunyi

Menambah field

Level check pada seluruh program pemakai

Mengubah panjang field

Level check, dan data terpotong

Mengubah tipe field

Level check, konversi gagal

Mengubah nama field

Program tidak lagi menemukannya

Memindahkan posisi

Aman, tetapi pengguna kehilangan kebiasaan

Mengubah kata kunci tampilan

Umumnya aman

Menambah record format

Aman bila program lama tidak memakainya

Empat baris pertama memicu level check, yaitu kondisi ketika program yang dikompilasi dengan definisi lama dijalankan terhadap file dengan definisi baru. Sistem menolak, dan job berakhir dengan CPF4131.

209.2 Menemukan seluruh program pemakai

```
/* Kumpulkan referensi program di library aplikasi */
DSPPGMREF PGM(APPPGM/*ALL) OBJTYPE(*ALL) +
          OUTPUT(*OUTFILE) OUTFILE(QTEMP/PGMREF)
-- Program yang memakai display file tertentu
SELECT WHPNAM AS program, WHLNAM AS library,
       WHSNAM AS berkas, WHUSAG AS penggunaan
  FROM QTEMP.PGMREF
 WHERE WHFNAM = 'LAYARCIF'
 ORDER BY program;
```

Jalankan ini sebelum menyentuh sumber DDS. Hasilnya adalah daftar program yang harus dikompilasi ulang dan diuji.

209.3 Prosedur perubahan yang aman

```
flowchart TD
    A[Kebutuhan perubahan layar] --> B[Inventarisasi program pemakai]
    B --> C{Perubahan memicu level check?}
    C -->|Tidak| D[Ubah DDS, kompilasi, uji]
    C -->|Ya| E[Rencanakan kompilasi ulang serentak]
    E --> F[Uji seluruh program di UAT]
    F --> G{Semua lulus?}
    G -->|Tidak| H[Perbaiki, ulangi]
    H --> F
    G -->|Ya| I[Jadwalkan penerapan di luar jam layanan]
    I --> J[Terapkan display file dan program bersamaan]
    J --> K[Verifikasi satu transaksi tiap kanal]
    D --> K
```

Butir yang paling menentukan adalah kotak berlabel terapkan bersamaan. Display file baru dan program yang dikompilasi ulang harus masuk produksi dalam satu operasi. Bila display file masuk lebih dulu, seluruh program lama langsung gagal dengan level check.

209.4 Menambah field tanpa memicu level check

Ada teknik yang memungkinkan penambahan field tanpa memaksa kompilasi ulang seluruh program: memakai LVLCHK(*NO) pada file, atau menambahkan field ke record format yang tidak dipakai program lama.

```
CHGDSPF FILE(APPPGM/LAYARCIF) LVLCHK(*NO)
```

Saya tidak menyarankan LVLCHK(*NO) untuk lingkungan produksi perbankan. Level check adalah perlindungan yang menangkap ketidaksesuaian sebelum data rusak. Mematikannya berarti program lama akan membaca field dari posisi yang salah tanpa peringatan apa pun, dan gejalanya berupa data yang tampak acak — jenis masalah yang paling mahal untuk didiagnosis.

Pendekatan yang lebih aman bila kompilasi ulang serentak tidak memungkinkan: buat record format baru dengan nama berbeda, biarkan program lama memakai format lama, dan pindahkan program satu per satu.

209.5 Menguji layar tanpa mengganggu pengguna

```
/* Jalankan program dengan display file versi uji */
OVRDSPF FILE(LAYARCIF) TOFILE(UJILIB/LAYARCIF) OVRSCOPE(*JOB)
CALL PGM(APPPGM/CIF001)
DLTOVR FILE(LAYARCIF)
```

Teknik override ini memungkinkan menguji layar baru dengan program produksi, tanpa mengubah apa pun di library produksi. Ia sangat berguna untuk memverifikasi tata letak sebelum penerapan.

Bab 210. Printer File dan Laporan

210.1 Struktur printer file

Printer file memakai DDS dengan kata kunci yang berbeda dari display file, tetapi struktur kolomnya sama.

```
     A                                     REF(DTALIB/REKENING)
     A                                     CHRSIZ(1 1)
     A          R JUDUL                    SKIPB(1)
     A                                1  2'BANK CONTOH'
     A                                1 45'Halaman:'
     A            HALAMAN        4S 0      1 55
     A                                     EDTCDE(Z)
     A                                2  2'LAPORAN SALDO REKENING'
     A                                2 45'Tanggal:'
     A            TGLCETAK        L        2 55
     A                                     DATFMT(*DMY) DATSEP('/')
     A                                4  2'No. Rekening'
     A                                4 22'Nama'
     A                                4 55'Saldo'
     A                                5  2'-------------------------------'
     A          R BARIS                    SPACEA(1)
     A            PNOREK    R              6  2REFFLD(NO_REK)
     A            PNAMA     R              6 22REFFLD(NAMA)
     A            PSALDO    R              6 50REFFLD(SALDO)
     A                                     EDTCDE(J)
     A          R TOTAL                    SPACEB(2)
     A                                1  2'TOTAL'
     A            PTOTAL        17S 2      1 48
     A                                     EDTCDE(J)
```

210.2 Kata kunci pengendali posisi

Kata kunci

Fungsi

SKIPB(n)

Lompat ke baris n sebelum mencetak

SKIPA(n)

Lompat ke baris n setelah mencetak

SPACEB(n)

Beri n baris kosong sebelum

SPACEA(n)

Beri n baris kosong sesudah

OVERLAY

Cetak di halaman yang sama

INDTXT

Dokumentasi indikator

Tentang REF dan REFFLD. Kata kunci ini mengambil definisi field dari tabel basis data, sehingga panjang dan tipe selalu cocok. Bila kolom NAMA di tabel diperpanjang, printer file cukup dikompilasi ulang tanpa mengubah sumbernya.

Ini praktik yang layak diterapkan pada seluruh printer file dan display file baru. Ia menghilangkan satu kelas kesalahan sepenuhnya: ketidakcocokan antara definisi tabel dan definisi tampilan.

210.3 Mengendalikan keluaran

```
/* Arahkan keluaran dan setel atributnya */
OVRPRTF FILE(LAPSALDO) +
        OUTQ(APPPGM/CETAK) +
        HOLD(*YES) +
        SAVE(*YES) +
        USRDTA('SALDO-HARIAN') +
        FORMTYPE('A4') +
        COPIES(2) +
        OVRSCOPE(*JOB)

CALL PGM(APPPGM/LAPSALDO)
DLTOVR FILE(LAPSALDO)
```

Parameter USRDTA layak dipakai secara konsisten. Ia mengisi field yang bisa dicari, sehingga menemukan laporan tertentu di antara ribuan spooled file menjadi satu kueri.

```
SELECT SPOOLED_FILE_NAME, JOB_NAME, USER_DATA,
       CREATE_TIMESTAMP, TOTAL_PAGES, STATUS
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
 WHERE OUTPUT_QUEUE_NAME = 'CETAK'
   AND USER_DATA LIKE 'SALDO%'
 ORDER BY CREATE_TIMESTAMP DESC;
```

210.4 Dari spooled file ke PDF

Banyak institusi masih mencetak laporan yang sebenarnya hanya diarsipkan. Mengubahnya menjadi PDF menghemat kertas dan membuat pencarian jauh lebih mudah.

```
/* Hasilkan PDF langsung dari printer file */
OVRPRTF FILE(LAPSALDO) +
        DEVTYPE(*AFPDS) +
        TOSTMF('/arsip/laporan/saldo-20260918.pdf') +
        WSCST(*PDF) +
        OVRSCOPE(*JOB)

CALL PGM(APPPGM/LAPSALDO)
DLTOVR FILE(LAPSALDO)
```

Pertimbangan retensi. Bila laporan yang sebelumnya dicetak kini disimpan sebagai PDF di IFS, kewajiban retensinya berpindah ke IFS. Pastikan direktori itu ikut dalam cakupan backup, dan pastikan wewenangnya dibatasi seperti laporan cetak dibatasi. Ini dua hal yang sering terlewat saat memindahkan laporan ke bentuk digital.

🚀 Bab 211. Modernisasi Tampilan Tanpa Menulis Ulang

211.1 Tiga tingkat modernisasi tampilan

Tingkat

Yang berubah

Usaha

Risiko

Refacing

Tampilan saja, lewat perkakas

Rendah

Rendah

Pembungkusan

Logika diekspos sebagai layanan

Sedang

Sedang

Penulisan ulang

Antarmuka dan logika baru

Tinggi

Tinggi

Tingkat pertama mengubah tampilan 5250 menjadi halaman web tanpa menyentuh program sama sekali. Perkakas membaca aliran data 5250 dan menerjemahkannya menjadi HTML.

Kelebihannya: cepat, murah, tidak berisiko terhadap logika bisnis.

Keterbatasannya: hasilnya tetap mengikuti alur layar hijau. Sebuah proses yang menuntut tujuh layar berurutan akan tetap menuntut tujuh halaman berurutan. Pengalaman penggunanya lebih rapi, tetapi tidak lebih efisien.

211.2 Kapan refacing cukup, kapan tidak

Refacing cukup ketika masalahnya adalah tampilan: pengguna baru kesulitan dengan layar hijau, atau tampilan itu menghambat citra institusi.

Refacing tidak cukup ketika masalahnya adalah alur: proses yang menuntut terlalu banyak langkah, atau data yang harus diketik ulang di beberapa layar.

Pertanyaan yang membedakan: bila layar ini tampil sebagai halaman web yang cantik, apakah pekerjaannya menjadi lebih cepat? Bila jawabannya tidak, masalahnya bukan di tampilan.

211.3 Pola pembungkusan yang praktis

Untuk proses yang perlu dipanggil dari kanal digital, pola yang bekerja adalah memisahkan logika dari layar.

```
**free
// SEBELUM: logika tercampur dengan layar
dow not indikator.keluar;
   exfmt DETAIL;
   if indikator.keluar;
      leave;
   endif;
   // validasi
   if nama = *blanks;
      pesan = 'Nama wajib diisi';
      indikator.namaSalah = *on;
      iter;
   endif;
   // proses
   exec sql UPDATE DTALIB.CIF_UTAMA
               SET NAMA = :nama WHERE ID_NASABAH = :idcif;
enddo;
**free
// SESUDAH: logika di prosedur, layar hanya memanggil
dcl-proc ubahNamaNasabah export;
  dcl-pi *n ind;
    pIdCif  char(10)     const;
    pNama   varchar(60)  const;
    pAlasan varchar(200);
  end-pi;

  if %trim(pNama) = '';
     pAlasan = 'Nama wajib diisi';
     return *off;
  endif;

  exec sql UPDATE DTALIB.CIF_UTAMA
             SET NAMA = :pNama WHERE ID_NASABAH = :pIdCif;

  if sqlcode <> 0;
     pAlasan = 'Gagal memperbarui, SQLCODE ' + %char(sqlcode);
     return *off;
  endif;

  return *on;
end-proc;
```

Setelah dipisahkan, prosedur itu bisa dipanggil dari tiga tempat: program layar 5250 yang lama, layanan REST untuk kanal digital, dan program batch untuk pemuatan massal. Validasinya sama di ketiganya, dan itulah

nilai utamanya.

Ini pekerjaan yang bisa dilakukan bertahap. Tidak perlu menulis ulang seluruh aplikasi. Setiap kali sebuah fungsi dibutuhkan oleh kanal baru, pisahkan fungsi itu saja. Setelah beberapa tahun, bagian yang paling sering dipakai sudah terpisah, dan sisanya bisa dibiarkan.

211.4 Menghitung nilai sebelum memulai

Sebelum mengusulkan proyek modernisasi tampilan, angka berikut membuat usulannya bisa dinilai.

Yang diukur

Cara mengukur

Berapa layar yang benar-benar dipakai

Audit journal atau pencatatan pemakaian

Berapa lama pelatihan pengguna baru

Tanya bagian SDM

Berapa transaksi per layar per hari

Tabel transaksi

Berapa kesalahan input per hari

Log kesalahan aplikasi

Berapa layar yang tidak pernah dipakai

Selisih dari inventaris

Baris terakhir sering mengejutkan. Pada aplikasi yang sudah berjalan dua puluh tahun, tidak jarang sepertiga layar tidak pernah dibuka dalam setahun terakhir. Layar-layar itu tidak perlu dimodernisasi; sebagian bahkan layak dihapus setelah dipastikan tidak dipakai.

```sql
-- Layar yang tidak pernah dipakai dalam setahun, dari tabel pencatatan
SELECT p.NAMA_PROGRAM, p.KETERANGAN
  FROM APPPRM.DAFTAR_LAYAR p
 WHERE NOT EXISTS (
       SELECT 1 FROM APPWRK.LOG_PEMAKAIAN l
        WHERE l.NAMA_PROGRAM = p.NAMA_PROGRAM
          AND l.WAKTU > CURRENT TIMESTAMP - 1 YEAR)
 ORDER BY p.NAMA_PROGRAM;
```

Bila institusi Anda belum mencatat pemakaian layar, menambahkannya adalah pekerjaan satu hari yang nilainya bertahan bertahun-tahun: satu panggilan ke utilitas pencatatan di awal setiap program.

Bagian LXI — Anatomi Insiden Lanjutan

Bagian XLVII memuat empat narasi insiden. Empat berikut melengkapinya, dipilih karena masing-masing memperlihatkan pola kegagalan yang berbeda dari yang sudah dibahas.

Bentuk narasi dipakai lagi karena alasan yang sama seperti sebelumnya: daftar penyebab dan tindakan mudah dilupakan, sedangkan cerita tentang orang yang membuat keputusan di bawah tekanan bertahan di ingatan. Yang diingat orang bukan bahwa mereka harus memeriksa lag replikasi, melainkan bahwa pernah ada malam ketika seseorang tidak memeriksanya.

Seluruh kejadian disamarkan dan digabungkan dari beberapa kasus. Kemiripan dengan institusi tertentu tidak disengaja.

🚀 Bab 212. Malam Ketika Angka Cocok Tetapi Salah

212.1 Pukul 22:15 — semuanya hijau

Fase rekonsiliasi EOD selesai pukul 22:15, lebih cepat sebelas menit dari rata-rata. Operator jaga, seorang staf bernama Wira yang sudah dua tahun memegang jadwal malam, melihat baris hijau di papan pemantauan dan mencentang daftar periksanya.

Selisih buku besar terhadap subsidiary: nol. Total debet sama dengan total kredit. Jumlah transaksi sesuai dengan yang tercatat di tabel kontrol. Tidak ada job yang berakhir tidak normal.

Ia mengirim pesan singkat ke grup: proses malam normal, akan lanjut ke fase laporan.

212.2 Pukul 23:40 — laporan selesai, semuanya masih hijau

Fase laporan menghasilkan seluruh berkas yang dijadwalkan. Laporan posisi harian, laporan transaksi kanal, dan berkas untuk regulator semuanya terbentuk dengan ukuran yang wajar.

Wira melakukan pemeriksaan terakhir, menjalankan date roll, dan menutup malam pukul 00:20. Dari sudut pandang mana pun, malam itu adalah malam yang baik.

212.3 Pukul 09:30 keesokan harinya — telepon pertama

Kepala cabang di sebuah kota menelepon bagian operasi. Seorang nasabah korporasi mengeluh bahwa transfer masuk sebesar dua miliar rupiah yang dikirim kemarin sore belum tercatat di rekeningnya.

Bagian operasi memeriksa. Transaksi itu ada di tabel transaksi, statusnya selesai, dengan stempel waktu

pukul 16:47. Tetapi saldo rekening tidak berubah.

Pukul 10:15, dua cabang lain melaporkan hal serupa. Semuanya transfer masuk dari satu bank yang sama, semuanya diterima setelah pukul 16:30.

212.4 Pukul 10:40 — penyelidikan dimulai

Tim yang dipanggil terdiri dari seorang System Engineer bernama Dimas dan seorang analis aplikasi.

Pemeriksaan pertama mereka adalah hal yang sama yang diperiksa Wira semalam: apakah buku besar seimbang. Jawabannya masih ya, nol selisih.

Ini yang membingungkan. Uang yang tidak masuk ke rekening nasabah seharusnya menghasilkan selisih di suatu tempat.

Dimas menjalankan kueri yang lebih rinci.

```
-- Transaksi masuk dari bank tertentu kemarin sore
SELECT t.NO_REF, t.NO_REK, t.NOMINAL, t.WAKTU_TERIMA,
       t.STATUS, t.KD_TRX,
       r.SALDO AS saldo_sekarang
  FROM DTALIB.TRX_MASUK t
  LEFT JOIN DTALIB.REKENING r ON r.NO_REK = t.NO_REK
 WHERE t.TGL = CURRENT DATE - 1 DAY
   AND t.KD_BANK = '014'
   AND t.WAKTU_TERIMA > '16:30:00'
 ORDER BY t.WAKTU_TERIMA;
```

Hasilnya menunjukkan empat puluh tujuh transaksi. Semuanya berstatus selesai. Semuanya punya nomor referensi.

Kemudian ia memeriksa buku besar.

```
-- Jurnal untuk referensi transaksi itu
SELECT NO_REF, KD_GL, DEBET, KREDIT, PGM_PENULIS, WAKTU
  FROM DTALIB.GLTRX
 WHERE NO_REF IN (SELECT NO_REF FROM DTALIB.TRX_MASUK
                  WHERE TGL = CURRENT DATE - 1 DAY
                    AND KD_BANK = '014'
                    AND WAKTU_TERIMA > '16:30:00');
```

Nol baris.

Empat puluh tujuh transaksi berstatus selesai, tidak satu pun punya jurnal. Dan buku besar tetap seimbang, karena transaksi itu tidak pernah masuk ke buku besar sama sekali — bukan masuk separuh.

212.5 Pukul 11:20 — akar masalahnya

Analis aplikasi menelusuri program yang memproses transaksi masuk dari bank itu. Ia menemukan perubahan yang diterapkan tiga hari sebelumnya: penambahan kode transaksi baru untuk jenis transfer tertentu yang diminta regulator.

Kode baru itu ditambahkan ke tabel parameter transaksi. Tetapi tabel pemetaan ke akun buku besar tidak diperbarui.

Program posting bekerja seperti ini: ia membaca kode transaksi, mencari pemetaan akun buku besar, dan bila tidak menemukan, ia mencatat pesan informasi dan melanjutkan ke transaksi berikutnya.

Melanjutkan. Bukan berhenti.

```
**free
// Kode yang menyebabkan masalah
exec sql SELECT KD_GL INTO :wKdGl FROM APPPRM.PETA_GL
          WHERE KD_TRX = :wKdTrx;

if sqlcode = 100;
   // Pemetaan tidak ditemukan
   catatInfo('Pemetaan GL tidak ada untuk ' + wKdTrx);
   iter;                    // <-- lanjut ke transaksi berikutnya
endif;
```

Pesan informasi itu memang tercatat. Empat puluh tujuh kali. Di job log, di antara ribuan pesan lain, tanpa ada yang membacanya.

212.6 Pukul 13:00 — perbaikan dan pertanyaan yang lebih besar

Perbaikan teknisnya sederhana: tambahkan pemetaan, jalankan program pemulihan yang memposting ulang empat puluh tujuh transaksi itu dengan tanggal buku kemarin, verifikasi saldo.

Selesai pukul 15:30, dengan koordinasi bagian akuntansi karena penyesuaiannya menyentuh tanggal buku yang sudah ditutup.

Pertanyaan yang lebih besar dibahas pada rapat keesokan harinya: mengapa seluruh sistem pemeriksaan melaporkan normal padahal ada uang nasabah yang tidak masuk?

212.7 Apa yang sebenarnya gagal

Ada tiga kegagalan yang bertumpuk, dan hanya yang pertama yang teknis.

Pertama, program memperlakukan kondisi yang tidak terduga sebagai kondisi yang bisa dilewati. Tidak menemukan pemetaan akun buku besar bukan situasi normal yang layak dicatat sebagai informasi; ia adalah kegagalan yang harus menghentikan proses.

Kedua, seluruh pemeriksaan rekonsiliasi membandingkan hal yang konsisten secara internal. Buku besar seimbang terhadap dirinya sendiri, dan subsidiary seimbang terhadap dirinya sendiri. Tidak ada pemeriksaan yang membandingkan jumlah transaksi yang diterima dengan jumlah transaksi yang diposting.

Ketiga, manajemen perubahan memperlakukan penambahan kode transaksi sebagai perubahan data, bukan perubahan yang berdampak. Tiket perubahan itu menyebutkan penambahan baris di tabel parameter, dan disetujui tanpa pertanyaan tentang tabel lain yang terkait.

212.8 Yang berubah setelahnya

Perbaikan yang diterapkan:

```
**free
// Perbaikan: kondisi tak terduga menghentikan proses
exec sql SELECT KD_GL INTO :wKdGl FROM APPPRM.PETA_GL
          WHERE KD_TRX = :wKdTrx;

if sqlcode = 100;
   kirimEscapeDicatat('POSTMASUK':
        'Transaksi ' + wNoRef:
        'Pemetaan GL tidak ada untuk kode ' + wKdTrx);
endif;
-- Kendali baru: jumlah diterima versus jumlah diposting
SELECT (SELECT COUNT(*) FROM DTALIB.TRX_MASUK
         WHERE TGL = CURRENT DATE - 1 DAY
           AND STATUS = 'S') AS diterima,
       (SELECT COUNT(DISTINCT NO_REF) FROM DTALIB.GLTRX
         WHERE TGL_BUKU = CURRENT DATE - 1 DAY
           AND SUMBER = 'TRXMASUK') AS diposting
  FROM SYSIBM.SYSDUMMY1;
```

Kendali ini menjadi bagian dari fase rekonsiliasi, dengan aturan: selisih tidak nol menghentikan EOD.

Dan aturan manajemen perubahan diperluas: setiap penambahan kode di tabel parameter menuntut daftar tabel terkait yang harus ikut diperiksa, ditandatangani oleh analis aplikasi.

Pelajaran yang paling layak diingat dari malam itu: sistem yang seluruh lampunya hijau belum tentu benar. Lampu hijau hanya berarti hal-hal yang diperiksa ternyata sesuai. Apa yang tidak diperiksa tetap tidak diketahui, dan dalam kasus ini yang tidak diperiksa adalah pertanyaan paling mendasar — apakah semua yang masuk sudah diproses.

Bab 213. Tiga Hari Mencari Sesuatu yang Tidak Ada

213.1 Senin pagi — keluhan yang tidak jelas

Bagian layanan nasabah melaporkan bahwa sejak akhir pekan, sebagian pembukaan rekening gagal dengan pesan kesalahan yang tidak dikenali. Tidak semua. Perkiraan mereka satu dari sepuluh.

Pesan yang muncul di layar teller: Terjadi kesalahan sistem. Hubungi administrator.

Seorang System Engineer bernama Rani mengambil tiket itu. Langkah pertamanya adalah mencari pesan yang sebenarnya di job log.

```
SELECT MESSAGE_ID, MESSAGE_TEXT, MESSAGE_SECOND_LEVEL_TEXT,
       MESSAGE_TIMESTAMP, FROM_PROGRAM, FROM_LIBRARY
  FROM TABLE(QSYS2.JOBLOG_INFO('123456/TELLER01/QPADEV0003'))
 WHERE SEVERITY >= 30
 ORDER BY MESSAGE_TIMESTAMP DESC;
```

Pesan aslinya: CPF5029 - Data mapping error on member CIF_UTAMA.

213.2 Senin siang — hipotesis pertama

Pesan CPF5029 menunjuk ke kesalahan pemetaan data, yang biasanya berarti ada nilai di kolom yang tidak sesuai dengan definisinya. Rani memeriksa tabel.

```
-- Cari nilai numerik yang tidak valid
SELECT ID_NASABAH, HEX(KD_CABANG) AS heks_cabang, KD_CABANG
  FROM DTALIB.CIF_UTAMA
 WHERE ID_NASABAH IN (SELECT ID_NASABAH FROM APPWRK.LOG_GAGAL
                      WHERE TGL >= CURRENT DATE - 3 DAYS);
```

Seluruh nilai tampak normal. Tidak ada karakter aneh, tidak ada nilai di luar rentang.

Ia memeriksa definisi tabel dan membandingkannya dengan definisi di program. Cocok. Ia memeriksa apakah ada perubahan pada tabel dalam sepekan terakhir. Tidak ada.

213.3 Selasa — hipotesis kedua, dan ketiga

Rani mengganti arah. Bila datanya benar dan definisinya cocok, mungkin masalahnya di program.

Ia membandingkan program di produksi dengan sumbernya.

```
SELECT PROGRAM_NAME, PROGRAM_LIBRARY,
       SOURCE_FILE, SOURCE_LIBRARY, SOURCE_MEMBER,
       SOURCE_TIMESTAMP, CREATION_TIMESTAMP
  FROM QSYS2.PROGRAM_INFO
 WHERE PROGRAM_LIBRARY = 'APPPGM'
   AND PROGRAM_NAME = 'CIF002';
```

Stempel waktu sumber lebih lama daripada stempel waktu kompilasi. Artinya program dikompilasi dari sumber yang ada. Cocok.

Ia memeriksa apakah ada override yang aktif. Tidak ada. Ia memeriksa library list job teller. Normal.

Selasa sore, setelah tujuh jam, ia belum menemukan apa pun.

213.4 Rabu pagi — pertanyaan yang mengubah arah

Pada rapat singkat pagi itu, seorang rekan bertanya hal yang tampak tidak relevan: “Kegagalannya satu dari sepuluh, atau satu dari sepuluh cabang?”

Rani menyadari ia tidak pernah memeriksa itu. Ia mengasumsikan kegagalan tersebar acak karena bagian layanan nasabah menyebutnya begitu.

```
-- Distribusi kegagalan per cabang
SELECT KD_CABANG, COUNT(*) AS jml_gagal
  FROM APPWRK.LOG_GAGAL
 WHERE TGL >= CURRENT DATE - 4 DAYS
 GROUP BY KD_CABANG
 ORDER BY jml_gagal DESC;
```

Hasilnya bukan sebaran acak. Seluruh kegagalan berasal dari tiga cabang, dan ketiganya berada di satu wilayah.

213.5 Rabu siang — penyebab yang sebenarnya

Tiga cabang itu punya satu kesamaan: semuanya baru dipindahkan ke jalur jaringan baru pada akhir pekan, bagian dari proyek peremajaan jaringan yang dikerjakan tim infrastruktur.

Jalur baru itu melewati perangkat yang melakukan inspeksi paket. Perangkat itu, dalam konfigurasi bawaannya, memodifikasi sebagian aliran data Telnet yang dianggapnya tidak sesuai standar.

Aliran data 5250 membawa field dalam format yang spesifik. Modifikasi sekecil apa pun merusaknya, dan

IBM i melaporkannya sebagai kesalahan pemetaan data — pesan yang menunjuk ke basis data, padahal penyebabnya di jaringan.

Kegagalan satu dari sepuluh terjadi karena perangkat itu hanya memodifikasi paket yang melebihi ukuran tertentu, dan itu tergantung panjang nama nasabah yang diketik.

213.6 Apa yang membuat penyelidikan ini panjang

Rani menghabiskan dua hari penuh mencari masalah di tempat yang salah. Beberapa hal yang menyebabkannya:

Pesan kesalahan menunjuk ke arah yang keliru. CPF5029 menyebut pemetaan data pada member tabel. Setiap orang yang membacanya akan mulai dari basis data. Pesan itu tidak salah secara teknis — pemetaan data memang gagal — tetapi penyebabnya berada tiga lapisan di bawah.

Deskripsi masalah diterima tanpa diverifikasi. "Satu dari sepuluh" terdengar seperti sebaran acak, dan itu membentuk seluruh arah penyelidikan. Satu kueri distribusi pada jam pertama akan memotong dua hari.

Perubahan di luar TI tidak terlihat. Pemindahan jalur jaringan tercatat di tiket tim infrastruktur, bukan di tiket perubahan yang biasa dibaca tim aplikasi. Tidak ada satu tempat yang memuat seluruh perubahan.

213.7 Yang berubah setelahnya

Tiga perubahan proses diterapkan.

Pertama, daftar periksa penyelidikan insiden dimulai dengan pertanyaan distribusi: siapa yang terpengaruh, dan apakah ada polanya. Sebelum hipotesis apa pun dibuat.

Kedua, seluruh perubahan infrastruktur yang menyentuh jalur ke sistem inti dicatat di kalender perubahan yang sama dengan perubahan aplikasi. Tim yang menyelidiki insiden bisa melihat satu daftar.

Ketiga, tabel rujukan pesan kesalahan diperluas dengan kolom "penyebab yang pernah ditemukan", sehingga CPF5029 kini punya catatan bahwa ia bisa berasal dari gangguan jaringan, bukan hanya dari data.

```
CREATE TABLE APPPRM.RUJUKAN_PESAN (
    MESSAGE_ID   CHAR(7)      NOT NULL,
    URAIAN       VARCHAR(200) NOT NULL,
    PENYEBAB     VARCHAR(500) NOT NULL,
    TGL_DITEMUKAN DATE        NOT NULL,
    NO_INSIDEN   VARCHAR(20),
    PRIMARY KEY (MESSAGE_ID, TGL_DITEMUKAN)
);
```

Pelajaran yang saya anggap paling berharga: pesan kesalahan memberi tahu apa yang gagal, bukan mengapa. Lapisan tempat gejala muncul jarang sama dengan lapisan tempat penyebabnya berada, dan penyelidikan yang berhenti di lapisan pertama akan menghabiskan waktu di tempat yang salah.

💾 Bab 214. Backup yang Berhasil Setiap Malam Selama Delapan Bulan

214.1 Latar

Institusi ini menjalankan backup harian lewat BRMS. Laporan pagi dikirim otomatis ke kotak surat tim operasi, dan setiap pagi isinya sama: seluruh grup berhasil, tidak ada kesalahan.

Delapan bulan berjalan tanpa keluhan.

214.2 Hari ketika ia dibutuhkan

Sebuah perubahan aplikasi yang diterapkan pada Jumat malam ternyata mengandung kesalahan yang menghapus baris pada tabel parameter produk. Baru ketahuan Senin pagi, ketika perhitungan bunga menghasilkan angka nol untuk sebagian produk.

Tindakan yang jelas: pulihkan tabel itu dari backup Jumat malam.

```
STRRCYBRM OPTION(*RESTORE) ACTION(*SELECT)
```

Seorang engineer bernama Anwar menjalankan pemulihan. Sistem melaporkan bahwa objek tidak ditemukan dalam media backup.

Ia mengira salah memilih tanggal. Ia mencoba Kamis. Tidak ditemukan. Rabu. Tidak ditemukan.

214.3 Pukul 09:20 — kesadaran yang tidak menyenangkan

Anwar memeriksa apa yang sebenarnya ada di dalam backup.

```
-- Objek yang tercatat dalam riwayat backup BRMS
SELECT SAVE_ITEM, SAVE_LIBRARY, SAVE_DATE, SAVE_TYPE,
       VOLUME_SERIAL, OBJECTS_SAVED
  FROM QUSRBRM.QA1AHS
 WHERE SAVE_DATE >= CURRENT DATE - 30 DAYS
   AND SAVE_LIBRARY = 'APPPRM'
 ORDER BY SAVE_DATE DESC;
```

Nol baris. Library APPPRM tidak pernah masuk backup.

Pemeriksaan berikutnya menunjukkan cakupan yang sebenarnya.

```
-- Library di sistem yang tidak ada di riwayat backup
SELECT o.OBJNAME AS library
  FROM TABLE(QSYS2.OBJECT_STATISTICS('QSYS','*LIB')) o
 WHERE o.OBJNAME NOT LIKE 'Q%'
   AND NOT EXISTS (
       SELECT 1 FROM QUSRBRM.QA1AHS h
        WHERE h.SAVE_LIBRARY = o.OBJNAME
          AND h.SAVE_DATE >= CURRENT DATE - 7 DAYS);
```

Empat library tidak pernah masuk backup: APPPRM, APPWRK, UTILLIB, dan SECLIB.

Keempatnya dibuat delapan bulan sebelumnya, sebagai bagian dari proyek pemisahan library aplikasi.

214.4 Mengapa laporan selalu berhasil

Kontrol grup BRMS didefinisikan dengan daftar library eksplisit. Ketika library baru dibuat, tidak ada yang menambahkannya ke kontrol grup.

BRMS melakukan persis apa yang diperintahkan: mencadangkan library yang terdaftar, seluruhnya berhasil, dan melaporkan berhasil.

Laporannya benar. Cakupannya yang salah.

214.5 Pemulihan

Data akhirnya dipulihkan dari sumber lain: sebuah salinan tabel parameter yang kebetulan dibuat seorang analis tiga minggu sebelumnya untuk keperluan pengujian, disimpan di library pengembangan yang masuk dalam cakupan backup.

Salinan itu tidak memuat tiga perubahan parameter yang dilakukan setelahnya. Ketiganya direkonstruksi dari tiket perubahan dan diverifikasi bersama bagian produk.

Layanan pulih pukul 14:40. Perhitungan bunga untuk hari Senin dijalankan ulang.

Institusi itu beruntung, dan semua orang di ruangan itu tahu.

214.6 Apa yang gagal

Kontrol grup backup memakai daftar eksplisit tanpa mekanisme yang menangkap objek baru. Ini pilihan yang wajar dan umum, tetapi ia menuntut disiplin: setiap library baru harus didaftarkan. Disiplin itu bergantung pada seseorang mengingat, dan orang lupa.

Tidak ada pemeriksaan yang membandingkan apa yang ada di sistem dengan apa yang di-backup. Laporan backup menjawab pertanyaan “apakah yang dijadwalkan berhasil”, bukan “apakah semuanya tercakup”. Dua pertanyaan yang berbeda, dan hanya yang pertama yang dijawab.

Uji restore tidak pernah dilakukan untuk library ini. Uji restore triwulanan yang berjalan memakai library yang sama setiap kali, dipilih karena ukurannya kecil dan cepat. Library yang tidak pernah diuji adalah library yang tidak pernah dibuktikan bisa dipulihkan.

214.7 Yang berubah setelahnya

Kendali baru, dijalankan mingguan, menjadi salah satu dari lima belas kendali yang dibahas di Bab 199.

```
-- K-16: Library di sistem yang tidak tercakup backup 7 hari terakhir
WITH tidak_tercakup AS (
  SELECT o.OBJNAME
    FROM TABLE(QSYS2.OBJECT_STATISTICS('QSYS','*LIB')) o
   WHERE o.OBJNAME NOT LIKE 'Q%'
     AND o.OBJNAME NOT IN (SELECT NAMA FROM AUDITLIB.LIB_DIKECUALIKAN)
     AND NOT EXISTS (
         SELECT 1 FROM QUSRBRM.QA1AHS h
          WHERE h.SAVE_LIBRARY = o.OBJNAME
            AND h.SAVE_DATE >= CURRENT DATE - 7 DAYS)
)
SELECT COUNT(*) AS jml_tidak_tercakup,
       LISTAGG(OBJNAME, ', ') AS daftar
  FROM tidak_tercakup;
```

Tabel LIB_DIKECUALIKAN memuat library yang memang sengaja tidak di-backup, masing-masing dengan alasan tertulis dan pemilik. Ini penting: tanpa daftar pengecualian yang eksplisit, kendali akan menghasilkan temuan palsu setiap pekan dan lama-lama diabaikan.

Selain itu, prosedur pembuatan library baru diperluas dengan satu langkah wajib: daftarkan ke kontrol grup BRMS atau ke daftar pengecualian, sebelum library dipakai.

Dan uji restore triwulanan diubah menjadi bergilir: library yang diuji berbeda setiap triwulan, dipilih dari

daftar, sehingga dalam tiga tahun seluruh library pernah diuji sedikitnya sekali.

Pelajaran yang paling penting: backup yang berhasil tidak sama dengan data yang terlindungi. Yang membedakannya adalah cakupan, dan cakupan hanya bisa diketahui dengan membandingkan apa yang ada terhadap apa yang disimpan — bukan dengan membaca laporan keberhasilan.

Bab 215. Perbaikan yang Memperburuk Keadaan

215.1 Pukul 20:10 — masalah pertama

Fase posting EOD berjalan lebih lambat dari biasanya. Setelah lima puluh menit, kemajuannya baru tiga puluh persen, sementara biasanya sudah tujuh puluh.

Engineer yang berjaga malam itu, seorang senior bernama Hendra, memeriksa job.

Status LCKW. Job posting menunggu kunci record.

```
SELECT JOB_NAME, LOCK_STATE, TABLE_SCHEMA, TABLE_NAME,
       RELATIVE_RECORD_NUMBER
  FROM QSYS2.RECORD_LOCK_INFO
 WHERE TABLE_SCHEMA = 'DTALIB';
```

Hasilnya menunjukkan sebuah job interaktif yang menahan kunci pada tabel rekening. Nama jobnya menunjuk ke sesi teller di sebuah cabang.

215.2 Pukul 20:25 — keputusan pertama

Hendra memeriksa sesi itu. Job dalam status DSPW, menunggu masukan di layar, sudah selama empat jam.

Kesimpulannya masuk akal: seorang teller pulang tanpa keluar dari aplikasi, meninggalkan transaksi setengah jadi yang menahan kunci.

Ia mengakhiri job itu.

```
ENDJOB JOB(345678/TELLER14/QPADEV0021) OPTION(*IMMED)
```

Kunci terlepas. Job posting melanjutkan. Kemajuan naik.

215.3 Pukul 20:50 — masalah kedua

Kemajuan berhenti lagi, kali ini di angka empat puluh dua persen. Status kembali LCKW.

Kali ini penahannya job interaktif yang berbeda, dari cabang lain.

Hendra mengakhirinya juga.

Pukul 21:15, kejadian yang sama untuk ketiga kalinya. Ia mengakhiri job ketiga.

215.4 Pukul 21:40 — masalah yang sebenarnya

Setelah job ketiga diakhiri, posting berjalan sebentar lalu berhenti dengan pesan kesalahan. Bukan menunggu kunci — berhenti.

Job log menunjukkan CPF5027: record sedang dipakai job lain, dan setelah itu serangkaian pesan kegagalan pembaruan.

Hendra mulai merasa ada yang tidak beres. Ia memeriksa lebih dalam, dan menemukan bahwa ketiga job yang ia akhiri bukan sesi teller yang ditinggalkan.

Ketiganya adalah job yang dijalankan oleh sebuah proses integrasi baru, yang memakai profil dengan nama menyerupai profil teller karena pemetaan identitas yang dibuat tiga pekan sebelumnya. Job-job itu dalam status DSPW karena program yang dipanggilnya memang menunggu, sebagai bagian dari pola antrean yang dirancang demikian.

Mengakhirinya di tengah transaksi meninggalkan pembaruan setengah jadi yang tidak di-rollback, karena program integrasi itu tidak memakai commitment control.

215.5 Pukul 22:00 sampai 02:30 — pemulihan

Empat setengah jam berikutnya dihabiskan untuk memperbaiki keadaan yang lebih buruk daripada masalah awal.

Yang harus dikerjakan:

Identifikasi seluruh record yang diubah oleh ketiga job sebelum diakhiri

Tentukan mana yang lengkap dan mana yang setengah jadi

Kembalikan yang setengah jadi ke keadaan semula

Jalankan ulang transaksi yang terpotong

Verifikasi keseimbangan sebelum melanjutkan EOD

Langkah pertama memakai jurnal.

```
-- Perubahan oleh job yang diakhiri
SELECT ENTRY_TIMESTAMP, JOB_NAME, JOURNAL_CODE, ENTRY_TYPE,
       OBJECT_NAME, RELATIVE_RECORD_NUMBER
  FROM TABLE(QSYS2.DISPLAY_JOURNAL('DTALIB','QSQJRN',
         STARTING_TIMESTAMP => '2026-09-18-20.00.00.000000'))
 WHERE JOB_NAME IN ('TELLER14','TELLER22','TELLER07')
 ORDER BY ENTRY_TIMESTAMP;
```

EOD baru selesai pukul 05:10. Layanan buka terlambat empat puluh menit.

215.6 Apa yang salah dari keputusannya

Keputusan Hendra mengakhiri job pertama tidak keliru berdasarkan informasi yang ia punya. Yang keliru adalah bahwa ia tidak menambah informasi sebelum mengulanginya dua kali lagi.

Kegagalan yang berulang adalah tanda bahwa diagnosisnya salah. Ketika masalah yang sama muncul untuk kedua kalinya setelah “diperbaiki”, itu adalah saat untuk berhenti dan menyelidiki, bukan untuk mengulangi tindakan yang sama.

Bila ia berhenti setelah kejadian kedua dan menjalankan satu kueri saja, ia akan melihatnya:

```
-- Profil apa yang sebenarnya menjalankan job ini
SELECT JOB_NAME, AUTHORIZATION_NAME, CLIENT_IP_ADDRESS,
       JOB_TYPE, SUBSYSTEM, FUNCTION, JOB_DESCRIPTION
  FROM TABLE(QSYS2.ACTIVE_JOB_INFO(DETAILED_INFO => 'ALL'))
 WHERE JOB_NAME = 'TELLER22';
```

Kolom alamat IP klien akan menunjukkan alamat server integrasi, bukan alamat cabang. Satu kueri, sepuluh detik.

215.7 Yang berubah setelahnya

Penamaan profil dipisahkan secara tegas. Profil untuk proses otomatis diberi awalan SVC, dan tidak boleh menyerupai profil manusia. Ini perubahan yang memakan waktu dua bulan untuk diterapkan penuh dan menghilangkan seluruh kelas kebingungan.

Runbook penanganan kunci diperluas dengan langkah verifikasi sebelum mengakhiri job mana pun: periksa profil, alamat IP, job description, dan program yang sedang dijalankan. Empat kolom dari satu kueri.

Aturan tiga kali ditetapkan: bila tindakan yang sama dibutuhkan untuk ketiga kalinya dalam satu malam, hentikan dan eskalasi. Pengulangan menandakan diagnosis yang salah, dan pada pukul sembilan malam kemampuan seseorang mendiagnosis ulang sendiri sudah menurun.

Dan program integrasi itu diperbaiki agar memakai commitment control, sehingga pengakhiran job di tengah transaksi akan menghasilkan rollback, bukan data setengah jadi.

Bab 216. Pola yang Muncul dari Delapan Narasi

Dengan empat narasi di Bagian XLVII dan empat di sini, ada pola yang cukup jelas untuk disebutkan.

216.1 Pemeriksaan menjawab pertanyaan yang tidak diajukan

Pada Bab 212, seluruh rekonsiliasi seimbang karena yang diperiksa adalah konsistensi internal, bukan kelengkapan. Pada Bab 214, laporan backup berhasil karena yang diperiksa adalah keberhasilan tugas, bukan cakupan.

Keduanya kasus yang sama dalam bentuk berbeda: **pemeriksaan yang menjawab pertanyaan yang lebih mudah daripada pertanyaan yang sebenarnya penting.**

Cara mendeteksinya: untuk setiap kendali yang Anda punya, tuliskan pertanyaan yang benar-benar ingin dijawab, lalu tanyakan apakah kendali itu betul-betul menjawabnya. Sering jawabannya tidak.

216.2 Gejala muncul jauh dari penyebab

Pada Bab 213, gangguan jaringan muncul sebagai pesan kesalahan basis data. Pada Bab 155 di Bagian XLVII, masalah MTU muncul sebagai keluhan aplikasi lambat.

Lapisan tempat gejala terlihat jarang sama dengan lapisan tempat penyebab berada. Penyelidikan yang berhenti di lapisan pertama akan menghabiskan waktu di tempat yang salah, kadang berhari-hari.

Cara mengurangi: mulai setiap penyelidikan dengan pertanyaan distribusi. Siapa yang terpengaruh, dan apakah ada polanya. Pola dalam distribusi hampir selalu menunjuk ke lapisan yang benar.

216.3 Tindakan di bawah tekanan memperburuk

Pada Bab 215, tiga tindakan yang masing-masing masuk akal menghasilkan keadaan yang jauh lebih buruk. Pada Bab 154, hampir terjadi hal serupa dan hanya dicegah oleh aturan dua orang.

Keputusan yang diambil di bawah tekanan waktu dan dalam keadaan lelah punya tingkat kekeliruan yang jauh lebih tinggi, dan yang paling berbahaya adalah keputusan yang terasa jelas.

Cara mengurangi: aturan yang ditetapkan sebelumnya, bukan pertimbangan yang dilakukan saat itu. Aturan tiga kali, aturan dua orang, batas waktu eskalasi. Semuanya bekerja karena ia menghilangkan kebutuhan menilai pada saat kemampuan menilai sedang terendah.

216.4 Yang tidak pernah diuji tidak pernah bekerja

Pada Bab 214, library yang tidak pernah diuji restore ternyata tidak ada di backup. Pada Bab 157, prosedur DR yang tidak pernah diuji penuh ternyata punya celah pada daftar IP.

Kemampuan yang tidak pernah dibuktikan adalah kemampuan yang belum tentu ada. Ini berlaku untuk backup, untuk DR, untuk prosedur darurat, dan untuk pengetahuan personel.

Cara mengurangi: pengujian bergilir, sehingga setiap hal pernah diuji dalam siklus yang wajar. Menguji hal yang sama setiap kali memberi rasa aman tanpa memberi bukti.

216.5 Memakai narasi ini untuk melatih tim

Cara yang saya sarankan, sama seperti di Bab 158:

Bacakan narasi sampai titik keputusan, lalu berhenti. Tanyakan apa yang akan mereka lakukan. Biarkan diskusi berjalan tanpa dikoreksi. Baru kemudian bacakan apa yang sebenarnya terjadi.

Titik berhenti yang saya sarankan untuk empat narasi di bagian ini:

Bab

Berhenti di

Pertanyaan

212

Pukul 10:40, sebelum kueri rinci

Semua seimbang tapi uang tidak masuk. Apa yang diperiksa?

213

Selasa sore, setelah dua hari buntu

Anda buntu. Apa yang belum dicoba?

214

Pukul 09:20, sebelum kueri cakupan

Restore gagal tiga tanggal berturut. Apa artinya?

215

Pukul 20:50, sebelum job kedua diakhiri

Masalah yang sama muncul lagi. Apa yang Anda lakukan?

Pertanyaan terakhir adalah yang paling berharga, dan hampir semua orang yang saya ajak berdiskusi menjawab “akhiri job itu juga” — yang persis merupakan keputusan yang membawa pada empat setengah jam pemulihan.

Menyadari bahwa mereka akan mengambil keputusan yang sama adalah pelajaran yang jauh lebih kuat daripada diberi tahu bahwa keputusan itu keliru.

📖 Indeks Subjek

Indeks ini disusun dari isi buku dan memuat istilah teknis, perintah CL, IBM i Services, kode pesan, serta konsep operasional. Nomor yang tercantum adalah nomor halaman sebagaimana tercetak di kaki halaman.

Istilah yang muncul di lebih dari tiga puluh halaman sengaja tidak dimasukkan, karena entri semacam itu tidak membantu pencarian. Untuk istilah yang sangat umum seperti nama platform atau nama modul, gunakan daftar isi atau daftar runbook di Bab 206.

A

ACCESS — 62, 151, 362, 561

ACCPTH — 50, 77, 330, 561

ACTGRP — 81, 112, 117, 119, 123, 201, 281, 283, 360, 365, 495, 561

ACTION — 228, 555, 561

ACTIVATE — 62, 151, 561

activation group — 111, 112, 120, 123, 130, 283, 324, 360, 363, 366, 369, 370, 371, 386, 388, 389, 408, 432, 495, 561

ACTIVE — 105, 190, 341, 476, 498, 561

ACTLVL — 28, 84, 197, 214, 335, 386, 467, 475, 476, 561

ADDAUTLE — 300, 561

ADDBNDDIRE — 117, 303, 561

ADDCADMRE — 144, 484, 561

ADDCMNE — 299, 561

ADDCOLNAM — 240, 561

ADDEXITPGM — 61, 137, 187, 301, 561

ADDJOBQE — 27, 28, 299, 474, 561

ADDJOBSCDE — 51, 110, 198, 299, 373, 479, 561

ADDLIBLE — 22, 297, 298, 307, 562

ADDMSGD — 303, 562

ADDPFM — 122, 300, 562

ADDPJE — 299, 562

ADDRDBDIRE — 271, 562

ADDRTGE — 27, 28, 299, 475, 562

ADDTCPHTE — 180, 302, 562

ADDTCPIFC — 42, 178, 179, 301, 562

ADDTCPRTE — 42, 176, 301, 562

ADDWSE — 299, 562

ADLJOBS — 215, 216, 477, 562

ADMDMN — 484, 562

ADMDMN1 — 484, 562

ADMDOMAIN — 144, 562

ADMIN — 42, 187, 562

adopted authority — 386, 562

AFOBJ — 61, 562

AFPDS — 545, 562

AFPGM — 61, 562

AFTER — 132, 562

AFUSER — 61, 562

AFVIOL — 61, 562

AGGPCY — 179, 562

AGGREGATE — 179, 562

AGGRSRCL — 179, 562

AKHIR — 349, 350, 351, 352, 353, 354, 355, 356, 357, 562

akhir hari — 397, 440, 562

AKRUAL — 154, 421, 422, 423, 562

AKRUL — 422, 562

AKSES — 233, 347, 562

AKTIF — 111, 136, 168, 287, 288, 292, 349, 352, 479, 500, 504, 562

aktif-aktif — 485, 486, 487, 563

aktif-pasif — 486, 487, 563

AKTIVITAS — 261, 263, 563

akun layanan — 61, 68, 101, 199, 453, 454, 503, 563

ALAMAT — 67, 563

alamat IP — 61, 102, 137, 168, 173, 177, 178, 185, 207, 233, 246, 261, 262, 283, 347, 391, 403, 405, 559, 563

ALASAN — 138, 268, 270, 430, 431, 563

ALCOBJ — 33, 35, 245, 298, 313, 429, 563

ALERT — 48, 563

ALGORITMA — 348, 350, 351, 352, 354, 355, 356, 563

aliran data — 66, 434, 532, 537, 546, 553, 563

ALLDIR — 226, 563

ALLOBJ — 25, 45, 46, 60, 62, 85, 141, 279, 282, 285, 336, 381, 386, 390, 402, 409, 503, 520, 563

ALLOW — 422, 563

ALLUSR — 50, 95, 105, 206, 208, 246, 297, 310, 311, 372, 400, 412, 424, 563

ALLWS — 219, 220, 340, 563

ALWCANCEL — 85, 94, 103, 456, 563

ALWCLNUP — 225, 563

ALWOBJDIF — 231, 563

ALWSSL — 186, 187, 189, 563

AMBANG — 287, 288, 289, 291, 292, 563

ambang batas — 21, 563

AMBIL — 363, 364, 563

AMBILKRED — 454, 563

ANALISIS — 218, 563

analisis akar masalah — 247, 521, 563

analisis tunggu — 213, 468, 563

ANGKA — 348, 563

angka kontrol — 233, 247, 340, 342, 344, 348, 383, 402, 404, 406, 418, 440, 563

antrean pesan — 19, 35, 303, 305, 387, 563

ANZDFTPWD — 301, 563

ANZOBJCVN — 206, 302, 564

APLIKASI — 288, 361, 454, 564

APPBND — 111, 117, 202, 360, 564

APPGRP — 111, 117, 201, 359, 360, 365, 564

APPLIB — 12, 19, 22, 32, 43, 46, 48, 51, 105, 564

APPPRM — 69, 74, 77, 108, 116, 134, 151, 153, 165, 168, 215, 267, 316, 320, 322, 480, 497, 516, 549, 551, 552, 554, 555, 564

APPSRC — 115, 117, 200, 202, 221, 257, 491, 492, 494, 564

APPSRV — 494, 495, 564

APYJRNCHG — 78, 94, 98, 232, 300, 308, 375, 456, 564

APYPTF — 302, 564

ARSIP — 225, 238, 564

arsitektur berbasis objek — 3, 18, 308, 564

ARTIFACT — 203, 564

ASCII — 234, 235, 387, 434, 564

ASEAN — 65, 391, 564

ASSIGN — 362, 564

ATTACHED — 98, 564

aturan tiga kali — 559, 564

AUDAF — 60, 61, 564

AUDIT — 45, 290, 291, 337, 564

audit mandiri — 505, 564

AUDITLIB — 500, 501, 504, 507, 508, 556, 564

AUDITLOG — 111, 564

AUDITQ — 272, 564

AUDLVL — 41, 141, 262, 279, 337, 457, 564

AUDRCV01 — 260, 564

AUTFAIL — 41, 60, 457, 564

AUTHOR — 361, 565

authorization list — 64, 282, 300, 306, 336, 386, 485, 565

AUTOSTART — 108, 178, 186, 187, 189, 334, 482, 565

AVAIL — 51, 100, 228, 229, 334, 565

AXELOS — 460, 565

B

BACKUP — 100, 334, 427, 482, 565

backup penuh — 41, 206, 207, 440, 565

BACKUP01 — 374, 565

BACKUPHRN — 51, 565

BAGIAN — 30, 565

bahasa akibat — 510, 565

BANDING — 208, 565

BARIS — 258, 267, 543, 565

BASELINE — 43, 191, 218, 565

baseline konfigurasi — 285, 382, 414, 458, 565

BASIC — 43, 235, 565

BATASDETIK — 428, 429, 565

BATCH — 422, 423, 565

batch window — 28, 57, 76, 242, 321, 323, 330, 381, 386, 390, 470, 509, 511, 565

BCKDAILY — 51, 565

BCKSTATUS — 34, 565

BEGIN — 132, 133, 228, 565

BENAR — 350, 351, 356, 357, 565

BERHENTI — 337, 565

BERIKUTNYA — 249, 349, 565

BERJALAN — 349, 565

BERKAS — 239, 565

berkas salinan — 491, 492, 565

BERKASOUT — 241, 565

BINARY — 236, 367, 566

binder language — 117, 386, 391, 494, 566

BINDIFC — 176, 566

BLOKIR — 111, 566

BNDDIR — 117, 360, 566

BNDSRVPGM — 81, 566

BOLEH — 77, 566

BREAK — 82, 326, 366, 566

BREXX — 422, 566

BTIKET123 — 332, 566

BUDI001 — 46, 144, 235, 484, 566

BUILD — 201, 203, 566

BUKTI — 264, 507, 566

buku besar — 159, 358, 361, 380, 418, 516, 549, 550, 551, 566

BULANAN — 228, 504, 566

BUNGA — 84, 316, 317, 318, 320, 362, 363, 366, 566

BUNGALOG — 80, 97, 154, 328, 566

BUNGALOGR — 80, 566

C

CABUT — 455, 566

CACHE — 313, 566

CAKUPAN — 263, 566

cakupan backup — 545, 556, 566

call stack — 82, 123, 245, 284, 327, 468, 566

CALLER — 495, 566

CAPACITY — 214, 314, 566

CATAT — 349, 566

CATATAN — 267, 566

CATEGORY — 463, 566

CCSID — 6, 9, 187, 200, 234, 235, 236, 237, 240, 244, 257, 305, 327, 381, 386, 434, 516, 517, 566

CEKAPPR — 423, 567

CEKEOD — 322, 567

CEKSALDO — 87, 360, 567

CETAK — 238, 239, 519, 545, 567

CETAKSALDO — 258, 567

CFGOBJ — 146, 177, 185, 334, 342, 345, 482, 567

CFGPFRCOL — 212, 567

CFGTCP — 178, 180, 301, 567

CFGTYPE — 145, 146, 147, 177, 178, 185, 334, 342, 345, 567

CHAIN — 88, 113, 114, 118, 119, 120, 152, 280, 328, 376, 567

CHANGELOG — 200, 567

CHECKSUM — 163, 445, 567

CHGAUT — 304, 567

CHGCLNUP — 225, 302, 567

CHGCLS — 27, 299, 567

CHGCLUNODE — 345, 567

CHGCMD — 424, 567

CHGCRGPRI — 146, 341, 345, 567

CHGCURLIB — 299, 567

CHGDSPF — 543, 567

CHGDTAARA — 34, 313, 326, 567

CHGFTPA — 186, 187, 302, 567

CHGJOB — 79, 99, 170, 283, 297, 299, 331, 513, 567

CHGJOBD — 298, 299, 567

CHGJOBQE — 27, 299, 335, 567

CHGJOBSCDE — 299, 567

CHGJRN — 300, 337, 567

CHGLIB — 298, 567

CHGLIBL — 22, 298, 567

CHGLINETH — 302, 567

CHGMSGQ — 303, 567

CHGNTPA — 334, 568

CHGOBJAUD — 141, 151, 298, 568

CHGOBJAUT — 260, 568

CHGOBJD — 298, 568

CHGOBJOWN — 298, 454, 568

CHGOWN — 304, 568

CHGPFCST — 330, 568

CHGPGM — 422, 454, 568

CHGPJE — 215, 299, 477, 568

CHGQRYA — 215, 243, 568

CHGRTGE — 299, 568

CHGSBSD — 299, 568

CHGSHRPOOL — 27, 28, 84, 197, 214, 299, 335, 475, 568

CHGSPLFA — 238, 302, 568

CHGSTS — 345, 568

CHGSYSLIBL — 299, 568

CHGSYSVAL — 41, 60, 102, 139, 186, 214, 215, 301, 337, 568

CHGTCPDMN — 42, 180, 301, 568

CHGTCPHTE — 180, 302, 568

CHGTCPIFC — 178, 179, 301, 568

CHGTCPRTE — 176, 301, 568

CHGTCPSVR — 482, 568

CHGTELNA — 186, 302, 568

CHGUSRAUD — 141, 262, 279, 284, 300, 568

CHGUSRPRF — 262, 279, 300, 568

CHGVAR — 29, 30, 32, 34, 48, 68, 77, 84, 313, 368, 373, 425, 426, 427, 428, 429, 430, 483, 568

CHKCOLSRV — 55, 568

CHKDMN — 263, 568

CHKOBJ — 34, 68, 167, 282, 298, 428, 568

CHKOBJITG — 263, 298, 568

CHKPGMSIG — 263, 568

CHKTAP — 334, 569

CHOICE — 422, 569

CHRSIZ — 258, 543, 569

CIF001 — 530, 543, 569

CIF002 — 553, 569

circuit breaker — 162, 276, 515, 569

CLEAR — 77, 427, 569

CLOSQLCSR — 360, 365, 432, 569

CLRJOBQ — 299, 569

CLRLIB — 298, 569

CLRPFM — 300, 569

CLRSAVF — 301, 569

CLSTR1 — 99, 144, 145, 146, 341, 344, 345, 482, 484, 569

CLUSTER — 99, 144, 145, 146, 341, 344, 345, 482, 484, 569

cluster resource group — 481, 482, 569

CMDLBL — 29, 32, 34, 68, 84, 170, 429, 569

CMDSTR — 423, 569

CMN03 — 177, 179, 569

CMN04 — 179, 569

CMPVAL — 28, 475, 569

CMTSCOPE — 119, 569

CNTRLD — 97, 99, 220, 222, 331, 341, 483, 569

COBIT — 460, 569

COBOL — 8, 357, 358, 361, 363, 364, 365, 366, 367, 368, 369, 370, 371, 380, 388, 569

COLHDG — 253, 569

COLLIB — 469, 569

COLOR — 254, 255, 530, 536, 569

COLPRF — 213, 569

COLSRVPRF — 55, 212, 569

COLSVCPFN — 463, 569

COLUMN — 62, 139, 150, 151, 271, 569

COMMA — 362, 570

commitment control — 5, 49, 118, 119, 120, 124, 130, 220, 245, 300, 305, 308, 324, 329, 367, 369, 386, 495, 558, 559, 570

COMMON — 308, 461, 570

COMPUTE — 363, 371, 380, 570

CONDITION — 496, 570

CONNECT — 271, 570

CONSISTENT — 488, 570

constraint — 132, 254, 270, 271, 305, 329, 330, 339, 382, 390, 402, 409, 526, 570

CONTINUE — 364, 570

CONTOH — 543, 570

CONTROL — 62, 151, 362, 570

COPIES — 545, 570

CPA0700 — 122, 570

CPA32B2 — 96, 570

CPA5305 — 48, 96, 100, 123, 570

CPC1129 — 220, 570

CPC1221 — 51, 570

CPC3701 — 123, 570

CPD0006 — 423, 570

CPD0032 — 123, 570

CPD5D03 — 123, 570

CPDBC01 — 124, 570

CPE3425 — 124, 570

CPE3426 — 124, 570

CPF0000 — 29, 31, 73, 77, 190, 223, 380, 425, 426, 427, 428, 483, 570

CPF0002 — 423, 570

CPF0670 — 121, 570

CPF0864 — 32, 570

CPF0907 — 48, 95, 121, 570

CPF0999 — 220, 570

CPF1002 — 33, 122, 429, 571

CPF1023 — 428, 571

CPF1107 — 122, 571

CPF1120 — 122, 571

CPF1164 — 246, 571

CPF1221 — 364, 571

CPF1240 — 48, 122, 571

CPF1241 — 122, 571

CPF1269 — 122, 571

CPF1338 — 122, 571

CPF1352 — 102, 122, 571

CPF1394 — 100, 122, 571

CPF1907 — 122, 571

CPF2105 — 31, 571

CPF2189 — 122, 571

CPF2204 — 122, 571

CPF2217 — 122, 571

CPF2451 — 123, 246, 571

CPF3700 — 100, 571

CPF3701 — 230, 571

CPF3730 — 123, 229, 230, 245, 246, 400, 406, 503, 571

CPF3768 — 230, 571

CPF3770 — 68, 123, 571

CPF3771 — 230, 571

CPF3782 — 122, 571

CPF3793 — 100, 123, 281, 571

CPF3800 — 100, 571

CPF3812 — 100, 123, 571

CPF4131 — 4, 83, 96, 100, 121, 122, 389, 542, 571

CPF4200 — 334, 571

CPF4218 — 117, 123, 571

CPF4257 — 100, 123, 334, 572

CPF4328 — 122, 330, 572

CPF5027 — 122, 557, 572

CPF5029 — 83, 122, 552, 554, 572

CPF5032 — 122, 572

CPF5116 — 122, 572

CPF5813 — 68, 572

CPF5D03 — 123, 572

CPF6700 — 334, 572

CPF7013 — 122, 572

CPF9801 — 34, 68, 122, 280, 331, 428, 572

CPF9810 — 122, 572

CPF9812 — 122, 572

CPF9815 — 122, 572

CPF9898 — 497, 572

CPF9899 — 124, 572

CPFA0A9 — 167, 572

CPI091D — 220, 572

CPI0948 — 48, 95, 121, 572

CPI0954 — 220, 572

CPI0999 — 246, 572

CPI099C — 121, 572

CPI3712 — 50, 572

CPI43 — 127, 376, 572

CPI4321 — 83, 127, 572

CPI4322 — 83, 93, 127, 376, 572

CPI4324 — 127, 572

CPI4326 — 127, 376, 572

CPI4327 — 83, 127, 376, 572

CPI4328 — 127, 572

CPI432F — 83, 127, 572

CPI4348 — 444, 445, 573

CPYFRMIMPF — 241, 300, 573

CPYFRMSTMF — 200, 236, 300, 573

CPYSPLF — 238, 302, 573

CPYTOIMPF — 240, 300, 573

CPYTOSTMF — 200, 236, 300, 573

CRGDTA — 99, 145, 146, 341, 344, 345, 482, 573

CRGEXIT — 482, 483, 573

CRGOWNER — 482, 573

CRGTYPE — 482, 573

CRTAUTL — 300, 573

CRTBNDCBL — 365, 573

CRTBNDCL — 201, 303, 573

CRTBNDDIR — 117, 303, 573

CRTBNDRPG — 18, 81, 90, 201, 303, 360, 573

CRTCAD — 484, 573

CRTCBLMOD — 365, 573

CRTCBLPGM — 365, 366, 573

CRTCLMOD — 303, 573

CRTCLS — 27, 28, 299, 474, 573

CRTCLU — 482, 573

CRTCMD — 421, 422, 573

CRTCRG — 482, 573

CRTDIR — 304, 573

CRTDSPF — 253, 303, 573

CRTDTAARA — 19, 34, 428, 454, 573

CRTDTAQ — 35, 272, 573

CRTJOBD — 215, 299, 475, 573

CRTJOBQ — 28, 299, 474, 573

CRTJRN — 49, 300, 375, 573

CRTJRNRCV — 49, 300, 375, 573

CRTLF — 253, 300, 574

CRTLIB — 18, 201, 260, 297, 298, 374, 574

CRTLINETH — 177, 179, 302, 574

CRTMSGF — 303, 574

CRTMSGQ — 303, 574

CRTOUTQ — 302, 574

CRTPF — 253, 300, 574

CRTPFRDTA — 55, 303, 463, 574

CRTPGM — 81, 303, 574

CRTPNLGRP — 424, 574

CRTPRTF — 253, 303, 574

CRTPRXCMD — 424, 574

CRTRPGMOD — 18, 81, 303, 574

CRTSAVF — 68, 272, 301, 334, 374, 574

CRTSBSD — 28, 197, 299, 474, 574

CRTSQLCBL — 365, 574

CRTSQLCBLI — 365, 574

CRTSQLRPGI — 117, 201, 303, 360, 574

CRTSRCPF — 300, 574

CRTSRVPGM — 18, 117, 303, 495, 574

CRTUSRPRF — 18, 46, 279, 300, 574

CTLGRP — 228, 574

CURCHAIN — 49, 60, 232, 245, 325, 375, 574

CURSORS — 131, 574

CUSTOM — 55, 212, 574

cutover — 6, 9, 16, 209, 210, 211, 440, 441, 574

CVNANZ — 206, 574

CVNSTS — 206, 574

CVNSTSTXT — 206, 574

CYCITV — 55, 574

CYCLE — 313, 574

CYCTIME — 55, 575

D

DAFTAR — 348, 575

daftar periksa — 140, 189, 206, 207, 221, 230, 233, 271, 328, 340, 342, 371, 399, 402, 403, 405, 419, 498, 499, 549, 554, 575

daftar risiko — 211, 508, 510, 511, 575

DAFTARCTL — 256, 575

DAFTARSFL — 256, 575

DAMPAK — 249, 457, 508, 575

dampak bisnis — 218, 277, 278, 470, 575

DARURAT — 347, 575

DARURAT01 — 334, 575

DASAR — 233, 575

data queue — 7, 34, 35, 86, 245, 272, 273, 274, 284, 305, 323, 387, 426, 432, 435, 436, 467, 575

DATABASE — 137, 341, 342, 362, 432, 575

date roll — 84, 98, 158, 318, 320, 459, 516, 549, 575

DATFMT — 530, 543, 575

DATSEP — 543, 575

DAYOFWEEK — 261, 575

DBGVIEW — 81, 90, 117, 201, 329, 360, 365, 371, 575

DBLQUOTE — 240, 241, 575

DBXREF — 222, 575

DEBET — 98, 147, 158, 159, 160, 312, 317, 320, 340, 342, 368, 380, 516, 550, 575

DEFERRED — 444, 575

DEGREE — 215, 243, 575

DELAY — 97, 99, 220, 222, 331, 341, 483, 575

DENDA — 157, 158, 575

DESCRIBED — 362, 363, 575

deskripsi job — 298, 575

deskripsi subsystem — 18, 28, 299, 474, 484, 575

DETACH — 442, 443, 575

DETACHED — 310, 576

DETAIL — 19, 32, 43, 121, 146, 341, 344, 365, 530, 547, 576

DEVPGM — 90, 201, 202, 576

DEVTYPE — 545, 576

DFTACTGRP — 201, 360, 365, 576

DFTCOLIVL — 55, 576

DFTCOLLVL — 463, 576

DFTROUTE — 42, 176, 576

DFTWAIT — 28, 474, 576

DIAGNOSIS — 457, 576

DIBATALKAN — 322, 576

DIBERI — 348, 576

DIBUKA — 278, 576

DIGITS — 67, 576

DIKENALI — 352, 576

DIKERJAKAN — 249, 576

DIKETAHUI — 249, 576

DIKIRIM — 287, 289, 576

DISABLED — 46, 101, 140, 262, 279, 576

DISETUJUI — 450, 576

DISKWATCH — 213, 576

DISPLAY — 82, 367, 380, 576

display file — 6, 10, 18, 253, 254, 257, 387, 529, 541, 542, 543, 544, 576

DITUTUP — 278, 576

DIUBAH — 134, 576

DIUJI — 348, 470, 576

DIURUTKAN — 354, 576

DIVISION — 361, 362, 364, 368, 576

DLCOBJ — 33, 35, 298, 313, 576

DLTCMNTRC — 185, 576

DLTJOBQ — 299, 576

DLTJRNRCV — 95, 577

DLTLIB — 298, 577

DLTMSGQ — 303, 577

DLTOBJ — 298, 577

DLTOPT — 95, 577

DLTOUTQ — 302, 577

DLTOVR — 258, 302, 543, 545, 577

DLTRCV — 49, 310, 577

DLTSPLF — 224, 238, 302, 519, 577

DLTUSRPRF — 300, 577

DLYJOB — 34, 35, 77, 167, 170, 219, 220, 222, 223, 245, 300, 577

DMNNAME — 42, 180, 577

DMNNODL — 484, 577

DMNSCHLIST — 180, 577

DOKUMEN — 233, 347, 445, 446, 577

DSCJOB — 41, 97, 186, 457, 577

DSPASPBAL — 214, 314, 577

DSPATR — 254, 255, 530, 532, 533, 535, 536, 537, 577

DSPAUTL — 300, 577

DSPAUTUSR — 300, 577

DSPBNDDIR — 123, 577

DSPCLUINF — 145, 482, 577

DSPCRGINF — 99, 145, 146, 341, 344, 577

DSPDTAARA — 19, 577

DSPFD — 18, 25, 200, 257, 259, 300, 329, 330, 577

DSPFFD — 18, 25, 257, 300, 577

DSPJOB — 235, 299, 577

DSPJOBLOG — 12, 75, 79, 96, 100, 104, 277, 299, 456, 577

DSPJRN — 18, 49, 60, 85, 232, 245, 300, 325, 375, 456, 577

DSPJRNRCVA — 95, 577

DSPLIB — 18, 36, 297, 298, 577

DSPLIBL — 22, 122, 282, 298, 327, 578

DSPLICKEY — 293, 578

DSPLIND — 178, 578

DSPLNK — 304, 578

DSPLOG — 51, 100, 303, 334, 578

DSPLOGBRM — 228, 229, 301, 578

DSPMOD — 303, 578

DSPMSG — 19, 303, 456, 578

DSPMSGD — 121, 303, 578

DSPOBJAUT — 43, 122, 123, 282, 298, 327, 456, 578

DSPOBJD — 19, 32, 43, 222, 298, 578

DSPOVR — 122, 259, 302, 327, 578

DSPPFM — 300, 578

DSPPFRCOL — 463, 578

DSPPGM — 303, 365, 578

DSPPGMREF — 100, 103, 120, 298, 324, 325, 370, 542, 578

DSPPTF — 39, 104, 277, 302, 578

DSPSAVF — 301, 578

DSPSBSD — 43, 122, 215, 299, 477, 578

DSPSFWRSC — 293, 302, 578

DSPSIZ — 254, 257, 530, 578

DSPSRVPGM — 18, 303, 578

DSPSYSSTS — 20, 578

DSPSYSVAL — 41, 139, 221, 235, 301, 334, 456, 578

DSPUSRPRF — 18, 43, 64, 104, 235, 300, 578

DTAARA — 19, 33, 34, 313, 326, 428, 429, 454, 578

DTAFMT — 236, 578

DTETIM — 464, 465, 578

dua orang — 76, 85, 97, 98, 248, 249, 283, 320, 385, 395, 396, 406, 423, 559, 560, 578

DUPLEX — 177, 578

DWCOL01 — 469, 578

DYNAMIC — 362, 579

E

EBCDIC — 234, 235, 236, 244, 386, 387, 579

EDTCDE — 254, 255, 256, 257, 258, 530, 536, 537, 543, 579

EDTLIBL — 299, 579

EDTRBDAP — 330, 579

EDTWRD — 255, 536, 579

EKSPOR — 236, 240, 579

EKSTERNAL — 263, 579

ELEMENT — 307, 579

EMAIL — 67, 579

ENABLE — 62, 151, 444, 579

ENABLED — 25, 107, 132, 141, 279, 297, 307, 330, 450, 579

ENCODED — 128, 579

ENDACTGRP — 432, 579

ENDASPBAL — 214, 302, 314, 579

ENDCMNTRC — 185, 302, 579

ENDCMTCTL — 119, 300, 579

ENDDBG — 82, 303, 456, 579

ENDDO — 31, 32, 33, 34, 68, 73, 77, 167, 170, 190, 223, 322, 358, 368, 423, 425, 427, 428, 429, 483, 579

ENDDW — 213, 303, 469, 579

ENDIF — 88, 119, 579

ENDING — 442, 579

ENDJOB — 97, 99, 122, 196, 262, 297, 299, 331, 377, 378, 557, 579

ENDJRNPF — 300, 579

ENDJWMON — 213, 303, 469, 579

ENDLINFMT — 200, 236, 579

ENDMOD — 360, 365, 579

ENDOPT — 223, 579

ENDPFRCOL — 303, 579

ENDPGMEXP — 117, 494, 580

ENDSBS — 27, 77, 220, 222, 299, 341, 424, 483, 580

ENDSELECT — 483, 580

ENDSRVJOB — 82, 303, 456, 580

ENDTCP — 220, 301, 580

ENDTCPIFC — 178, 301, 580

ENDTCPSVR — 137, 186, 187, 220, 301, 333, 341, 432, 483, 580

ENDWCH — 48, 303, 580

ENDWTR — 238, 302, 580

ENFORCED — 62, 151, 580

enkripsi — 5, 64, 86, 138, 139, 186, 187, 229, 239, 387, 580

ENTTYP — 49, 60, 232, 245, 325, 375, 580

ENVVAR — 485, 580

EODALL — 213, 580

EODBUNGA — 81, 82, 120, 325, 580

EODCLS — 28, 474, 475, 580

EODCTL — 422, 429, 580

EODGL — 366, 580

EODGRP — 80, 81, 580

EODJOB — 475, 580

EODJOBD — 84, 215, 475, 580

EODJOBQ — 28, 51, 84, 215, 335, 474, 475, 580

EODLOCK — 33, 429, 580

EODOPS — 73, 77, 322, 580

EODPOST — 30, 75, 79, 80, 82, 213, 245, 369, 497, 580

EODRPT — 238, 580

EODSBS — 28, 43, 76, 213, 220, 223, 245, 335, 341, 342, 345, 474, 475, 483, 512, 580

EODSTART — 518, 580

EODUSER — 75, 79, 80, 82, 213, 238, 245, 327, 366, 369, 497, 580

EODVLD — 422, 580

EODWATCH — 213, 580

ERRCOD — 425, 426, 581

ERRLOG — 31, 581

ERRLVL — 32, 241, 581

ERRMSG — 255, 536, 581

ERROR — 364, 495, 581

ESCAPE — 30, 31, 33, 34, 35, 68, 73, 77, 322, 368, 423, 425, 427, 429, 497, 581

ESKALASI — 352, 458, 581

estimasi — 41, 158, 211, 581

ETHAGG01 — 179, 581

ETHLINE — 42, 581

ETHLINE01 — 177, 178, 185, 581

ETHVLAN20 — 177, 581

EVALUASI — 288, 581

EVALUATE — 364, 581

EVENTF — 81, 90, 201, 360, 365, 581

EVFEVENT — 201, 581

EXCLUDE — 41, 107, 141, 260, 379, 422, 454, 457, 581

exit point — 17, 61, 86, 135, 301, 432, 581

EXITPGM — 482, 483, 581

EXITPNT — 61, 137, 187, 432, 581

EXPORT — 117, 494, 495, 581

EXTERNALLY — 362, 363, 581

F

FAILED — 162, 581

FAILOVER — 347, 483, 581

failover — 8, 59, 77, 103, 145, 147, 148, 179, 185, 246, 283, 344, 345, 347, 387, 391, 405, 483, 488, 581, 619

fase posting — 245, 318, 319, 323, 333, 360, 392, 467, 470, 513, 557, 581

FATAL — 68, 77, 581

faulting — 47, 54, 57, 99, 214, 304, 334, 335, 456, 464, 476, 513, 581

FEATURE — 293, 581

FIELDPROC — 139, 387, 582

file fisik — 18, 582

FILTER — 349, 582

FIXED — 475, 476, 582

FLDDLM — 240, 241, 582

FMTBCD — 185, 582

FMTTCP — 185, 582

FOLDER — 239, 582

FOOTER — 530, 538, 582

FORCE — 272, 273, 387, 582

FORENSIK — 260, 261, 263, 582

FORMAT — 61, 137, 187, 425, 432, 582

FORMATS — 362, 363, 582

FORMTYPE — 545, 582

FPNIK — 139, 582

FREKUENSI — 500, 504, 582

FREQUENCY — 144, 243, 480, 518, 582

FROMENT — 232, 582

FROMFILE — 240, 582

FROMMBR — 200, 236, 582

FROMMGTCOL — 55, 463, 582

FROMRCD — 241, 582

FROMSTMF — 200, 236, 241, 582

FROMTIME — 49, 232, 325, 582

FTPEXIT — 187, 582

FTPLOGON — 187, 582

G

GABUNG — 30, 582

GAGAL — 29, 73, 74, 141, 142, 164, 167, 200, 201, 202, 223, 229, 321, 322, 356, 582

gerbang keputusan — 440, 582

GETSIZE — 167, 583

GLBALANCE — 421, 583

GLSALDO — 516, 583

GLTRX — 98, 118, 147, 154, 158, 159, 160, 312, 317, 320, 340, 342, 516, 550, 552, 583

GLTRXR — 118, 583

GOBACK — 362, 363, 371, 388, 583

GORESTRICT — 222, 583

GRAPHIC — 237, 583

group profile — 46, 64, 282, 450, 452, 583

GRPCS — 151, 583

GRPEODOPS — 422, 583

GRPOPS — 62, 583

GRPPRF — 46, 583

GRPPUSAT — 62, 151, 583

GRPTELLER — 46, 583

GRTOBJAUT — 298, 422, 454, 583

GRTUSRAUT — 300, 583

GTOTAL — 258, 583

H

hak minimum — 454, 583

HALAMAN — 543, 583

hari kerja — 69, 207, 222, 394, 398, 411, 418, 419, 441, 455, 479, 480, 583

hari libur — 69, 479, 480, 583

HARIAN — 228, 258, 427, 504, 545, 583

HASIL — 32, 138, 218, 322, 362, 363, 364, 368, 373, 428, 429, 483, 501, 504, 583

HEADER — 530, 583

HIPOTESIS — 470, 583

hipotesis — 218, 353, 382, 394, 406, 470, 552, 553, 554, 583

HISTORY — 133, 583

HITBUNGA — 81, 82, 84, 120, 200, 201, 231, 325, 326, 327, 360, 361, 365, 369, 497, 583

HITUNG — 29, 31, 362, 363, 365, 495, 584

HITUNGH — 491, 492, 584

HLDJOB — 299, 584

HLDJOBQ — 27, 299, 513, 584

HLDJOBSCDE — 300, 584

HLDSPLF — 238, 302, 584

HLPID — 424, 584

HLPPNLGRP — 424, 584

HOSTNAME — 42, 180, 584

HOSTSCHPTY — 180, 584

HOURS — 101, 107, 125, 151, 220, 230, 246, 279, 336, 337, 584

HTTPS — 184, 584

HTTPSVR — 42, 186, 187, 333, 342, 483, 584

hypercare — 209, 441, 584

I

IASP01 — 145, 146, 147, 342, 345, 482, 584

IDCIF — 530, 584

idempotensi — 154, 168, 192, 194, 195, 269, 385, 410, 518, 584

IDENTITAS — 233, 347, 584

IGNINQMSG — 78, 95, 232, 584

IMAGES — 49, 271, 329, 375, 584

IMMED — 77, 97, 238, 262, 557, 584

IMMEDIATE — 444, 584

IMPOR — 236, 584

INACTTIMO — 187, 584

INCLUDE — 128, 284, 364, 387, 584

INCOSE — 460, 584

INCREMENT — 313, 584

INDARA — 533, 584

index rusak — 330, 339, 526

INDEXED — 362, 584

indikator — 83, 117, 119, 120, 253, 256, 325, 327, 334, 358, 495, 531, 533, 534, 537, 538, 544, 547, 585

INDTXT — 544, 585

INETD — 482, 585

INITIALLY — 444, 585

INLJOBS — 215, 216, 477, 478, 585

INLLIBL — 215, 475, 585

INLMNU — 46, 279, 585

INLPGM — 46, 585

INOREK — 254, 585

INPUT — 362, 585

INQUIRY — 48, 96, 585

INSIDEN — 249, 263, 585

INSPTF — 302, 585

INSTALLED — 25, 109, 293, 297, 585

INTEGRITAS — 263, 585

integritas data — 59, 149, 210, 308, 407, 585

INTERACT — 84, 257, 321, 341, 422, 519, 585

INTERAKTIF — 352, 585

INTERVAL — 55, 212, 213, 463, 585

INTJOB — 425, 585

INTNETADR — 42, 178, 179, 180, 181, 585

INTSBS — 474, 585

inventarisasi — 93, 207, 245, 257, 371, 381, 384, 510, 542, 585

IOSYSCFG — 45, 585

IPLSRC — 220, 585

IREXX — 422, 585

ISACA — 460, 585

J

jalur akses — 5, 61, 103, 126, 253, 254, 330, 389, 390, 432, 446, 466, 512, 585

jam layanan — 4, 36, 76, 99, 100, 151, 169, 170, 215, 283, 287, 289, 419, 449, 458, 512, 513, 542, 585

JANGAN — 97, 98, 271, 516, 586

JARINGAN — 288, 347, 586

JBCPU — 464, 586

JBDBR — 464, 586

JBDBW — 464, 586

JBNAME — 464, 586

JBNBR — 464, 586

JBNDBR — 464, 586

JBNDBW — 464, 586

JBUSER — 464, 586

JEJAK — 268, 269, 326, 440, 586

jejak audit — 45, 64, 157, 291, 309, 333, 336, 337, 449, 452, 586

jendela pemeliharaan — 270, 334, 409, 416, 586

JMLJOB — 421, 425, 586

job description — 144, 215, 299, 388, 475, 484, 559, 586

job queue — 17, 27, 28, 43, 47, 67, 105, 215, 282, 298, 299, 304, 335, 372, 388, 391, 456, 458, 474, 476, 513, 518, 586

job scheduler — 51, 57, 59, 144, 221, 342, 485, 518, 528, 586

JOBACT — 464, 586

JOBCTL — 45, 586

JOBI0200 — 425, 586

JOBLOG — 225, 586

JOBMI — 462, 586

JOBNAME — 73, 428, 586

JOBNBR — 428, 586

JOBOS — 462, 586

JOBPTY — 215, 281, 282, 475, 586

JOBQ01 — 12, 586

JOBUSER — 428, 586

JOENTT — 50, 232, 325, 375, 586

JOESD — 50, 586

JOJOB — 50, 232, 325, 375, 587

JOPGM — 50, 232, 325, 326, 375, 587

JOTSTP — 50, 232, 325, 375, 587

journal receiver — 38, 47, 57, 77, 85, 95, 96, 121, 141, 224, 225, 230, 260, 261, 283, 310, 388, 458, 467, 468, 469, 587

JOUSER — 50, 232, 325, 375, 587

JRNCARI — 232, 325, 587

JRNDTA — 49, 78, 98, 109, 232, 245, 271, 325, 329, 344, 587

JRNLIB — 49, 77, 78, 95, 98, 232, 245, 271, 325, 329, 344, 587

JRNOUT — 49, 50, 375, 587

JRNRCV — 18, 49, 77, 95, 260, 337, 375, 587

JRNREK — 245, 587

JRNUJI — 375, 587

JUDUL — 258, 543, 587

JUMLAH — 32, 133, 134, 208, 244, 340, 342, 357, 587

JWCOL01 — 469, 587

K

KALENDER — 480, 587

kalender — 69, 226, 394, 396, 406, 414, 480, 554, 587

KANAL — 348, 587

KARANTINA — 92, 587

KARTU — 139, 141, 261, 264, 587

KAS001 — 368, 587

KASUS — 352, 353, 587

kata sandi — 9, 233, 347, 422, 434, 452, 453, 454, 536, 537, 587

KATEGORI — 208, 286, 287, 288, 292, 508, 587

KDCAB — 530, 587

KDFASE — 73, 587

KDPRD — 267, 269, 439, 440, 587

KEAMANAN — 263, 587

kebijakan retensi — 51, 102, 103, 224, 239, 519, 587

KELUAR — 351, 588

KELUARAN — 348, 350, 351, 352, 354, 355, 356, 357, 588

KEMBALIKAN — 349, 351, 352, 353, 354, 355, 356, 357, 588

KENDALI — 500, 504, 508, 588

kendali akses — 61, 64, 135, 137, 588

kendali atas kendali — 504, 588

kepatuhan — 3, 64, 91, 141, 142, 264, 279, 280, 285, 332, 379, 400, 401, 414, 423, 437, 446, 498, 526, 588

KEPUTUSAN — 250, 588

KERJA — 31, 588

KESALAHAN — 80, 362, 363, 436, 588

kesinambungan usaha — 404, 487, 588

KETERANGAN — 110, 134, 190, 287, 289, 317, 326, 346, 439, 442, 450, 480, 549, 588

ketersediaan tinggi — 10, 390, 481, 588

KEUANGAN — 355, 588

KEWAJIBAN — 348, 588

KEYED — 272, 588

KEYLEN — 272, 588

KINERJA — 218, 470, 588

KIRIMALERT — 110, 588

KLIST — 358, 588

kode transaksi — 550, 551, 588

KODEHASIL — 368, 588

KOLEKTOR — 291, 588

KONTAK — 233, 347, 588

KONTEKS — 250, 498, 588

kontensi — 99, 122, 151, 152, 212, 282, 289, 352, 360, 376, 390, 466, 467, 588

KONTROL — 268, 269, 348, 588

KOSONG — 354, 355, 356, 588

KRED001 — 454, 588

KREDIT — 98, 154, 157, 158, 159, 160, 164, 312, 316, 317, 320, 325, 340, 342, 368, 380, 516, 550, 588

KRITIS — 277, 291, 355, 433, 483, 521, 588

KRONOLOGI — 263, 589

KUERI — 500, 589

kunci objek — 465, 589

kunci record — 119, 151, 152, 212, 257, 376, 390, 465, 467, 468, 557, 589

KURSOR — 364, 589

L

lag replikasi — 98, 146, 341, 344, 402, 487, 489, 503, 549, 589

LAKUKAN — 97, 98, 349, 350, 354, 355, 356, 516, 589

LAMPIRAN — 264, 589

LANGUAGE — 134, 589

LANJUT — 218, 589

LAPKREDIT — 332, 589

LAPOR — 332, 362, 363, 364, 589

LAPORAN — 238, 239, 258, 263, 421, 470, 543, 589

laporan regulator — 5, 163, 225, 238, 239, 241, 332, 339, 459, 519, 526, 589

LAPSALDO — 258, 259, 545, 589

LASTCHG — 484, 589

LASTSAVE — 78, 98, 232, 375, 589

latensi — 275, 437, 438, 486, 589

LATERAL — 100, 589

LATIHAN — 374, 375, 377, 589

LAYARCIF — 534, 540, 542, 543, 589

LAYARUTAMA — 254, 589

LCKLVL — 119, 589

LCLINTNETA — 182, 589

LCLRSP — 463, 589

LEAVE — 223, 589

LEMPAR — 349, 350, 351, 589

LENGTH — 236, 237, 241, 257, 266, 434, 438, 589

level check — 4, 83, 96, 100, 122, 339, 389, 525, 541, 542, 543, 590

LIBRARY — 479, 590

library list — 22, 83, 96, 122, 123, 124, 215, 282, 298, 299, 327, 388, 389, 424, 553, 590

LICPGM — 39, 277, 590

LINESPEED — 177, 590

LINKAGE — 362, 368, 371, 590

LISTEN — 108, 140, 184, 191, 590

LMTCPB — 46, 64, 279, 284, 590

LNKAGG — 179, 590

LOCAL — 89, 180, 198, 590

LOCATION — 229, 590

LODPTF — 302, 590

LOGCLPGM — 79, 215, 475, 590

LOKUP — 358, 590

LOWER — 30, 590

LPRINTF — 306, 307, 590

LULUS — 141, 142, 164, 167, 321, 322, 356, 590

LVLCHK — 543, 590

M

MAINTAINED — 444, 590

manajemen perubahan — 64, 382, 396, 551, 552, 590

manajemen risiko — 4, 10, 64, 93, 294, 371, 489, 508, 590

manajemen vendor — 7, 277, 590

MASALAH — 218, 263, 470, 590

MASIH — 349, 590

MASUKAN — 348, 350, 351, 352, 354, 355, 356, 357, 590

MATCHED — 134, 590

MAXACT — 27, 28, 212, 214, 335, 475, 590

MAXDTA — 469, 590

MAXFRAME — 177, 590

MAXHOP — 182, 591

MAXJOBS — 477, 478, 591

MAXLEN — 35, 272, 591

MAXOPT3 — 49, 591

MAXPOS — 422, 591

MAXRCDS — 213, 591

MAXSTG — 185, 591

MAXUSE — 215, 216, 477, 478, 591

MBRLIST — 200, 591

MBROPT — 200, 231, 236, 240, 241, 591

MCH0000 — 31, 73, 591

memory pool — 17, 27, 28, 43, 197, 214, 372, 386, 389, 392, 462, 464, 591

MENGANDUNG — 352, 591

MENURUT — 354, 591

MENUTELLER — 46, 591

MERGE — 134, 591

message file — 303, 591

METODE — 470, 591

METRIK — 110, 190, 286, 287, 288, 289, 290, 291, 292, 294, 295, 591

MGTCOL — 463, 591

MIGGRP — 268, 591

MIGRASI — 268, 269, 270, 439, 440, 591

migrasi data — 6, 9, 16, 129, 265, 313, 438, 448, 591

MINGGUAN — 228, 504, 591

MINUTES — 161, 276, 288, 289, 290, 291, 292, 321, 591

MITIGASI — 508, 591

MITRA — 348, 591

MIXED — 422, 591

MKDIR — 304, 591

MNGRCV — 49, 591

MODULE — 18, 81, 82, 117, 201, 365, 495, 591

MONITOR — 117, 213, 286, 287, 288, 289, 290, 291, 292, 294, 295, 495, 592

MONMSG — 29, 31, 32, 33, 34, 35, 68, 73, 77, 167, 190, 223, 303, 373, 380, 425, 426, 427, 428, 429, 483, 592

MONOBJ — 144, 592

MONTHS — 170, 243, 291, 294, 295, 592

MOVEL — 358, 592

MOVOBJ — 92, 297, 298, 592

MSGDTA — 30, 31, 73, 423, 592

MSGFLIB — 31, 592

MSGID — 29, 30, 31, 32, 33, 34, 48, 51, 68, 73, 77, 100, 167, 190, 223, 334, 380, 423, 425, 426, 427, 428, 429, 483, 592

MSGKEY — 426, 592

MSGMODE — 181, 592

MSGTXT — 31, 592

MSGTYPE — 30, 31, 33, 34, 68, 73, 77, 223, 322, 368, 373, 423, 425, 427, 428, 429, 592

MULAI — 133, 222, 268, 348, 350, 351, 352, 354, 355, 356, 357, 450, 479, 592

multi-factor — 61, 592

MUTASI — 515, 592

N

NAMAKUNCI — 428, 429, 592

NAMATABEL — 329, 592

NAMES — 362, 592

NAMING — 32, 201, 592

NASABAH — 67, 139, 141, 530, 592

NBRPKT — 181, 190, 592

NETSTAT — 173, 176, 178, 179, 182, 302, 592

NEWOBJ — 271, 592

NEWOWN — 454, 592

NEXTHOP — 42, 176, 592

NILAI — 34, 110, 190, 286, 287, 288, 289, 290, 291, 292, 294, 295, 340, 342, 592

NILAI1 — 326, 592

NILAI2 — 326, 592

NODEDR — 482, 484, 593

NODEPRIMER — 345, 593

NODEPROD — 482, 484, 593

NOMAX — 213, 245, 422, 593

NOMINAL — 161, 162, 164, 167, 241, 311, 318, 333, 340, 342, 346, 380, 442, 443, 550, 593

NOMOR — 33, 313, 593

nomor relatif — 541, 593

NONSYS — 50, 222, 223, 593

NOPWD — 107, 138, 333, 433, 593

NOREF — 368, 593

NORMAL — 277, 291, 337, 355, 421, 593

NOURUT — 313, 593

NSLOOKUP — 179, 593

NVARCHAR — 237, 593

O

OBJAUD — 41, 141, 151, 337, 457, 593

OBJAUT — 43, 593

OBJCREATED — 92, 105, 225, 262, 329, 424, 593

OBJDEFINER — 485, 593

OBJEK — 208, 593

OBJLIB — 206, 593

OBJLIST — 19, 32, 43, 270, 593

OBJMGT — 141, 262, 279, 593

OBJNAME — 92, 105, 109, 142, 206, 225, 246, 262, 329, 424, 485, 504, 520, 555, 556, 593

OBJOWNER — 92, 105, 262, 485, 520, 593

OBJSIZE — 92, 95, 105, 109, 310, 311, 329, 372, 593

OBJTEXT — 105, 262, 424, 520, 593

ODBCCLS — 477, 593

ODBCEXIT — 61, 136, 137, 593

ODCDAT — 19, 593

ODLUSD — 19, 594

ODOBNM — 19, 32, 594

ODOBSZ — 19, 32, 594

ODOBTP — 19, 594

OMTJRNE — 49, 329, 594

ONAMA — 254, 594

ONLINE — 482, 594

open source — 5, 17, 89, 196, 197, 199, 252, 389, 594

OPESAN — 255, 594

OPMSRC — 82, 594

OPNCLO — 49, 329, 594

OPTIMIZE — 129, 215, 594

OSALDO — 254, 594

OSSSBS — 197, 594

OTHER — 364, 594

OTOMATIS — 500, 594

OUTFILE — 19, 32, 43, 49, 60, 100, 120, 200, 206, 232, 245, 263, 325, 375, 542, 594

OUTFILFMT — 49, 60, 232, 375, 594

OUTMBR — 206, 594

OUTPTY — 215, 475, 594

OUTPUT — 19, 20, 32, 39, 43, 49, 60, 79, 100, 120, 200, 232, 245, 277, 293, 325, 330, 362, 375, 456, 542, 594

output queue — 144, 238, 298, 302, 305, 456, 519, 528, 594

OVERLAY — 255, 256, 258, 536, 537, 544, 594

OVRDSPF — 543, 594

OVRFLW — 258, 594

OVRPRTF — 258, 302, 545, 594

OVRSCOPE — 543, 545, 594

OWNER — 46, 141, 263, 422, 454, 594

P

PACKED — 362, 363, 364, 367, 368, 594

PAGESIZE — 258, 595

PAGING — 475, 476, 595

paging — 475, 476, 595

PAGNBR — 258, 595

PANJANG — 30, 35, 165, 427, 595

PARAM1 — 34, 595

PARAMETER — 479, 595

PARM1 — 29, 595

PARM2 — 29, 595

PASSWORD — 46, 279, 595

PEGAWAI — 62, 595

PELAPORAN — 263, 348, 595

PEMBARUAN — 249, 595

pemeriksaan pagi — 69, 99, 226, 281, 419, 420, 516, 595

PEMETAAN — 267, 595

PEMICU — 457, 595

PEMILIK — 450, 454, 500, 504, 508, 595

pemisahan tugas — 64, 67, 85, 203, 223, 284, 503, 595

pemuatan bertahap — 257, 539, 595

PEMULIHAN — 233, 263, 347, 595

pemulihan bencana — 8, 64, 233, 339, 384, 402, 406, 416, 595

pemulihan ke titik waktu — 49, 96, 374, 388, 446, 595

PENAHANAN — 263, 595

penanda selesai — 435, 514, 515, 595

penanganan kesalahan — 10, 117, 495, 595

PENANGGUNG — 507, 595

pencatatan — 94, 132, 137, 188, 189, 194, 195, 196, 215, 278, 279, 388, 424, 428, 430, 432, 490, 498, 548, 549, 595

PENDING — 161, 595

PENERAPAN — 218, 595

pengerasan sistem — 5, 140, 595

PENGGUNA — 132, 596

pengujian bergilir — 560, 596

PENGUKURAN — 218, 596

penjadwalan — 3, 10, 51, 299, 479, 596

PENUGASAN — 450, 455, 596

PENYEBAB — 554, 596

penyimpanan sementara — 304, 310, 330, 335, 376, 391, 448, 478, 512, 513, 527, 539, 596

PERAN — 450, 483, 596

PERBAIKAN — 264, 457, 596

perencanaan kapasitas — 56, 291, 292, 596

PERFDTA — 463, 464, 465, 469, 596

PERFORM — 362, 363, 364, 365, 596

PERHATIAN — 198, 236, 277, 291, 355, 596

PERIKSA — 267, 596

PERINGATAN — 96, 97, 99, 101, 103, 287, 288, 289, 329, 330, 333, 334, 335, 336, 457, 513, 519, 521, 596

PERINTAH — 427, 596

PERIOD — 51, 100, 133, 228, 229, 334, 596

PERIODE — 163, 164, 596

PERKIRAAN — 249, 596

PERMISSION — 62, 151, 189, 596

PERNYATAAN — 470, 596

pertahanan berlapis — 60, 596

PESAN — 29, 30, 68, 147, 167, 194, 195, 255, 287, 288, 289, 428, 479, 498, 514, 530, 596

pesan diagnostik — 426, 596

pesan escape — 122, 425, 498, 596

PFILE — 254, 596

PFRDATA — 55, 213, 596

PGMBDY — 497, 596

PGMFAIL — 60, 596

PGMFASE — 73, 596

PGMLAMA — 92, 597

PGMLVL — 117, 494, 597

PGMNBR — 61, 137, 187, 432, 597

PGMREF — 100, 120, 245, 259, 325, 480, 542, 597

PGMSTK — 75, 245, 284, 597

PILIHAN — 250, 597

PINDAH — 272, 597

PKTLEN — 181, 242, 283, 379, 398, 597

plan cache — 126, 127, 306, 390, 448, 597

PLANCACHE — 127, 597

PMTCTL — 422, 597

PNAMA — 543, 597

PNLGRP — 424, 597

PNOREK — 543, 597

POACTLVL — 464, 597

PODBFLT — 464, 597

POINT — 362, 597

POKOK — 380, 597

POLLITV — 334, 597

PONBR — 464, 597

PONDBFLT — 464, 597

pool memori — 57, 170, 391, 449, 472, 475, 597

POOLID — 28, 475, 597

POOLS — 28, 197, 474, 597

POSISI — 30, 165, 290, 597

POSITION — 22, 240, 307, 597

POSIZE — 464, 597

POSTEOD — 228, 597

POSTING — 365, 380, 421, 494, 495, 597

POSTINGGL — 365, 366, 368, 369, 380, 597

POSTMASUK — 552, 597

PRASYARAT — 457, 598

PRBLOG — 225, 598

PRDIBMI — 30, 42, 68, 180, 423, 598

PRDID — 293, 598

PREEOD — 228, 598

PREFIFC — 179, 598

prestart job — 10, 215, 216, 299, 390, 477, 478, 598

PRINT — 20, 39, 43, 79, 120, 277, 293, 325, 330, 456, 530, 598

printer file — 6, 10, 18, 253, 258, 259, 302, 303, 387, 543, 544, 545, 598

prioritas job — 282, 390, 513, 598

PROCESS — 362, 598

PRODUK — 153, 267, 322, 365, 497, 598

PROFIL — 208, 262, 598

profil pengguna — 4, 18, 25, 100, 108, 235, 301, 306, 426, 434, 449, 456, 598

PROGRAM — 22, 125, 326, 361, 428, 479, 498, 598

PROMPT — 421, 422, 598

PROSES — 31, 362, 363, 364, 365, 380, 598

PROSESBLOK — 170, 598

PROSESEOD — 33, 380, 421, 598

PROTOCOL — 42, 108, 140, 184, 191, 598

PRT01 — 238, 519, 598

PRTCMNTRC — 185, 298, 302, 598

PRTPUBAUT — 301, 598

PRTPVTAUT — 301, 598

PRTUSRPRF — 301, 598

PSALDO — 543, 598

PTFID — 277, 598

PTOTAL — 543, 598

PUBLIC — 45, 107, 141, 260, 282, 379, 422, 454, 598

PULIH — 249, 598

PUSTAKA — 208, 598

pustaka utilitas — 9, 421, 427, 599

PUTOVR — 536, 599

PWDEXPITV — 279, 599

PWRDWNSYS — 219, 220, 302, 599

Q

QACTJOB — 102, 599

QADLACTJ — 102, 599

QALWOBJRST — 41, 42, 372, 457, 599

QAPMDISK — 464, 599

QAPMETH — 464, 599

QAPMJOBMI — 464, 599

QAPMJOBOS — 464, 599

QAPMJOBWT — 464, 599

QAPMPOOL — 464, 599

QAPMSYSTEM — 464, 465, 599

QAQQINI — 129, 310, 331, 390, 448, 513, 599

QAUDCTL — 41, 42, 141, 336, 337, 372, 457, 599

QAUDENDACN — 337, 599

QAUDJRN — 60, 63, 64, 101, 107, 108, 151, 251, 261, 264, 279, 290, 332, 336, 337, 390, 455, 520, 599

QAUDLVL — 41, 42, 60, 337, 457, 599

QAUDLVL2 — 337, 599

QBATCH — 28, 220, 472, 473, 599

QCAPCMD — 426, 599

QCCSID — 235, 599

QCLLESRC — 200, 599

QCLSRC — 221, 599

QCMDEXC — 240, 306, 307, 426, 427, 599, 603

QCPFMSG — 121, 423, 497, 599

QCRTAUT — 41, 42, 372, 457, 599

QDATE — 33, 43, 69, 77, 599

QDDSSRC — 200, 257, 600

QDSPSGNINF — 457, 600

QDWDFT — 469, 600

QEZSNDMG — 426, 600

QHTTPSVR — 28, 189, 600

QINACTITV — 41, 97, 186, 457, 519, 600

QINACTMSGQ — 41, 97, 186, 457, 600

QINTER — 28, 77, 97, 220, 246, 341, 342, 473, 600

QJWDFT — 469, 600

QLMTSECOFR — 41, 42, 457, 600

QMAXJOB — 102, 600

QMAXSGNACN — 41, 453, 457, 600

QMAXSIGN — 41, 42, 372, 453, 457, 600

QMHRCVPM — 426, 600

QMHRMVPM — 426, 600

QMHSNDPM — 426, 497, 600

QMPGDATA — 55, 212, 463, 600

QPADEV0001 — 262, 600

QPADEV0003 — 552, 600

QPADEV0021 — 557, 600

QPFRADJ — 214, 600

QPWDEXPITV — 453, 457, 600

QPWDLVL — 41, 42, 372, 453, 457, 600

QPWDMAXLEN — 453, 600

QPWDMINLEN — 453, 600

QPWDRQDDIF — 41, 453, 457, 600

QPWDRULES — 453, 600

QQPARM — 129, 448, 600

QQRID — 127, 600

QQRYDEGREE — 215, 600

QQTEXT — 129, 448, 600

QQVAL — 129, 448, 601

QRCVDTAQ — 35, 245, 272, 426, 601

QRMTSIGN — 457, 601

QRPGLEREF — 115, 202, 491, 492, 601

QRPGLESRC — 89, 199, 200, 601

QRPLOBJ — 224, 601

QRYFILE — 25, 601

QSECOFR — 141, 601

QSECURITY — 41, 42, 142, 372, 390, 457, 484, 601

QSERVER — 28, 474, 601

QSNDDTAQ — 35, 272, 426, 601

QSQJRN — 558, 601

QSQLSRC — 200, 601

QSRVSRC — 117, 123, 494, 601

QSSLCSL — 139, 433, 601

QSSLCSLCTL — 139, 601

QSSLPCL — 139, 433, 601

QSTRUP — 101, 221, 601

QSTRUPPGM — 221, 601

QSYRUSRI — 426, 601

QSYS2.ACTIVE_QUERY_INFO — 106, 305, 331, 601

QSYS2.ASP_INFO — 304, 601

QSYS2.AUTHORIZATION_LIST_INFO — 306, 601

QSYS2.AUTHORIZATION_LIST_USER_INFO — 306, 601

QSYS2.BOUND_MODULE_INFO — 100, 120, 325, 601

QSYS2.BOUND_SRVPGM_INFO — 120, 325, 494, 495, 601

QSYS2.CERTIFICATE_INFO — 107, 138, 306, 333, 433, 601

QSYS2.CLUSTER_ADMIN_DOMAIN_MRE_INFO — 488, 601

QSYS2.CLUSTER_INFO — 145, 297, 482, 488, 601

QSYS2.CLUSTER_RESOURCE_GROUP_INFO — 145, 341, 488, 602

QSYS2.DATA_AREA_INFO — 305, 602

QSYS2.DATA_QUEUE_ENTRIES — 273, 305, 602

QSYS2.DATA_QUEUE_INFO — 273, 305, 437, 602

QSYS2.DB_TRANSACTION_INFO — 119, 131, 220, 305, 322, 341, 369, 602

QSYS2.DISPLAY_FILE_INFO — 257, 602

QSYS2.DISPLAY_JOURNAL — 64, 101, 107, 108, 151, 251, 261, 264, 279, 290, 305, 332, 336, 337, 455, 520, 558, 602

QSYS2.DUMP_PLAN_CACHE — 127, 306, 602

QSYS2.ENSURE_STATISTICS — 129, 306, 448, 602

QSYS2.ENVIRONMENT_VARIABLE_INFO — 306, 602

QSYS2.EXIT_PROGRAM_INFO — 107, 306, 602

QSYS2.FUNCTION_INFO — 306, 602

QSYS2.FUNCTION_USAGE — 107, 306, 602

QSYS2.GENERATE_SQL — 270, 306, 602

QSYS2.GENERATE_SQL_OBJECTS — 270, 306, 602

QSYS2.GROUP_PROFILE_ENTRIES — 306, 602

QSYS2.GROUP_PTF_INFO — 25, 40, 109, 304, 602

QSYS2.HISTORY_LOG_INFO — 125, 220, 230, 245, 246, 305, 602

QSYS2.IFS_OBJECT_STATISTICS — 106, 226, 262, 305, 332, 446, 514, 602

QSYS2.IFS_READ — 201, 305, 307, 602

QSYS2.IFS_WRITE — 305, 602

QSYS2.JOB_INFO — 105, 304, 498, 602

QSYS2.JOB_QUEUE_INFO — 105, 304, 476, 602

QSYS2.JOBLOG_INFO — 80, 97, 104, 127, 224, 304, 373, 376, 552, 602

QSYS2.JOURNAL_INFO — 109, 305, 602

QSYS2.JOURNAL_RECEIVER_INFO — 95, 98, 305, 310, 602

QSYS2.JOURNALED_OBJECTS — 109, 142, 305, 329, 504, 602

QSYS2.LIBRARY_INFO — 305, 602

QSYS2.LICENSE_INFO — 293, 304, 602

QSYS2.MEMORY_POOL — 104, 214, 304, 335, 476, 513, 602

QSYS2.MEMORY_POOL_INFO — 304, 603

QSYS2.MESSAGE_FILE_DATA — 121, 305, 603

QSYS2.MESSAGE_QUEUE_INFO — 48, 305, 603

QSYS2.NETSTAT_INFO — 108, 140, 183, 184, 190, 191, 288, 306, 603

QSYS2.NETSTAT_INTERFACE_INFO — 108, 178, 190, 191, 306, 603

QSYS2.NETSTAT_JOB_INFO — 42, 108, 183, 306, 314, 603

QSYS2.NETSTAT_ROUTE_INFO — 176, 191, 306, 603

QSYS2.OBJECT_LOCK_INFO — 106, 305, 603

QSYS2.OBJECT_PRIVILEGES — 107, 305, 603

QSYS2.OBJECT_STATISTICS — 92, 95, 105, 109, 142, 208, 225, 246, 262, 297, 305, 310, 311, 329, 372, 424, 485, 504, 520, 555, 556, 603

QSYS2.OUTPUT_QUEUE_ENTRIES — 51, 225, 239, 305, 519, 545, 603

QSYS2.OUTPUT_QUEUE_INFO — 305, 603

QSYS2.OVERRIDE_QAQQINI — 127, 306, 376, 444, 603

QSYS2.PROGRAM_INFO — 100, 141, 204, 263, 325, 327, 331, 366, 369, 371, 553, 603

QSYS2.PTF_INFO — 40, 104, 109, 220, 304, 603

QSYS2.QCMDEXC — 240, 306, 307, 603

QSYS2.RECORD_LOCK_INFO — 76, 96, 104, 106, 152, 282, 305, 377, 512, 557, 603

QSYS2.SCHEDULED_JOB_INFO — 144, 243, 304, 480, 518, 603

QSYS2.SECURITY_INFO — 306, 603

QSYS2.SERVER_SHARE_INFO — 42, 108, 186, 189, 306, 603

QSYS2.SERVICE_PROGRAM_INFO — 495, 603

QSYS2.SOFTWARE_PRODUCT_INFO — 109, 293, 297, 304, 603

QSYS2.SUBSYSTEM_INFO — 105, 304, 341, 476, 603

QSYS2.SUBSYSTEM_POOL_INFO — 304, 603

QSYS2.SYSCOLUMNS — 236, 244, 305, 516, 603

QSYS2.SYSCOLUMNSTAT — 129, 305, 447, 603

QSYS2.SYSCST — 132, 305, 330, 603

QSYS2.SYSDISKSTAT — 21, 35, 95, 105, 214, 297, 304, 314, 469, 603

QSYS2.SYSINDEXES — 305, 603

QSYS2.SYSIXADV — 55, 106, 305, 603

QSYS2.SYSPARTITIONINDEXSTAT — 103, 106, 129, 305, 604

QSYS2.SYSPARTITIONSTAT — 443, 604

QSYS2.SYSROUTINES — 305, 604

QSYS2.SYSTABLES — 305, 444, 604

QSYS2.SYSTABLESTAT — 55, 85, 102, 106, 305, 307, 517, 604

QSYS2.SYSTEM_STATUS — 104, 110, 197, 287, 304, 322, 372, 513, 604

QSYS2.SYSTEM_STATUS_INFO — 104, 220, 304, 604

QSYS2.SYSTEM_VALUE_INFO — 42, 102, 142, 144, 304, 337, 372, 604

QSYS2.SYSTRIGGERS — 132, 305, 604

QSYS2.SYSVIEWS — 305, 604

QSYS2.TCPIP_HOST_TABLE — 180, 306, 604

QSYS2.TCPIP_INFO — 180, 306, 604

QSYS2.USER_INFO — 25, 46, 85, 101, 104, 107, 141, 144, 208, 261, 297, 306, 307, 336, 450, 604

QSYS2.USER_INFO_BASIC — 306, 604

QSYSOPR — 48, 95, 221, 342, 456, 458, 604

QSYSWRK — 28, 604

QTIME — 33, 170, 222, 334, 604

QTIMZON — 334, 604

QTOTJOB — 102, 604

QUERY — 444, 604

QUSCRTUS — 426, 604

QUSLJOB — 425, 604

QUSLOBJ — 425, 604

QUSRBRM — 229, 230, 246, 310, 555, 556, 604

QUSRJOBI — 425, 604

QUSROBJD — 425, 604

QUSRSYS — 129, 144, 448, 604

QUSRTVUS — 426, 604

QUSRWRK — 28, 102, 196, 215, 474, 477, 478, 604

QUTCOFFSET — 334, 604

QVC3001 — 127, 604

QWCRSVAL — 426, 605

QZDASOINIT — 102, 198, 199, 215, 216, 477, 478, 605

R

RANGE — 121, 255, 421, 422, 423, 442, 536, 605

rantai kepercayaan — 432, 605

RBDACCPTH — 85, 605

RCDDLM — 240, 241, 605

RCLSPLSTG — 102, 302, 605

RCLSTG — 21, 95, 222, 224, 226, 283, 302, 456, 605

RCV0001 — 375, 605

RCVDTA0001 — 49, 605

RCVDTA0087 — 95, 605

RCVLEN — 425, 605

RCVMSG — 31, 73, 298, 303, 605

RCVRNG — 49, 60, 78, 98, 232, 245, 325, 375, 605

RCVSIZOPT — 49, 605

RCVVAR — 425, 605

RCYDMN — 482, 605

READC — 540, 605

READE — 113, 605

README — 200, 605

READS — 134, 605

READY — 239, 605

RECORD — 362, 605

RECURSIVE — 134, 605

REFFLD — 544, 605

REFRESH — 444, 605

regenerasi — 17, 93, 605

REGULATOR — 225, 239, 258, 605

REJECT — 457, 605

rekening penampungan — 152, 282, 467, 606

REKENINGR — 80, 113, 114, 118, 131, 253, 254, 360, 380, 606

REKNO — 267, 269, 440, 606

RELEASE — 131, 606

RENCANA — 507, 606

rencana eksekusi — 127, 390, 448, 606

RENDAH — 352, 353, 433, 606

reorganisasi — 55, 84, 85, 94, 99, 103, 129, 159, 300, 416, 442, 446, 448, 456, 513, 606

REPEAT — 237, 241, 438, 606

REPLACE — 90, 141, 164, 200, 201, 206, 236, 238, 240, 241, 264, 277, 289, 291, 307, 321, 360, 365, 495, 606

replikasi asinkron — 437, 606

replikasi sinkron — 37, 390, 437, 438, 485, 486, 606

RERUN — 351, 606

RESET — 271, 606

RESTART — 219, 220, 606

RESTORE — 228, 351, 555, 606

RESTRICT — 132, 222, 606

restricted state — 6, 219, 222, 223, 283, 456, 520, 606

RETAIN — 131, 606

RETPERIOD — 55, 212, 606

RETURN — 29, 62, 68, 134, 151, 167, 170, 429, 606

RETURNS — 134, 606

REVISI — 233, 348, 606

REWRITE — 363, 606

REXEC — 140, 606

RGZPFM — 85, 94, 300, 456, 517, 606

RINCIAN — 501, 504, 606

RINGKASAN — 263, 278, 606

RISIKO — 500, 508, 606

RIWAYAT — 233, 348, 458, 606

RLSJOB — 299, 606

RLSJOBQ — 299, 607

RLSJOBSCDE — 300, 607

RLSSPLF — 238, 302, 607

RMDIR — 304, 607

RMTLOCNAME — 271, 607

RMTSYS — 181, 182, 183, 190, 242, 334, 398, 607

RMVAUTLE — 300, 607

RMVDIR — 304, 607

RMVEXITPGM — 301, 432, 607

RMVJOBSCDE — 299, 607

RMVJRNCHG — 232, 300, 607

RMVLIBLE — 297, 298, 607

RMVPTF — 302, 607

RMVTCPHTE — 180, 607

RMVTCPIFC — 178, 607

RMVTCPRTE — 176, 607

RNMOBJ — 271, 298, 607

RNQ0000 — 73, 607

RNQ0100 — 83, 96, 123, 393, 607

RNQ0103 — 83, 123, 607

RNQ0112 — 83, 123, 607

RNQ0115 — 123, 607

RNQ0202 — 83, 123, 607

RNQ0211 — 83, 123, 607

RNQ0221 — 123, 607

RNQ0231 — 83, 123, 607

RNQ0401 — 123, 607

RNX0000 — 31, 73, 607

RNX0301 — 123, 607

ROUNDED — 363, 371, 607

routing entry — 17, 27, 28, 122, 299, 392, 475, 607

RPGLE — 12, 608

RPGPGM — 308, 461, 608

RPGPPOPT — 201, 360, 608

RPTJOBQ — 513, 608

RPTSBS — 474, 608

RSRCNAME — 177, 179, 608

RSTAUT — 231, 301, 608

RSTCFG — 301, 608

RSTLIB — 78, 98, 222, 231, 232, 272, 297, 301, 332, 374, 608

RSTOBJ — 231, 272, 301, 332, 374, 608

RSTUSRPRF — 301, 608

RTEDEST — 42, 176, 608

RTGDTA — 475, 608

RTNVAR — 33, 34, 43, 77, 170, 222, 313, 608

RTVCFGSTS — 147, 608

RTVCLSRC — 221, 298, 608

RTVDDSSRC — 257, 298, 608

RTVDTAARA — 34, 313, 608

RTVJOBA — 33, 73, 297, 299, 428, 608

RTVNETA — 33, 68, 423, 608

RTVSYSVAL — 33, 35, 43, 77, 170, 222, 608

RUNBOOK — 457, 608

RUNEOD — 421, 422, 424, 608

RUNEODH — 424, 608

RUNPTY — 28, 99, 170, 215, 281, 282, 283, 331, 390, 474, 513, 608

RUNQRY — 25, 300, 608

RUNSQL — 31, 32, 34, 35, 73, 78, 110, 147, 190, 191, 208, 225, 287, 288, 300, 322, 423, 428, 608

RUNSQLSTM — 32, 201, 300, 608

RVKOBJAUT — 298, 608

S

SALAH — 354, 356, 357, 609

SALDOAKHIR — 267, 269, 440, 609

SALINAN — 271, 609

SAMPAI — 271, 350, 609

sampel statistik — 469, 609

SANDI — 233, 609

SANTOSO — 150, 447, 609

SATUAN — 287, 288, 289, 291, 292, 609

SAVACT — 50, 77, 100, 283, 284, 334, 401, 609

SAVACTMSGQ — 50, 77, 609

SAVACTWAIT — 50, 51, 77, 609

SAVCFG — 50, 301, 609

SAVCHGOBJ — 50, 298, 301, 609

SAVDLO — 301, 609

SAVEPOINT — 131, 609

SAVFHAPUS — 225, 609

SAVLIB — 50, 77, 78, 98, 203, 207, 222, 223, 231, 232, 272, 297, 301, 332, 334, 374, 427, 609

SAVMSGQ — 50, 77, 609

SAVOBJ — 50, 68, 77, 260, 272, 301, 609

SAVRST — 41, 262, 279, 457, 609

SAVSECDTA — 50, 301, 609

SAVSYS — 41, 45, 50, 207, 222, 223, 228, 284, 301, 609

SBMJOB — 84, 222, 228, 299, 456, 609

SCDDAY — 51, 609

SCDTIME — 51, 609

SDEBET — 256, 609

SEBELUM — 340, 342, 547, 609

SECADM — 41, 45, 336, 609

SECGRP — 136, 431, 609

SECLIB — 61, 136, 137, 138, 139, 141, 142, 187, 264, 430, 431, 450, 454, 455, 555, 610

SECLVL — 79, 215, 373, 475, 610

SECOWNER — 454, 610

SECTION — 362, 364, 368, 371, 610

SECURITY — 41, 60, 262, 279, 457, 610

SEDANG — 249, 433, 610

SEHARUSNYA — 109, 610

SEKARANG — 260, 610

SELAMA — 349, 350, 351, 610

SELEKSI — 537, 610

SELESAI — 32, 133, 269, 349, 351, 352, 353, 354, 355, 356, 357, 362, 366, 450, 455, 479, 610

SELURUH — 362, 610

SEQNBR — 28, 77, 475, 610

SEQUENCE — 313, 610

serah terima — 250, 610

sertifikat — 5, 9, 22, 107, 138, 144, 183, 189, 207, 306, 333, 339, 382, 387, 402, 409, 432, 433, 434, 485, 503, 527, 610

SERVER — 42, 137, 186, 187, 220, 333, 334, 341, 342, 432, 482, 483, 610

SERVICE — 45, 610

SESUDAH — 342, 547, 610

SETAHUN — 362, 363, 610

SETELAH — 218, 610

SETLL — 113, 610

SETOFF — 358, 610

SETON — 358, 610

SEVERITY — 80, 97, 121, 125, 278, 373, 552, 610

SFLCLR — 256, 537, 538, 540, 610

SFLCTL — 255, 256, 537, 538, 610

SFLDSP — 256, 533, 537, 538, 540, 610

SFLDSPCTL — 256, 537, 538, 540, 610

SFLDTL — 537, 538, 540, 610

SFLEND — 256, 537, 538, 541, 611

SFLPAG — 256, 537, 541, 611

SFLRCDNBR — 541, 611

SFLSIZ — 256, 257, 537, 541, 611

SHRPOOL2 — 28, 84, 214, 335, 611

SHRPOOL3 — 474, 475, 611

SHRPOOL4 — 197, 611

SHRUPD — 85, 611

SIGNATURE — 117, 494, 495, 611

signature — 117, 123, 138, 188, 386, 391, 493, 494, 495, 611

SIGNOFF — 46, 279, 611

siklus hidup — 4, 16, 39, 92, 338, 389, 412, 442, 451, 611

SIMUL — 421, 611

single-level storage — 3, 14, 15, 20, 196, 308, 391, 611

SIRKUIT — 276, 611

SISTEM — 33, 233, 287, 347, 611

sistem operasi — 13, 14, 18, 24, 50, 61, 222, 223, 293, 296, 301, 388, 390, 391, 403, 407, 462, 464, 611

sistem pemantauan — 104, 198, 292, 355, 357, 383, 400, 611

SISTEMDR — 271, 611

SKIPA — 258, 544, 611

SKIPB — 258, 543, 544, 611

SKNBI — 461, 611

SKREDIT — 256, 611

SNAMA — 537, 611

SNDALERT — 48, 611

SNDBRKMSG — 219, 220, 303, 340, 611

SNDDST — 239, 611

SNDMSG — 32, 35, 43, 48, 68, 73, 77, 84, 147, 167, 170, 190, 222, 223, 225, 298, 303, 322, 380, 430, 611

SNDPGMMSG — 30, 31, 33, 34, 35, 68, 73, 77, 303, 322, 368, 373, 423, 425, 427, 428, 429, 611

SNDPTFORD — 277, 302, 611

SNDUSRMSG — 303, 611

SNGVAL — 422, 612

SNOREK — 537, 612

SOLUSI — 278, 612

SOURCE — 81, 90, 117, 201, 360, 365, 371, 612

SPACEA — 258, 543, 544, 612

SPACEB — 258, 543, 544, 612

SPCAUT — 46, 279, 612

SPCVAL — 421, 422, 612

SPECIAL — 362, 612

SPLCTL — 45, 612

SPLHAPUS — 225, 612

SPLIT — 306, 307, 612

split brain — 147, 391, 482, 612

SPLNBR — 238, 612

spooled file — 45, 51, 64, 95, 102, 224, 225, 226, 238, 239, 283, 298, 302, 305, 456, 458, 519, 545, 612

SQLCA — 364, 612

SQLCODE — 124, 329, 364, 365, 387, 496, 547, 612

SQLSTR — 428, 612

SRCDBG — 365, 612

SRCDIR — 90, 612

SRCFILE — 117, 221, 257, 495, 612

SRCMBR — 257, 612

SRCSTMF — 32, 81, 90, 117, 201, 360, 365, 422, 424, 495, 612

SRVPGM — 18, 82, 117, 495, 612

SSALDO — 537, 612

SSLV3 — 188, 612

SSNID — 48, 213, 612

STAGING — 32, 612

STANDARDP — 463, 612

START — 133, 313, 482, 483, 612

STARTING — 442, 612

STATE — 119, 131, 220, 330, 369, 613

STATISTICS — 213, 613

statistik kolom — 126, 129, 305, 447, 613

STDDEV — 289, 613

STMFCCSID — 200, 236, 240, 434, 613

STMFOPT — 200, 236, 238, 613

STOPTRC — 185, 613

STORAGE — 362, 364, 613

STRASPBAL — 214, 302, 314, 613

STRBKUBRM — 228, 301, 613

STRCAD — 484, 613

STRCLNUP — 225, 302, 613

STRCMNTRC — 185, 302, 613

STRCMTCTL — 119, 300, 613

STRCRG — 482, 613

STRDBG — 82, 303, 366, 456, 613

STRDLM — 240, 241, 613

STRDW — 213, 303, 469, 613

STRIP — 67, 613

STRJRNPF — 49, 271, 300, 329, 375, 613

STRJWMON — 213, 303, 469, 613

STRPDM — 303, 613

STRPFRCOL — 55, 212, 303, 463, 613

STRPGMEXP — 117, 494, 613

STRPRTWTR — 238, 302, 519, 613

STRQSH — 304, 613

STRRCYBRM — 228, 301, 555, 613

STRSBS — 18, 27, 28, 197, 222, 223, 297, 299, 342, 345, 483, 613

STRSEU — 303, 613

STRSQL — 300, 613

STRSRVJOB — 82, 303, 366, 456, 613

STRTCP — 186, 223, 301, 345, 614

STRTCPIFC — 124, 178, 301, 614

STRTCPSVR — 42, 137, 186, 187, 301, 333, 334, 342, 432, 482, 483, 614

STRTIME — 225, 614

STRWCH — 48, 303, 614

STSVAR — 147, 614

STUDI — 218, 470, 614

subfile — 10, 255, 256, 257, 391, 533, 534, 537, 538, 539, 540, 541, 614

SUBNETMASK — 42, 176, 178, 179, 614

SUBSTR — 62, 67, 82, 84, 150, 151, 157, 232, 614

SUBSYSTEM — 25, 48, 96, 99, 104, 189, 197, 307, 558, 614

SUCCESS — 162, 163, 614

SUKSES — 351, 614

SUMBER — 266, 267, 269, 346, 436, 438, 439, 440, 507, 552, 614

SVCBUNGA — 115, 117, 614

SVCRATE — 81, 82, 614

SVCTGL — 82, 614

SVRSPCVAL — 482, 614

SWITCHOVER — 483, 614

switchover — 8, 38, 59, 99, 145, 146, 148, 179, 207, 246, 340, 341, 342, 344, 347, 384, 387, 388, 391, 402, 483, 486, 488, 614

SYMBOL — 117, 494, 614

SYNCLIB — 50, 77, 100, 283, 284, 334, 401, 614

SYNCOPT — 484, 614

SYSCOLUMNS — 236, 244, 305, 327, 516, 603, 614

SYSCPU — 464, 614

SYSCST — 132, 305, 330, 603, 614

SYSDUMMY1 — 69, 87, 147, 269, 271, 318, 321, 322, 328, 334, 440, 480, 516, 552, 614

SYSENG — 32, 43, 48, 68, 84, 147, 167, 170, 190, 222, 223, 225, 614

SYSIBM — 69, 87, 147, 269, 271, 318, 321, 322, 328, 334, 440, 480, 516, 552, 614

SYSINDEXES — 305, 603, 614

SYSIXADV — 55, 106, 305, 603, 615

SYSLVL — 462, 615

SYSMSG — 225, 615

SYSNAME — 33, 68, 423, 615

SYSOPR — 380, 615

SYSTABLES — 305, 444, 604, 615

SYSTEM — 49, 444, 615

SYSTOOLS — 87, 306, 307, 615

SYSVAL — 33, 41, 43, 60, 77, 102, 139, 170, 186, 214, 215, 221, 222, 235, 334, 337, 484, 615

SYSVIEWS — 305, 604, 615

T

TABEL — 366, 615

tabel kontrol — 72, 283, 318, 322, 377, 382, 393, 511, 549, 615

tabel parameter — 83, 94, 123, 133, 165, 327, 393, 396, 550, 552, 555, 556, 615

TABELUJI — 374, 375, 615

TAHAP — 340, 342, 615

TAHUNAN — 228, 615

TAMBAHKAN — 355, 356, 615

TANGGAL — 33, 439, 480, 507, 615

tanggal buku — 34, 69, 85, 98, 281, 282, 283, 284, 311, 318, 321, 332, 377, 387, 420, 421, 442, 444, 458, 551, 615

TANGKAP — 349, 356, 615

TAPMLB01 — 50, 77, 78, 98, 223, 231, 232, 332, 334, 615

tata kelola — 4, 10, 85, 89, 93, 199, 308, 331, 421, 498, 520, 528, 615

TCP0000 — 190, 615

TCP1A04 — 124, 615

TCP2665 — 124, 615

TCPIPADR — 185, 615

TCPL0300 — 187, 615

TELLER01 — 552, 615

TELLER07 — 558, 615

TELLER14 — 557, 558, 616

TELLER22 — 558, 616

TELNET — 42, 183, 186, 616

TEMUAN — 470, 507, 616

temuan audit — 42, 46, 62, 139, 142, 187, 203, 239, 294, 296, 301, 335, 390, 415, 452, 507, 519, 616

TENGGAT — 507, 616

tenggat keputusan — 472, 616

TERLEWAT — 271, 616

TERPAKAI — 428, 429, 616

TESTGRP — 202, 616

TESTLIB — 22, 616

TESTN — 358, 616

TGLBUKA — 267, 616

TGLBUKU — 34, 69, 74, 368, 421, 616

TGLCETAK — 543, 616

TGLLHR — 530, 616

THRESHOLD — 49, 215, 216, 477, 616

TIKET — 68, 422, 423, 430, 450, 616

time slice — 27, 299, 474, 616

TIMESLICE — 28, 170, 474, 616

TIMEZONE — 334, 616

TIMLMT — 214, 314, 616

TIMMRKTIMO — 186, 616

TINDAK — 218, 616

TINDAKAN — 263, 430, 431, 616

TINGGI — 352, 353, 355, 433, 616

TINGKAT — 287, 288, 289, 428, 507, 616

TINJAU — 455, 616

tinjauan akses — 9, 450, 455, 456, 504, 616

TIPEJOB — 33, 616

TITIK — 326, 616

titik panas — 152, 285, 467, 469, 617

titik pulih — 77, 100, 229, 283, 329, 334, 617

TLSV1 — 139, 188, 617

TOCCSID — 236, 617

TOENT — 232, 617

TOFILE — 241, 543, 617

TOINTNET — 239, 617

TOLAK — 268, 269, 270, 440, 617

TOLIB — 55, 92, 463, 617

TOMBR — 200, 236, 617

TOMSGQ — 219, 220, 340, 617

TOOBJ — 236, 617

TOOLS — 307, 415, 617

TOPGMQ — 31, 68, 373, 617

TOSTMF — 200, 236, 238, 240, 545, 617

TOTCPU — 464, 465, 617

TOTIME — 49, 78, 98, 232, 375, 456, 617

TOUSR — 32, 43, 48, 68, 73, 77, 84, 147, 167, 170, 190, 222, 223, 225, 322, 380, 617

TRACEROUTE — 173, 179, 182, 283, 302, 379, 617

transaksi ganda — 332, 339, 526, 617

TRANSFER — 119, 617

transfer pengetahuan — 93, 418, 617

TRANSLATE — 150, 237, 240, 241, 438, 617

TRCFULL — 185, 617

TRXGRP — 118, 617

TRXHIST — 85, 127, 128, 129, 132, 133, 134, 152, 211, 244, 311, 313, 330, 333, 340, 342, 345, 380, 442, 443, 447, 448, 617

TRXMASUK — 552, 617

TSTBUNGA — 202, 617

TSTSUITE — 203, 617

TUJUAN — 454, 617

TUNGGU — 34, 35, 352, 618

TUNGGUJOB — 84, 618

TUTUP — 111, 349, 362, 363, 618

TYPE4 — 49, 232, 375, 618

TYPE5 — 60, 618

U

UATDTA — 68, 618

UATPGM — 327, 618

uji restore — 16, 51, 85, 93, 226, 245, 400, 401, 406, 409, 410, 414, 458, 503, 509, 510, 556, 560, 618

UJILIB — 543, 618

UKURAN1 — 167, 618

UKURAN2 — 167, 618

ULANG — 421, 423, 618

UNBOUNDED — 133, 618

UNLOAD — 223, 618

UNTIL — 362, 365, 618

UPDPGM — 303, 618

UPDPROD — 82, 366, 456, 618

UPDSRVPGM — 303, 618

UPPER — 30, 150, 438, 618

URAIAN — 507, 508, 554, 618

URUTAN — 73, 74, 165, 233, 243, 321, 347, 479, 618

USAGE — 107, 618

USEADPAUT — 422, 454, 618

user space — 425, 426, 618

USERID — 62, 136, 137, 138, 151, 430, 431, 450, 454, 455, 618

USRCLS — 46, 279, 618

USRDFN — 139, 618

USRDTA — 185, 258, 545, 618

USRMSG — 225, 619

USRPRF — 18, 43, 46, 141, 144, 235, 262, 263, 279, 298, 422, 454, 482, 484, 619

UTILITY — 81, 619

UTILLIB — 429, 483, 555, 619

UTLALERT — 427, 483, 619

UTLCEKENV — 427, 619

UTLJOBSTS — 428, 619

UTLLOCK — 427, 429, 619

UTLLOG — 427, 429, 483, 619

UTLPARAM — 427, 619

UTLTUNGGU — 427, 619

UTLUNLOCK — 427, 619

V

VALIDASI — 421, 495, 619

VALUE — 29, 30, 32, 33, 34, 41, 48, 60, 68, 77, 84, 102, 139, 186, 214, 215, 313, 326, 337, 362, 368, 373, 425, 426, 427, 428, 429, 430, 483, 619

VECTOR — 128, 619

VENDOR — 278, 619

VENDOR01 — 141, 279, 619

VERBOSE — 181, 619

VERIFIKASI — 78, 233, 457, 470, 619

verifikasi pasca-failover — 147, 619

VERIFPOST — 483, 619

VERIFY — 457, 619

VERSI — 163, 164, 619

VERSION — 482, 619

VERSIONING — 133, 619

VIRTUALIP — 179, 619

VLANID — 177, 619

VLDCKR — 422, 619

VLRQ0100 — 187, 619

volume transaksi — 36, 93, 151, 152, 155, 211, 296, 310, 382, 392, 410, 441, 620

VRYCFG — 100, 123, 146, 177, 302, 334, 342, 345, 620

W

wait accounting — 6, 10, 212, 392, 465, 620

WAITTIME — 181, 182, 190, 620

WAJIB — 82, 165, 360, 479, 620

WATCH — 82, 326, 366, 620

WATCHOPS — 48, 620

WCHHANDLER — 48, 620

WCHMSG — 48, 620

WCHMSGQ — 48, 620

WCHOPT — 48, 620

WCHPGM — 48, 620

WEBPROD — 186, 187, 333, 342, 483, 620

WEEKLY — 51, 620

WEWENANG — 454, 620

wewenang adopsi — 262, 422, 620

wewenang berlebih — 46, 264, 335, 339, 454, 527, 620

WHFNAM — 100, 120, 245, 259, 325, 480, 542, 620

WHLNAM — 120, 245, 259, 325, 480, 542, 620

WHOTYP — 120, 620

WHPNAM — 100, 120, 245, 259, 325, 480, 542, 620

WHSNAM — 542, 620

WHUSAG — 120, 245, 325, 480, 542, 620

WITHIN — 291, 620

WKUNCI — 119, 620

WORKING — 362, 364, 620

WRITE — 380, 620

WRKACTJOB — 12, 31, 35, 47, 89, 99, 104, 196, 213, 222, 297, 299, 456, 620

WRKCADMRE — 144, 484, 620

WRKCFGSTS — 145, 146, 173, 178, 302, 334, 342, 621

WRKCLU — 99, 145, 341, 482, 621

WRKCMTDFN — 300, 621

WRKCRG — 145, 621

WRKCTLGBRM — 228, 621

WRKDSKSTS — 20, 47, 95, 99, 302, 456, 621

WRKFCNUSG — 301, 621

WRKHDWRSC — 177, 621

WRKJOB — 12, 75, 245, 284, 299, 621

WRKJOBQ — 18, 27, 47, 299, 456, 621

WRKJOBSCDE — 51, 299, 621

WRKJRN — 18, 300, 621

WRKJRNA — 47, 49, 57, 300, 337, 344, 456, 621

WRKLICINF — 293, 302, 621

WRKLNK — 304, 621

WRKMEDBRM — 228, 301, 621

WRKMEDIBRM — 100, 228, 231, 301, 621

WRKMLBSTS — 334, 621

WRKMSGD — 303, 621

WRKMSGQ — 19, 303, 621

WRKOBJ — 463, 621

WRKOBJLCK — 75, 96, 104, 298, 456, 621

WRKOUTQ — 18, 238, 302, 456, 621

WRKPCYBRM — 228, 621

WRKPFRCOL — 463, 621

WRKPRB — 303, 621

WRKPTFGRP — 39, 277, 302, 388, 456, 621

WRKREGINF — 61, 301, 621

WRKSBMJOB — 299, 621

WRKSBS — 18, 27, 299, 456, 621

WRKSHRPOOL — 299, 475, 621

WRKSPLF — 238, 302, 327, 456, 622

WRKSYSACT — 299, 622

WRKSYSSTS — 20, 27, 47, 54, 95, 96, 99, 104, 251, 283, 299, 456, 622

WRKSYSVAL — 41, 43, 300, 622

WRKUSRJOB — 299, 622

WRKUSRPRF — 300, 456, 622

WRKWCH — 48, 303, 622

WRKWTR — 238, 302, 519, 622

WSALDO — 88, 119, 622

WSCST — 238, 545, 622

Y

YEARS — 19, 92, 622

YYYYMMDD — 241, 267, 622

Z

ZDAI0100 — 61, 137, 432, 622