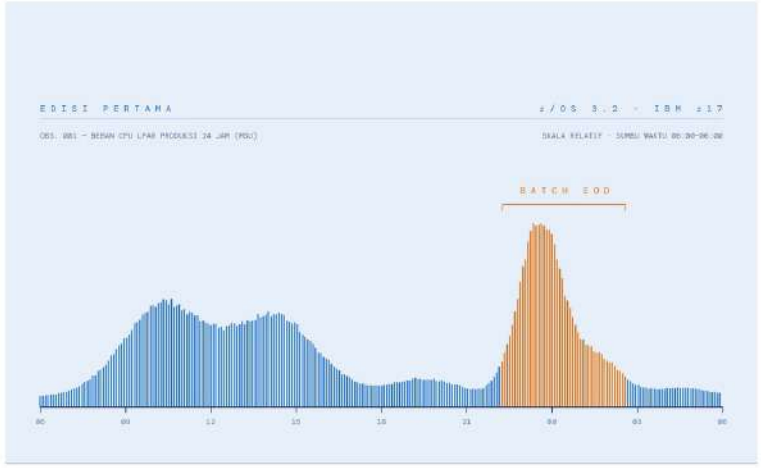



PANDUAN INSINYUR SISTEM

# HANDBOOK OF SYSTEM ENGINEERING IBM Z · MAINFRAME

Arsitektur · Operasi · Jaringan · Core Banking

**Romi Nur Ismanto**

romiNur.com · hello@romiNur.com

100 BAB · 60 RUNBOOK · 132 GAMBAR

# Handbook of System Engineering for IBM Z (Mainframe)

🏦 **Arsitektur, Operasi, Jaringan, dan Core Banking**

**Romi Nur Ismanto**

rominur.com · hello@rominur.com

Edisi Pertama · 20 September 2026

# Daftar Isi

📖 Tentang Buku Ini ..... 12

🧭 **Bagian I — Fondasi ..... 14**

Bab 1. Posisi IBM Z dalam Arsitektur TI Modern ..... 14
Bab 2. Evolusi Platform ..... 17
Bab 3. Peran dan Kompetensi System Engineer z/OS ..... 21

🏗️ **Bagian II — Arsitektur Platform ..... 25**

Bab 4. Anatomi Central Processor Complex ..... 25
Bab 5. PR/SM, LPAR, HMC, dan SE ..... 28
Bab 6. Subsistem I/O ..... 30
Bab 7. Storage Hardware ..... 34
Bab 8. Sistem Operasi di IBM Z ..... 36
Bab 9. Komponen Inti z/OS ..... 37
Bab 10. Address Space dan Virtual Storage ..... 40

🛠️ **Bagian III — Perencanaan dan Implementasi ..... 45**

Bab 11. Desain Topologi LPAR dan Sysplex ..... 45
Bab 12. Proses IPL ..... 47
Bab 13. IODF dan HCD ..... 53
Bab 14. Instalasi dan Pemeliharaan dengan SMP/E ..... 55
Bab 15. Baseline dan Manajemen Konfigurasi ..... 58

💾 **Bagian IV — Dataset dan Manajemen Storage ..... 61**

Bab 16. Dataset ..... 61
Bab 17. Katalog ..... 64
Bab 18. DFSMS dan Storage Group ..... 67

⚙️ **Bagian V — Batch: JCL dan JES2 ..... 71**

Bab 19. Anatomi JCL ..... 71
Bab 20. Utilitas Standar ..... 75
Bab 21. JES2 ..... 78
Bab 22. Penjadwalan Batch dan Restart ..... 80

🖥️ **Bagian VI — Antarmuka dan Otomasi ..... 83**

Bab 23. TSO/E dan ISPF ..... 83
Bab 24. SDSF ..... 85
Bab 25. z/OS UNIX System Services ..... 87
Bab 26. REXX untuk System Engineer ..... 89

🔐 **Bagian VII — Keamanan ..... 92**

Bab 27. Konsep RACF ..... 92
Bab 28. Administrasi RACF Sehari-hari ..... 95
Bab 29. Integritas Sistem ..... 97
Bab 30. Audit dan SMF ..... 99
Bab 31. Enkripsi dan Kriptografi ..... 101

🏦 **Bagian VIII — Middleware dan Core Banking ..... 104**

Bab 32. CICS Transaction Server ..... 104
Bab 33. Db2 for z/OS ..... 106
Bab 34. IMS dan IBM MQ ..... 109
Bab 35. Lanskap Core Banking di IBM Z ..... 111

🌐 **Bagian IX — Jaringan ..... 115**

Bab 36. z/OS Communications Server ..... 115
Bab 37. VIPA dan Sysplex Distributor ..... 116
Bab 38. Diagnosis Jaringan ..... 119
Bab 39. Transfer File dan Interface ..... 121

📈 **Bagian X — Kinerja dan Kapasitas ..... 124**

Bab 40. Workload Manager ..... 124
Bab 41. SMF dan RMF ..... 127
Bab 42. Capacity Planning dan Biaya ..... 130

🛡️ **Bagian XI — Ketersediaan, Backup, dan Pemulihan Bencana ..... 134**

Bab 43. Parallel Sysplex ..... 134
Bab 44. HyperSwap dan GDPS ..... 136
Bab 45. Backup dengan DFSMShsm dan DFSMSdss ..... 138
Bab 46. Merancang Latihan DR yang Bermakna ..... 141

📋 **Bagian XII — Operasi Harian dan Referensi Perintah ..... 144**

Bab 47. Irama Operasional ..... 144

Bab 48. Referensi Perintah Console z/OS.....146

🚨 **Bagian XIII — Troubleshooting dan Katalog Runbook.....150**

Bab 49. Katalog Abend Sistem.....150
Bab 50. Katalog Pesan Operasional.....152
Bab 51. Dump dan IPCS.....154
Bab 52. Katalog Runbook.....157

🚀 **Bagian XIV — Integrasi dan Modernisasi.....162**

Bab 53. Strategi Modernisasi.....162
Bab 54. API dan Konektivitas.....164
Bab 55. DevOps untuk z/OS.....166
Bab 56. IBM Z dan Hybrid Cloud.....168

🧮 **Bagian XV — Algoritma Operasional.....171**

Bab 57. Algoritma Diagnosis Job Batch Gagal.....171
Bab 58. Algoritma Restart Fase EOD yang Aman.....173
Bab 59. Algoritma Respons Kapasitas.....175
Bab 60. Algoritma Rolling IPL dalam Sysplex.....177
Bab 61. Algoritma Verifikasi Pasca-EOD.....180
Bab 62. Algoritma Analisis Kinerja Top-Down.....182

🧩 **Bagian XVI — Pendalaman CICS.....184**

Bab 63. Arsitektur Region dan Topologi.....184
Bab 64. Definisi Resource dan Parameter Sistem.....186
Bab 65. Storage, TCB, dan Kinerja.....189
Bab 66. VSAM RLS dan CICS.....192
Bab 67. Diagnosis CICS.....194

🗄️ **Bagian XVII — Pendalaman Db2 for z/OS.....198**

Bab 68. Arsitektur Internal Db2.....198
Bab 69. Buffer Pool dan Memori.....200
Bab 70. Logging, Locking, dan Commit.....203
Bab 71. Data Sharing.....205
Bab 72. Utilitas, Recovery, dan Access Path.....207

📜 **Bagian XVIII — COBOL untuk System Engineer.....214**

Bab 73. Anatomi Program COBOL....214
Bab 74. Tipe Data dan Representasi di Memori....217
Bab 75. Compile, Link, dan Deploy....220
Bab 76. Language Environment....223
Bab 77. Algoritma Analisis S0C7....226

🔬 **Bagian XIX — HLASM dan Membaca Dump....230**

Bab 78. Register dan PSW....230
Bab 79. Dari Abend ke Baris Source....232
Bab 80. Konvensi Linkage dan Rantai Save Area....235
Bab 81. Studi Kasus Analisis Dump....239

🤖 **Bagian XX — REXX Lanjutan dan Otomasi....242**

Bab 82. Parsing dan Struktur Data....242
Bab 83. Membaca Dataset dan Menangkap Output....244
Bab 84. Penanganan Error yang Disiplin....246
Bab 85. Otomasi Console....248
Bab 86. Skrip Health Check Harian Lengkap....250
Bab 87. Zowe CLI untuk Otomasi dari Luar Mainframe....254

📕 **Bagian XXI — Katalog Runbook Lanjutan....257**

Bab 88. Indeks Runbook....257
Bab 89. Runbook Storage dan Dataset....260
Bab 90. Runbook Batch dan JES2....263
Bab 91. Runbook Sistem dan Sysplex....265
Bab 92. Runbook Keamanan....268
Bab 93. Runbook Jaringan dan Middleware....270
Bab 94. Runbook Hardware....272
Bab 95. Memelihara Katalog Runbook....274

🏛️ **Bagian XXII — Modul Core Banking....278**

Bab 96. Customer Information File....278
Bab 97. Tabungan dan Giro....281
Bab 98. Deposito Berjangka....284
Bab 99. Kredit....286
Bab 100. General Ledger....290

💸 **Bagian XXIII — Pembayaran dan Interface.....294**

Bab 101. Lanskap Sistem Pembayaran.....294
Bab 102. Arsitektur Integrasi dan Idempotensi.....296
Bab 103. Suspense dan Rekonsiliasi.....299

🌙 **Bagian XXIV — Pendalaman End of Day.....301**

Bab 104. Anatomi Batch Window.....301
Bab 105. Tabel Kontrol Fase.....303
Bab 106. Kalender Bisnis dan Periode Khusus.....306
Bab 107. Skenario EOD Bermasalah dan Runbook.....308
Bab 108. Metrik dan Pencegahan.....310

🧾 **Bagian XXV — Pelaporan Regulator.....312**

Bab 109. Lanskap Pelaporan.....312
Bab 110. Pipeline Data Pelaporan dari Mainframe.....313

🚨 **Bagian XXVI — Respons Insiden Keamanan.....317**

Bab 111. Prinsip Penanganan.....317
Bab 112. Deteksi dan Pengamanan Bukti.....319
Bab 113. Ransomware, Korupsi Logis, dan Cyber Vault.....322
Bab 114. Runbook Insiden Keamanan.....325

🔁 **Bagian XXVII — Prosedur Pemulihan Bencana Mendalam.....327**

Bab 115. Switchover Terencana.....327
Bab 116. Failover Tak Terencana.....331
Bab 117. Dokumen Pemulihan yang Harus Dicetak.....334

📖 **Bagian XXVIII — Anatomi Insiden.....335**

Bab 118. Narasi Pertama: Malam Ketika Akrual Hampir Berjalan Dua Kali .....335
Bab 119. Narasi Kedua: Cabang yang Lambat Selama Tiga Pekan.....336
Bab 120. Narasi Ketiga: Backup yang Enam Bulan Tidak Lengkap.....338
Bab 121. Narasi Keempat: Uji Pemulihan yang Gagal Sebagian.....339
Bab 122. Benang Merah Empat Narasi.....341

📅 **Bagian XXIX — Setahun dalam Hidup System Engineer.....342**

Bab 123. Kuartal Pertama.....342

Bab 124. Kuartal Kedua....343
Bab 125. Kuartal Ketiga....343
Bab 126. Kuartal Keempat....343
Bab 127. Irama Mingguan dan Harian....345

🗂️ **Bagian XXX — Manajemen Proyek Teknis....347**

Bab 128. Proyek Upgrade z/OS....347
Bab 129. Proyek Migrasi CPC....350
Bab 130. Cutover....352
Bab 131. Risiko dan Rollback....355

🧪 **Bagian XXXI — Praktikum....358**

Bab 132. Menyiapkan Lingkungan Lab....358
Bab 133. Praktikum 1: Dataset, JCL, dan GDG....358
Bab 134. Praktikum 2: Backup dan Restore....361
Bab 135. Praktikum 3: Mengukur Biaya CPU....363
Bab 136. Praktikum 4: EOD Mini dengan Restart....365
Bab 137. Praktikum 5: Melindungi Dataset dengan RACF....367

📝 **Bagian XXXII — Latihan Soal Lanjutan....370**

Bab 138. Soal Pilihan Ganda....370
Bab 139. Soal Situasi....375
Bab 140. Kunci Jawaban....375

🎯 **Bagian XXXIII — Proyek Studi Mandiri....378**

Bab 141. Proyek Tingkat Menengah....378
Bab 142. Proyek Tingkat Lanjut....379
Bab 143. Rubrik Penilaian....380

🔤 **Bagian XXXIV — CCSID, EBCDIC/Unicode, dan Pertukaran Data....382**

Bab 144. Code Page dan CCSID....382
Bab 145. Di Mana Konversi Terjadi....384
Bab 146. Angka dan Urutan Lintas Platform....386
Bab 147. Validasi File Interface....388

🔒 **Bagian XXXV — Jaringan dan Keamanan Transport Mendalam....392**

Bab 148. AT-TLS....392

Bab 149. Keyring dan Sertifikat di RACF....395
Bab 150. Penyaringan Lalu Lintas IP....398
Bab 151. Runbook Transport....399

🧬 **Bagian XXXVI — Db2 Lanjutan....401**

Bab 152. Partisi Tabel....401
Bab 153. Materialized Query Table....403
Bab 154. Objek Besar (LOB)....404
Bab 155. Kendali Sumber Daya Kueri....406
Bab 156. REORG Berbasis Real-Time Statistics....408

✍️ **Bagian XXXVII — Esai Teknis....410**

Bab 157. Mengapa Integritas Dibangun ke Dalam, Bukan Ditambahkan ....410
Bab 158. Ekonomi Keandalan....410
Bab 159. Tentang Dokumentasi yang Sungguh Dipakai....412

🐧 **Bagian XXXVIII — z/VM dan Linux on IBM Z....413**

Bab 160. Arsitektur z/VM....413
Bab 161. SSI dan Live Guest Relocation....415
Bab 162. Linux on IBM Z dari Sudut Pandang Operasi....417
Bab 163. Konsolidasi dan Kinerja....419

📨 **Bagian XXXIX — IBM MQ dan IMS Mendalam....421**

Bab 164. Arsitektur Queue Manager di z/OS....421
Bab 165. Channel dan Keandalan Pesan....423
Bab 166. Keamanan MQ....425
Bab 167. IMS untuk System Engineer z/OS....427

🎛️ **Bagian XL — WLM Mendalam dan Analisis SMF dengan Python....431**

Bab 168. Rumus di Balik WLM....431
Bab 169. Period, Resource Group, dan Pola Harian....433
Bab 170. Mengambil Data SMF....435
Bab 171. Parser Header SMF dengan Python....436
Bab 172. Dari Data ke Keputusan Kapasitas....439

🤝 **Bagian XLI — Operasi Tim, Vendor, dan Lisensi....442**

Bab 173. Jaga dan Eskalasi....442
Bab 174. Komunikasi Saat Insiden....444
Bab 175. Postmortem Tanpa Menyalahkan....446
Bab 176. IBM Support dan Vendor....447
Bab 177. Lisensi dan Total Biaya Kepemilikan....449

🔗 **Bagian XLII — Parallel Sysplex Mendalam....452**

Bab 178. Signaling XCF....452
Bab 179. Structure dan Duplexing....454
Bab 180. SFM, ARM, dan GRS Star....457
Bab 181. Algoritma Pemeriksaan Kesehatan Sysplex Harian....458

🧱 **Bagian XLIII — Hardening dan Monitoring Terpadu....461**

Bab 182. Checklist Hardening z/OS....461
Bab 183. Arsitektur Monitoring Terpadu....463
Bab 184. Integrasi dengan SIEM....465
Bab 185. Metrik Observability untuk Mainframe....467

🧩 **Bagian XLIV — Studi Kasus Troubleshooting....469**

Bab 186. Kasus Storage, Batch, dan Aplikasi....469
Bab 187. Kasus Operasi dan Konfigurasi....471
Bab 188. Kasus Kapasitas, Pemantauan, dan Keamanan....473
Bab 189. Pola dari Dua Puluh Kasus....475

📑 **Bagian XLV — Referensi Perintah Ringkas....489**

Bab 190. Lembar Contekan per Area....489
Bab 191. Triage: Gejala ke Perintah Pertama....491

🌅 **Bagian XLVI — Penutup....494**

Bab 192. Peta Jalan Pengembangan Diri....494
Bab 193. Kata Penutup....496

📚 **Lampiran....498**

Lampiran A. Peta Konsep IBM i ke IBM Z....498
Lampiran B. Glosarium (A–Z)....499
Lampiran C. Latihan Soal....505
Lampiran D. Peta Jalan Pengembangan Diri....506

Lampiran E. Daftar Pustaka....506
Lampiran F. Praktikum Lanjutan....508
Lampiran G. Analisis Lanjutan....515
Lampiran H. Bank Soal dan Studi Latihan....525
Lampiran I. Kumpulan Rumus....531
Lampiran J. Templat Kerja....536
Lampiran K. Glosarium Tambahan....540
Lampiran L. Studi Kasus K-41 sampai K-60....543
Lampiran M. Peta Silang Gejala....553
Lampiran N. Menyesuaikan dengan Ukuran Bank....555
Lampiran O. Rencana 90 Hari....561
Lampiran P. Jawaban Model Studi Latihan....565
Lampiran Q. Studi Kasus Berseri: Setahun di Bank Nusantara....571
Lampiran R. Catatan Implementasi per Komponen....577
Lampiran S. Soal Latihan per Bagian....590
Lampiran T. Tentang Buku Ini....595
Indeks Tambahan....598
Indeks Subjek....602

# 📖 Tentang Buku Ini

Buku ini adalah pegangan kerja system engineer yang mengelola IBM Z dengan z/OS, khususnya di lingkungan perbankan. Susunannya meniru *Handbook of System Engineering for IBM i (AS/400)*: fondasi, arsitektur, implementasi, operasi, keandalan, core banking, modernisasi, lalu katalog runbook dan referensi perintah.

Fokus utama adalah z/OS. z/VM dan Linux on IBM Z dibahas secukupnya untuk memahami konsolidasi. Setiap bab berusaha menjawab tiga hal: apa konsepnya, perintah apa yang dipakai, dan apa yang harus dilakukan ketika terjadi masalah.

Konvensi penulisan:

- Perintah console operator ditulis dengan awalan `D`, `F`, `S`, `P`, `V`, dan diketik di console MCS/SMCS atau lewat SDSF.
- Perintah TSO ditulis apa adanya, misalnya `LISTCAT`, `LU`, `ALLOC`.
- Nama dataset contoh memakai HLQ fiktif `BANK`, `SYS1` untuk dataset sistem, dan `SYSn` untuk nama sistem dalam sysplex.
- Suffix member PARMLIB ditulis `xx`, misalnya `IEASYSxx` berarti `IEASYS00`, `IEASYS01`, dan seterusnya.
- Angka versi mengacu pada z/OS 3.2 dan IBM z17 kecuali disebutkan lain.

## Sasaran pembaca

| Pembaca | Bagian yang paling relevan |
|---|---|
| System programmer z/OS baru | I, II, III, IV, V |
| Operator senior yang naik ke peran engineer | V, VI, XII, XIII |
| Engineer IBM i yang pindah ke mainframe | I, II, Lampiran A |
| Application Support core banking | V, VIII, XIII, XV, XVI, XVII |
| Arsitek dan technical lead | II, X, XI, XIV |
| Auditor TI dan risk officer | VII, XI, Bab 61 |

## Peringatan penting soal kerahasiaan

Seluruh contoh yang menyangkut core banking bersifat generik dan teranonimkan. Tidak ada nama dataset, nama job, region CICS, struktur tabel, atau parameter dari sistem produksi bank mana pun.

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 VI. Pembaca berpengalaman bisa langsung ke Bagian X, XI, atau XV, tetapi Bab 12 (IPL) dan Bab 21 (JES2) adalah prasyarat untuk hampir semua pembahasan operasional setelahnya.

# 🧭 Bagian I — Fondasi

## Bab 1. Posisi IBM Z dalam Arsitektur TI Modern

IBM Z tetap menjadi sistem pencatatan (*system of record*) utama untuk transaksi keuangan bervolume tinggi. Sebagian besar bank besar dunia, termasuk beberapa bank BUMN dan swasta di Indonesia, menjalankan core banking, kartu, dan switching di atasnya.

Mainframe tidak bersaing di harga per core. Ia bersaing di throughput transaksi, integritas data, dan ketersediaan yang sulit ditiru arsitektur terdistribusi tanpa biaya rekayasa besar.

### 1.1 Kapan IBM Z unggul

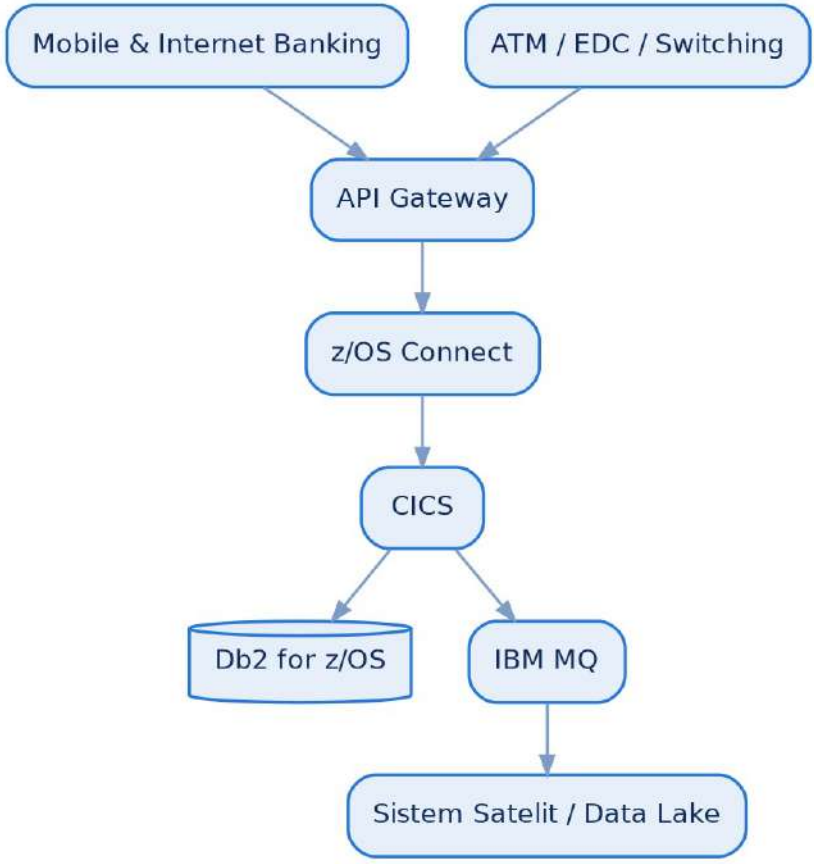
| Karakteristik beban kerja | Alasan IBM Z unggul |
|---|---|
| OLTP bervolume sangat tinggi (ribuan TPS) | CICS dan Db2 dirancang untuk ini sejak 1970-an; I/O dilayani prosesor khusus (SAP) |
| Batch malam dengan tenggat ketat | JES2, WLM, dan I/O channel paralel memproses jutaan record per jam |
| Integritas data kritikal | Two-phase commit, RRS, dan data sharing Db2 di Parallel Sysplex |
| Ketersediaan 99,999% | Parallel Sysplex, GDPS, dan komponen hardware redundan |
| Regulasi keamanan ketat | RACF, pervasive encryption, kriptografi hardware bersertifikat FIPS 140-2/140-3 |
| Konsolidasi server Linux | IFL dan z/VM menjalankan ratusan guest Linux dalam satu mesin |

### 1.2 Kapan bukan pilihan terbaik

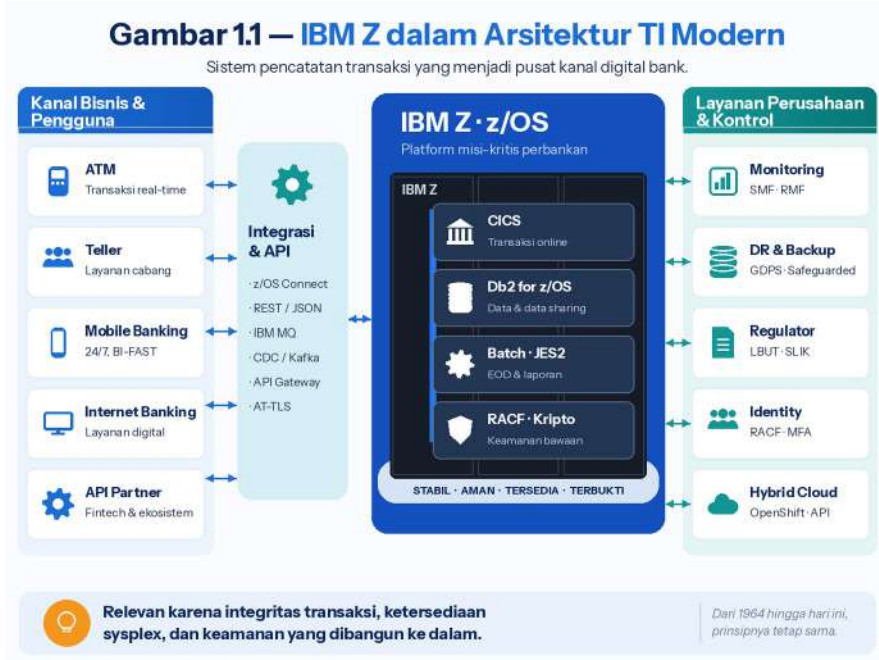
- Aplikasi baru tanpa ketergantungan data mainframe dan tanpa kebutuhan transaksi berat.
- Beban kerja yang sangat elastis dan berumur pendek, misalnya lingkungan uji sementara.

- Organisasi yang tidak sanggup mempertahankan tim dengan keahlian z/OS.

Posisi realistis hari ini adalah **hybrid**: mainframe sebagai inti transaksi, dengan kanal digital di cloud atau Kubernetes yang memanggilnya lewat API.




Kanal digital masuk lewat gateway, diterjemahkan menjadi transaksi CICS, dan data tetap tersimpan di Db2 atau VSAM di mainframe.




*Gambar 1.1 — IBM Z dalam arsitektur TI modern*

## 1.3 Analisis: Menilai Platform dengan Biaya per Transaksi

Perdebatan tentang mainframe sering berhenti di satu angka: harga mesin atau tagihan lisensi bulanan. Angka itu memang besar, tetapi tidak bermakna tanpa pembandingnya, yaitu jumlah kerja yang dihasilkan. Ukuran yang lebih adil adalah biaya per transaksi, dihitung dari seluruh biaya kepemilikan (Bab 177).

$$C_{\text{per transaksi}} = \frac{TCO_{\text{tahunan}}}{N_{\text{transaksi per tahun}}}$$

Contoh ilustratif: TCO tahunan Rp150 miliar untuk platform yang memproses 3 miliar transaksi setahun menghasilkan sekitar Rp50 per transaksi. Angka itu baru bermakna bila dibandingkan dengan biaya per transaksi alternatifnya, yang dihitung dengan cara yang sama lengkapnya.

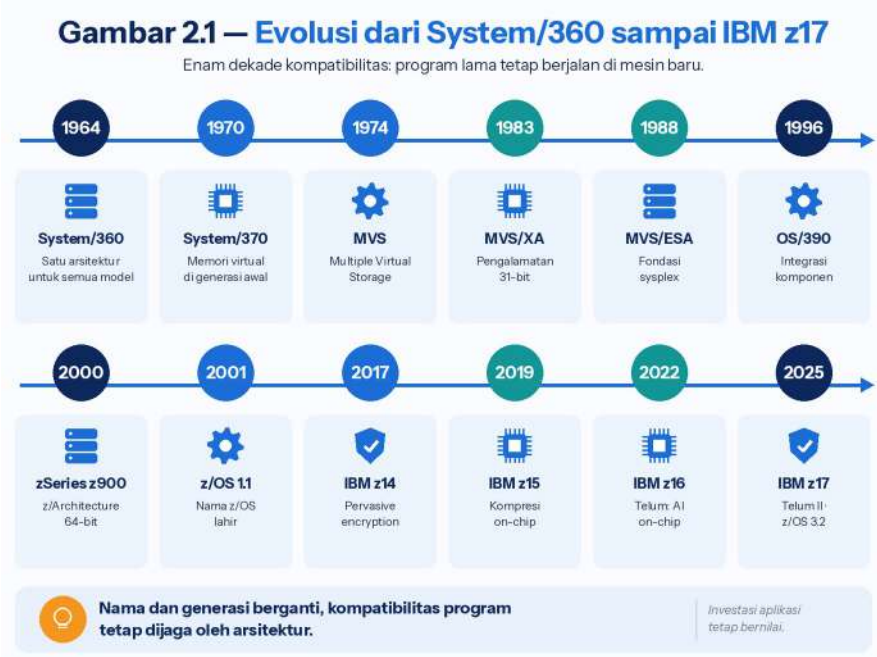
| Komponen yang sering terlupa saat membandingkan | Mengapa penting |
|---|---|
| Biaya ketersediaan tinggi dan DR di platform alternatif | Replikasi, klaster, dan situs cadangan menambah biaya besar |
| Biaya keamanan dan audit | Enkripsi, pemantauan, dan bukti kepatuhan punya biaya di setiap platform |
| Biaya migrasi | Penulisan ulang aplikasi, pengujian, dan risiko selama transisi |
| Biaya operasional jangka panjang | Jumlah server, lisensi per core, dan tim yang mengelolanya |
| Biaya gangguan | Kerugian per jam layanan berhenti dikali perbedaan ketersediaan (Bab 46.1) |

Perbandingan yang jujur tidak selalu menghasilkan jawaban “tetap di mainframe”. Untuk beban tertentu, platform lain memang lebih cocok, dan arsitektur hybrid (Bab 56) memanfaatkan kekuatan masing-masing. Yang penting adalah keputusan diambil dari perhitungan yang lengkap, bukan dari satu angka yang paling mudah terlihat.

System engineer punya peran penting dalam perhitungan ini. Data SMF memberi jumlah transaksi dan pemakaian sumber daya yang akurat, dan pengetahuan teknis diperlukan untuk memastikan perbandingan memakai asumsi ketersediaan dan keamanan yang setara.

## Bab 2. Evolusi Platform

IBM Z adalah keturunan langsung System/360 dan mempertahankan kompatibilitas program sejak 1964. Program yang dikompilasi puluhan tahun lalu umumnya masih berjalan tanpa kompilasi ulang.



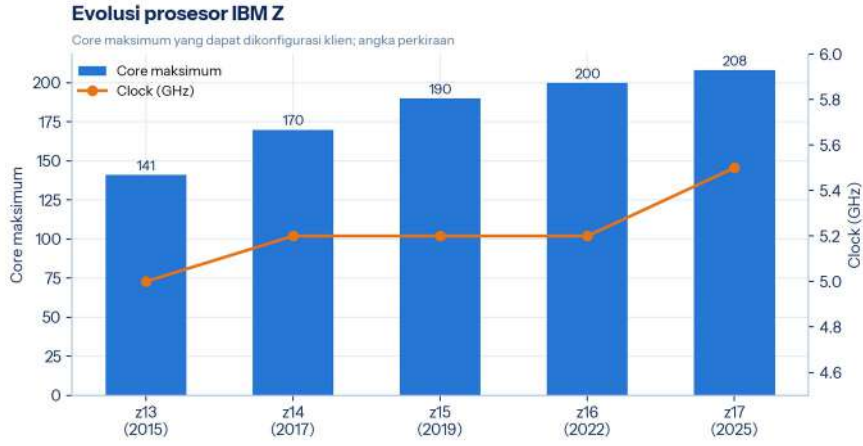

*Gambar 2.1 — Evolusi dari System/360 sampai IBM z17*

| Tahun | Hardware | Sistem operasi | Catatan penting |
|---|---|---|---|
| 2025 | IBM z17 | z/OS 3.2 | Prosesor Telum II, akselerator AI Spyre |
| 2023 | — | z/OS 3.1 | Penomoran versi 3, fokus AI dan cloud-native |
| 2022 | IBM z16 | z/OS 2.5 | Prosesor Telum dengan akselerator AI on-chip |
| 2019 | IBM z15 | z/OS 2.4 | Kompresi on-chip, Data Privacy Passports |
| 2017 | IBM z14 | z/OS 2.3 | Pervasive encryption |
| 2015 | IBM z13 | z/OS 2.2 | SMT untuk zIIP dan IFL |
| 2008 | System z10 | z/OS 1.10 | Quad-core, 4,4 GHz |
| 2000 | zSeries z900 | z/OS 1.1 (2001) | Arsitektur 64-bit |

| Tahun | Hardware | Sistem operasi | Catatan penting |
|---|---|---|---|
| | | | z/Architecture |
| 1990 | ES/9000 | MVS/ESA, OS/390 (1996) | ESCON, fondasi Parallel Sysplex |
| 1983 | System/370-XA | MVS/XA | Pengalamatan 31-bit |
| 1974 | System/370 | MVS | Multiple Virtual Storage |
| 1964 | System/360 | OS/360 | Awal keluarga arsitektur |

Warisan panjang ini punya dua sisi. Kompatibilitas melindungi investasi aplikasi, tetapi juga berarti Anda akan menemui kode COBOL, Assembler, dan JCL berusia 30 tahun di lingkungan produksi.

Jumlah core naik hampir 50% dari z13 ke z17, sementara clock hanya naik sekitar 10%. Peningkatan kinerja per generasi kini lebih banyak datang dari cache, akselerator on-chip, dan efisiensi instruksi.

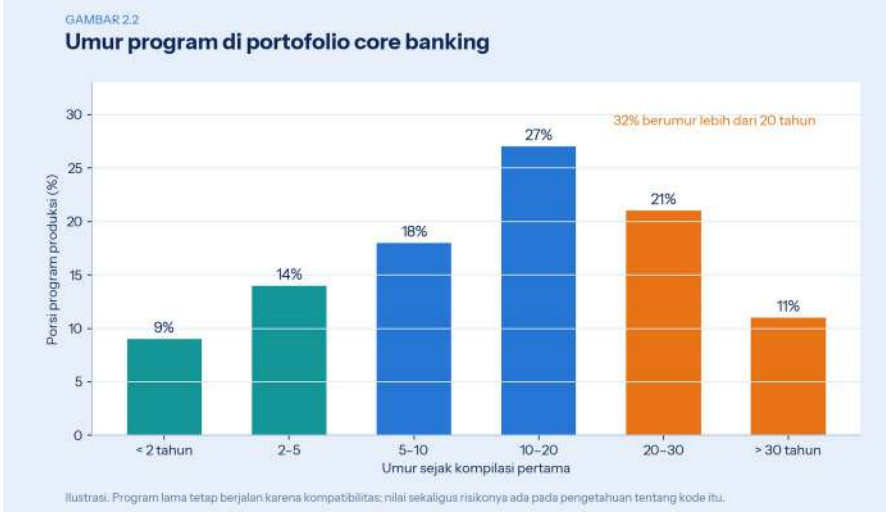



*Perkiraan dari dokumentasi IBM; verifikasi di Technical Guide per model*

## 2.1 Analisis: Umur Program dan Nilai Kompatibilitas

Kompatibilitas lintas generasi adalah alasan utama program berumur puluhan tahun masih berjalan di produksi. Bagi bank, ini berarti investasi

aplikasi yang dibuat puluhan tahun lalu masih menghasilkan nilai. Namun umur program juga membawa risiko yang perlu diukur.




*Gambar 2.2 — Umur program di portofolio core banking*

Dalam ilustrasi, sekitar sepertiga program berumur lebih dari 20 tahun. Program seperti ini biasanya stabil karena sudah teruji oleh waktu. Risikonya bukan pada kodenya, tetapi pada pengetahuan tentang kode itu: siapa yang masih memahaminya, dan apakah dokumentasinya masih cocok dengan isinya.

| Kelompok umur | Nilai | Risiko | Tindakan yang disarankan |
|---|---|---|---|
| Kurang dari 5 tahun | Masih dipahami pembuatnya | Belum teruji panjang | Pastikan uji otomatis dan dokumentasi ikut dibuat |
| 5–20 tahun | Stabil dan masih ada yang memahami | Pengetahuan mulai terkonsentrasi | Catat pemahaman kunci selagi orangnya ada |
| Lebih dari 20 tahun | Sangat stabil | Pengetahuan bisa hilang; dikompilasi dengan compiler lama | Inventaris, analisis dengan alat, rencana kompilasi ulang bertahap |

**Kompilasi ulang program lama** dengan compiler dan ARCH yang lebih baru dapat menurunkan CPU secara nyata (Bab 75.3), tetapi bukan tanpa risiko.

Compiler modern lebih ketat, dan program lama kadang bergantung pada perilaku yang tidak terdokumentasi. Strategi yang aman adalah kompilasi ulang bertahap: mulai dari program yang paling banyak memakan CPU, uji dengan data produksi yang disamarkan, lalu bandingkan hasil output secara byte per byte dengan versi lama.

Inventaris umur program bisa dibangun dari informasi di load library dan listing compile. Gabungkan dengan data pemakaian CPU dari SMF dan faktor bus tim (Bab 3.4). Hasilnya menunjukkan program mana yang paling penting, paling tua, dan paling sedikit dipahami. Kelompok itulah prioritas pertama untuk dokumentasi dan transfer pengetahuan.

# Bab 3. Peran dan Kompetensi System Engineer z/OS

Di dunia mainframe, peran “system programmer” (sysprog) setara dengan system engineer di buku IBM i. Sysprog bertanggung jawab atas instalasi, konfigurasi, pemeliharaan, dan ketersediaan sistem operasi beserta subsistemnya.

## 3.1 Batas tanggung jawab

| Aktivitas | Sysprog z/OS | CICS/Db2 SysAdmin | App Support | Operator | Security (RACF) |
|---|---|---|---|---|---|
| Instalasi PTF lewat SMP/E | R/A | C | I | I | I |
| Perubahan PARMLIB dan IPL | R/A | C | I | R | I |
| Tuning WLM | R/A | C | C | I | — |
| Membuat user ID | I | — | — | — | R/A |
| Analisis batch EOD gagal | C | C | R/A | R | — |
| Restore dataset | R/A | C | C | R | I |

| Aktivitas | Sysprog z/OS | CICS/Db2 SysAdmin | App Support | Operator | Security (RACF) |
|---|---|---|---|---|---|
| Analisis dump abend | R/A | R | C | I | — |
| Uji DR tahunan | R/A | R | C | R | C |

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

## 3.2 Kompetensi inti

1. Wajib dikuasai
   - JCL dan utilitas dasar (IEBGENER, IDCAMS, DFSORT)
   - Struktur PARMLIB, proses IPL, dan perintah console
   - SMP/E untuk instalasi dan pemeliharaan
   - SDSF, SYSLOG/OPERLOG, dan membaca abend code
   - Dasar RACF dan DFSMS
2. Sangat disarankan
   - REXX untuk otomasi
   - WLM, SMF, dan RMF untuk analisis kinerja
   - Parallel Sysplex dan Coupling Facility
   - Analisis dump dengan IPCS
3. Nilai tambah
   - z/OSMF, Zowe, Ansible for IBM Z
   - z/VM dan Linux on IBM Z
   - Dasar HLASM untuk membaca exit dan dump

## 3.3 Masalah regenerasi SDM

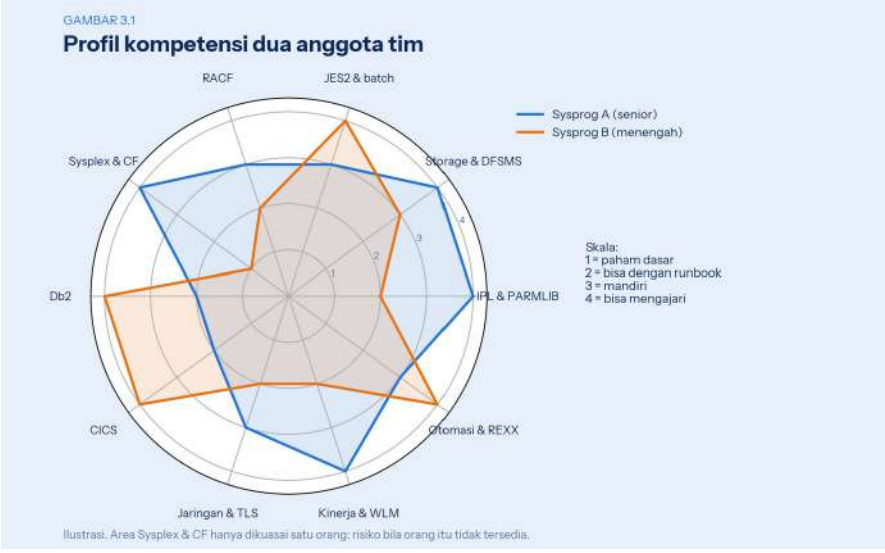
Masalah regenerasi di mainframe lebih akut daripada di IBM i. Banyak sysprog senior di Indonesia mendekati usia pensiun, sementara kampus hampir tidak mengajarkan z/OS.

- Tulis runbook untuk setiap prosedur yang hanya dipahami satu orang.

- Manfaatkan program IBM Z Xplore dan IBM Z Academic Initiative sebagai jalur masuk lulusan baru.
- Pasangkan junior dengan senior pada setiap IPL, instalasi RSU, dan uji DR.
- Gunakan z/OSMF dan VS Code dengan Zowe agar lulusan baru memakai alat yang sudah mereka kenal.

## 3.4 Analisis: Peta Kompetensi dan Faktor Bus Tim

Tim mainframe di banyak bank menghadapi risiko yang tidak terlihat di laporan teknis mana pun: pengetahuan penting yang hanya dimiliki satu atau dua orang, yang mendekati usia pensiun. Risiko ini dapat diukur dengan pemetaan kompetensi sederhana.




*Gambar 3.1 — Profil kompetensi dua anggota tim*

Setiap anggota tim menilai dirinya, lalu divalidasi oleh atasan atau rekan senior, pada skala 1 sampai 4 untuk setiap area teknis. Hasilnya digabung menjadi peta tim. Ukuran yang paling berguna dari peta ini adalah *faktor bus*: berapa orang di setiap area yang mampu bekerja mandiri.

$$\text{Faktor bus}_a = |\{\text{anggota tim dengan skor} \geq 3 \text{ di area } a\}|$$

Area dengan faktor bus 1 berarti satu orang yang cuti, sakit, atau pindah kerja membuat area itu tidak tertangani. Dalam ilustrasi, area Sysplex dan CF hanya dikuasai sysprog A. Jika sysprog A tidak tersedia saat terjadi masalah CF, tim harus menunggu atau meminta bantuan dari luar.

| Faktor bus | Status | Tindakan |
|---|---|---|
| 0 | Kritis | Pelatihan segera, atau dukungan eksternal yang disepakati |
| 1 | Berisiko | Pasangkan dengan anggota lain dalam pekerjaan nyata di area itu |
| 2 | Cukup | Pertahankan dengan rotasi tugas |
| 3 atau lebih | Kuat | Anggota senior dapat mulai mendalami area baru |

Cara paling efektif menaikkan faktor bus bukan kursus terpisah, tetapi pekerjaan nyata berpasangan. Anggota yang belajar ikut menangani perubahan, insiden, dan latihan di area itu bersama ahlinya (Bab 91.1). Setelah beberapa kali, perannya dibalik: yang belajar memimpin, yang ahli mengamati.

Perbarui peta kompetensi setiap tahun dan setiap kali ada anggota yang keluar atau masuk. Laporkan jumlah area dengan faktor bus 1 atau 0 kepada manajemen sebagai risiko operasional, bersama rencana untuk menurunkannya.

# 🏗️ Bagian II — Arsitektur Platform

## Bab 4. Anatomi Central Processor Complex

Satu mesin IBM Z disebut CPC (*Central Processor Complex*). Isinya adalah rak berisi CPC drawer, masing-masing memuat chip prosesor, memori, dan fanout untuk I/O.

IBM z17 memakai prosesor Telum II berclock 5,5 GHz dengan akselerator AI on-chip dan cache L2 besar yang juga membentuk L3/L4 virtual. Angka maksimum core dan memori berubah per model, jadi selalu rujuk *IBM z17 Technical Guide* (Redbooks) untuk konfigurasi Anda.

### 4.1 Karakterisasi Processing Unit

Setiap core fisik disebut PU dan dikarakterisasi untuk satu peran. Karakterisasi ini sangat menentukan biaya lisensi software.

| Jenis PU | Fungsi | Dihitung ke MSU software IBM? |
|---|---|---|
| CP (General Purpose) | Menjalankan z/OS, CICS, Db2, batch — semua beban umum | Ya |
| zIIP | Beban yang memenuhi syarat: Db2 DRDA, Java, XML, sebagian SRB, z/OS Connect | Tidak |
| IFL | Khusus Linux on IBM Z dan z/VM | Tidak (berlisensi per core) |
| ICF | Menjalankan Coupling Facility Control Code | Tidak |
| SAP | System Assist Processor, mengelola I/O | Tidak (tak terlihat oleh OS) |
| IFP | Integrated Firmware Processor untuk fungsi firmware PCIe | Tidak |
| Spare | Cadangan; mengambil alih PU yang gagal secara | Tidak |

| Jenis PU | Fungsi | Dihitung ke MSU software IBM? |
|---|---|---|
| | transparan | |

Pelajaran praktis: memindahkan beban dari CP ke zIIP adalah cara paling langsung menurunkan biaya MLC bulanan. Pantau metrik `zIIP-eligible on CP` di laporan RMF untuk melihat beban yang "tumpah" ke CP.

## 4.2 Kapasitas dan model

Kapasitas CP dinyatakan dalam kode model kapasitas, misalnya `7xx` untuk full capacity atau `4xx`, `5xx`, `6xx` untuk sub-capacity. Satuan untuk lisensi adalah MSU (*Millions of Service Units per hour*).

| Fitur Capacity on Demand | Fungsi | Kapan dipakai |
|---|---|---|
| CBU (Capacity Backup) | Mengaktifkan kapasitas tambahan di mesin DR | Uji DR dan bencana nyata |
| On/Off CoD | Menyewa kapasitas sementara per hari | Puncak akhir bulan atau akhir tahun |
| System Recovery Boost | Mempercepat IPL dan shutdown sementara | Otomatis saat IPL, bisa juga untuk recovery tertentu |
| Tailored Fit Pricing | Model harga berbasis konsumsi | Negosiasi kontrak dengan IBM |

## 4.3 Analisis: Merancang Campuran Processing Unit

Keputusan karakterisasi PU adalah keputusan biaya yang dibungkus sebagai keputusan teknis. Satu CP yang seharusnya zIIP menambah tagihan software setiap bulan selama umur mesin.


*Gambar 4.1 — Anatomi CPC dan karakterisasi PU*

| Pola beban | PU yang tepat | Alasan | Risiko bila salah pilih |
|---|---|---|---|
| COBOL batch dan CICS klasik | CP | Hanya CP yang menjalankan beban umum | Tidak ada alternatif |
| Java, Liberty, z/OS Connect | zIIP | Sebagian besar eligible | Beban berjalan di CP dan menaikkan MSU |
| Akses Db2 lewat DDF (JDBC/ODBC) | zIIP | Porsi signifikan eligible | Sama seperti di atas |
| Guest Linux di z/VM | IFL | Tidak dihitung MSU z/OS | Menaruh Linux di CP sangat mahal |
| Coupling Facility produksi | ICF dedicated | Latensi CF kritikal untuk data sharing | ICF shared memperlambat seluruh sysplex |
| I/O sangat tinggi | SAP tambahan | SAP memproses permintaan I/O | Antrean I/O di SAP meski disk cepat |

Pada generasi z13 ke atas, jumlah zIIP boleh hingga dua kali jumlah CP. Batas ini jarang tercapai, tetapi perlu diingat saat merencanakan pertumbuhan beban Java dan API.

**Porsi kapasitas LPAR saat terjadi kontensi.** Bobot PR/SM hanya berlaku ketika CP bersama diperebutkan. Porsi yang dijamin untuk LPAR ke-i adalah:

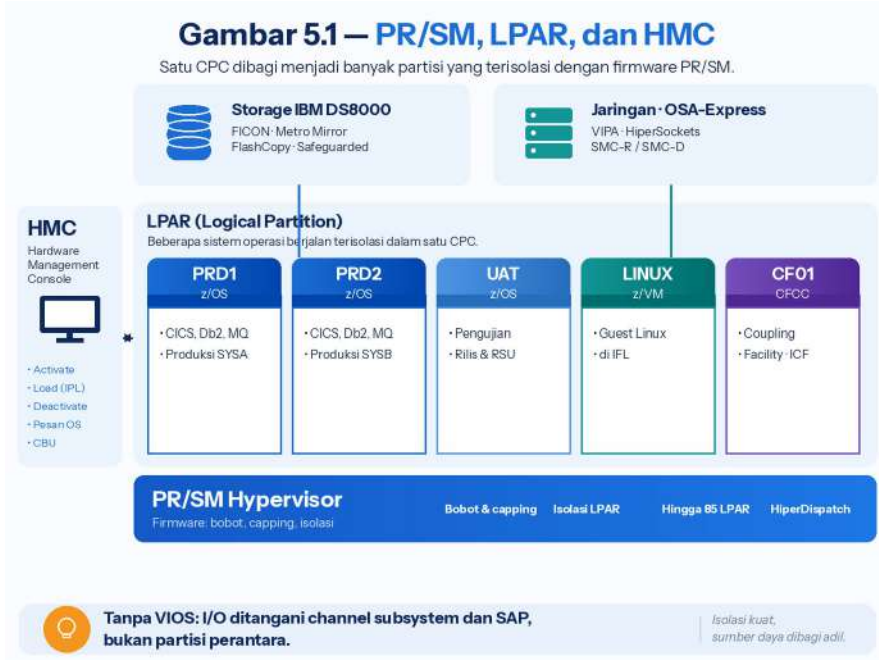
$$C_i = \frac{w_i}{\sum_j w_j} \times C_{\text{shared}}$$

Contoh: 10 CP bersama dibagi tiga LPAR berbobot 600 (produksi), 300 (UAT), dan 100 (development). Saat semua LPAR sibuk, produksi dijamin 6 CP, UAT 3 CP, dan development 1 CP. Jika produksi didefinisikan dengan 10 logical CP, empat di antaranya hanya mendapat sebagian waktu CP fisik. Kondisi ini disebut *short engine*: menambah latensi dan overhead tanpa menambah kapasitas.

| Gejala di RMF | Kemungkinan penyebab | Tindakan |
|---|---|---|
| Logical CP jauh lebih banyak daripada porsi bobot | Short engine | Samakan jumlah logical CP dengan porsi bobot plus sedikit cadangan |
| LPAR produksi tercekik saat UAT sibuk | Bobot UAT terlalu besar | Turunkan bobot non-produksi |
| Capping aktif di jam EOD | Defined capacity terlalu rendah | Naikkan batas atau geser beban (Bab 42) |
| “zIIP-eligible on CP” tinggi | zIIP kurang atau sibuk | Tambah zIIP atau kurangi beban eligible di jam puncak |

## Bab 5. PR/SM, LPAR, HMC, dan SE

PR/SM adalah hypervisor firmware yang membagi satu CPC menjadi beberapa LPAR (*Logical Partition*). Mesin generasi z15 ke atas mendukung hingga 85 LPAR.




*Gambar 5.1 — PR/SM, LPAR, dan HMC*

## 5.1 Parameter LPAR yang paling berpengaruh

| Parameter | Arti | Kesalahan umum |
|---|---|---|
| Weight | Porsi relatif CP bersama saat terjadi kontensi | Bobot produksi disamakan dengan bobot development |
| Logical CP | Jumlah CP logis yang dilihat z/OS | Terlalu banyak logical CP menambah overhead (short CP) |
| Initial capping | Hard cap: LPAR tidak bisa melebihi bobotnya | Dipasang di produksi lalu lupa dicabut |
| Defined capacity | Batas rolling 4-hour average (4HRA) dalam MSU | Terlalu rendah sehingga batch EOD tercekik |
| Group capacity | Batas 4HRA untuk sekelompok LPAR | Tidak dipantau saat LPAR baru ditambahkan |
| Storage | Memori sentral dan reserved | Reserved tidak dikonfigurasi, sehingga tak bisa menambah memori tanpa aktivasi ulang |

HiperDispatch sebaiknya aktif (`HIPERDISPATCH=YES` di `IEAOPTxx`). Fitur ini membuat z/OS bekerja sama dengan PR/SM agar pekerjaan tetap dekat dengan cache prosesornya.

### 5.2 HMC dan Support Element

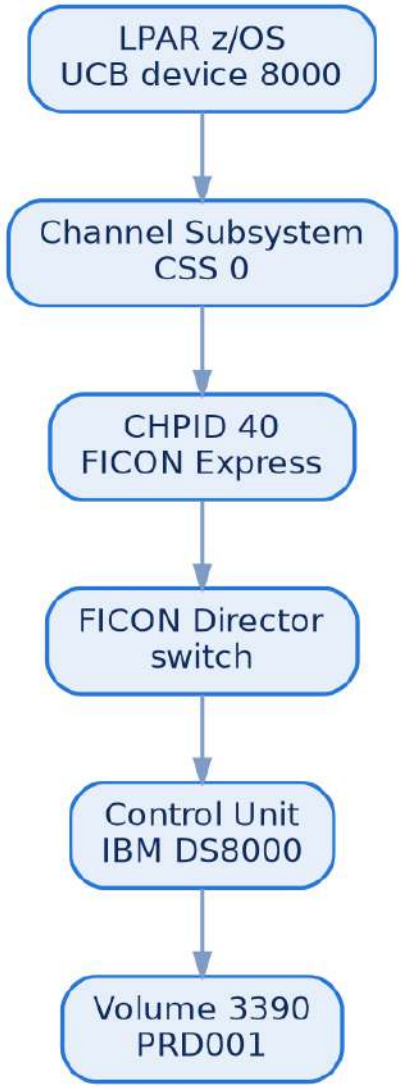
HMC (*Hardware Management Console*) adalah titik kendali seluruh CPC. Support Element (SE) adalah konsol yang melekat di setiap CPC.

| Tugas | Di HMC |
|---|---|
| Mengaktifkan CPC | *Activate* dengan Reset Profile |
| IPL z/OS | *Load* dengan Load Profile atau alamat IPL dan loadparm |
| Melihat pesan sebelum console z/OS siap | *Operating System Messages* |
| Console darurat | *Integrated 3270 Console* |
| Menghentikan LPAR | *Deactivate* (hanya setelah z/OS di-shutdown bersih) |
| Mengaktifkan CBU/On-Off CoD | *Perform Model Conversion* |

Hak akses HMC harus diperlakukan seperti akses root. Siapa pun yang bisa melakukan *Deactivate* bisa mematikan core banking dalam satu klik.

## Bab 6. Subsistem I/O

I/O di IBM Z tidak ditangani CPU umum, melainkan oleh channel subsystem dan SAP. Inilah alasan mainframe sanggup melayani ribuan I/O per detik tanpa membebani CP.




Setiap device z/OS dipetakan dari device number ke subchannel, CHPID, switch, control unit, hingga volume fisik. Pemetaan ini didefinisikan di IODF lewat HCD.

## 6.1 Istilah I/O yang wajib dipahami

| Istilah | Arti |
|---|---|
| CSS | Channel Subsystem; satu CPC bisa punya beberapa CSS |
| Subchannel set | Kumpulan subchannel; set tambahan dipakai untuk alias PAV dan volume sekunder |
| CHPID | Channel Path ID, nomor logis jalur I/O |
| PCHID | Physical Channel ID, lokasi fisik port adapter |
| Device number | Nomor 4 digit hex yang dipakai operator, misalnya `8000` |
| UCB | Unit Control Block, representasi device di memori z/OS |
| PAV / HyperPAV | Alias device agar satu volume bisa melayani banyak I/O paralel |
| zHyperLink | Koneksi latensi sangat rendah ke DS8000 untuk baca sinkron |

## 6.2 Perintah pemeriksaan I/O

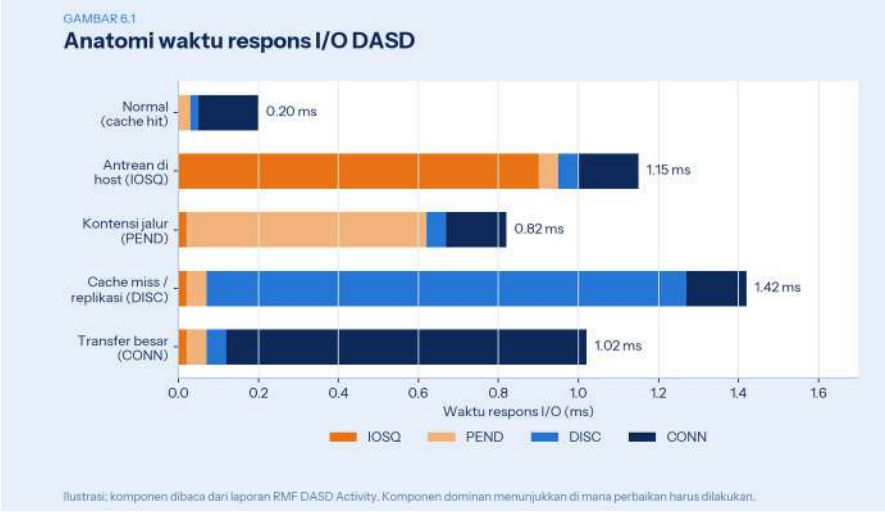
```
D M=CPU                  Status prosesor logis
D M=CHP(40)              Status CHPID 40
D M=DEV(8000)            Jalur ke device 8000
D U,DASD,ONLINE          Daftar DASD online
D IOS,CONFIG             IODF yang aktif
D IOS,HYPERPAV           Status HyperPAV
V PATH(8000,40),OFFLINE  Menonaktifkan satu jalur
```

## 6.3 Analisis: Membaca Waktu Respons I/O

Setiap I/O ke DASD menghabiskan waktu di beberapa tahap yang berbeda, dan RMF mencatat masing-masing tahap secara terpisah. Keluhan “disk lambat” hampir tidak pernah bisa diperbaiki tanpa tahu tahap mana yang memakan waktu, karena setiap tahap punya penyebab dan obat yang berbeda.

$$RT_{\text{I/O}} = T_{\text{IOSQ}} + T_{\text{PEND}} + T_{\text{DISC}} + T_{\text{CONN}}$$

**IOSQ** adalah waktu menunggu di dalam z/OS sebelum I/O dikirim ke device, biasanya karena device sedang melayani I/O lain dan tidak ada alias PAV yang bebas. **PEND** adalah waktu antara I/O diterima channel subsystem sampai device mulai merespons, dipengaruhi kesibukan jalur, switch FICON, dan control unit. **DISC** (*disconnect*) adalah waktu device bekerja tanpa memakai jalur, terutama saat data harus dibaca dari disk fisik karena tidak ada di cache, atau saat menunggu konfirmasi replikasi sinkron. **CONN** (*connect*) adalah waktu transfer data melalui jalur, sebanding dengan jumlah data.




*Gambar 6.1 — Anatomi waktu respons I/O DASD*

| Komponen dominan | Penyebab umum | Tindakan |
|---|---|---|
| IOSQ | Terlalu banyak I/O bersamaan ke satu volume; alias kurang | Aktifkan atau tambah HyperPAV; sebar dataset panas ke beberapa volume |
| PEND | Jalur FICON atau control unit sibuk | Tambah jalur; periksa pemakaian CHPID dan port switch |
| DISC | Cache miss, replikasi sinkron jarak jauh, array sibuk | Perbesar cache, periksa jarak dan kesehatan Metro Mirror, pertimbangkan zHyperLink untuk baca |
| CONN | Blok besar atau transfer sequential panjang | Sering normal untuk batch; periksa BLKSIZE dan pola |

| Komponen dominan | Penyebab umum | Tindakan |
|---|---|---|
| | | akses |

**Contoh analisis.** Volume `PRD101` yang berisi file master CIF melayani 3.000 I/O per detik dengan waktu respons 1,15 ms, dan 0,90 ms di antaranya IOSQ. Dengan hukum Little, rata-rata ada sekitar 3.000 × 0,00115 ≈ 3,5 I/O yang sedang berlangsung bersamaan. Jika volume hanya punya satu base device tanpa alias, I/O lainnya harus antre di host, dan itulah IOSQ yang terlihat. Mengaktifkan HyperPAV dengan beberapa alias biasanya menurunkan waktu respons mendekati 0,25 ms tanpa mengubah hardware.

Waktu respons I/O juga harus dilihat dari sisi transaksi. Transaksi yang melakukan 40 I/O sinkron dengan 0,3 ms masing-masing menghabiskan 12 ms hanya untuk I/O. Kenaikan menjadi 1,2 ms per I/O menambah 36 ms ke setiap transaksi, cukup untuk membuat anggaran latensi di Bab 35.2 terlampaui.

## Bab 7. Storage Hardware

DASD produksi di mainframe hampir selalu disk array IBM DS8000 yang mengemulasikan volume 3390. Tape umumnya berupa virtual tape IBM TS7700 yang di belakangnya berisi disk dan cartridge fisik.

| Fungsi copy DS8000 | Jenis | Kegunaan |
|---|---|---|
| FlashCopy | Point-in-time, lokal | Backup konsisten sebelum EOD, kloning data uji |
| Metro Mirror (PPRC) | Sinkron, jarak dekat | Dasar HyperSwap dan GDPS Metro |
| Global Mirror | Asinkron, jarak jauh | Replikasi ke DR di kota lain |
| Safeguarded Copy | Snapshot tak dapat diubah | Pemulihan dari ransomware atau korupsi logis |

Ukuran volume 3390 yang lazim adalah model 9 (sekitar 8,5 GB), model 27, model 54, dan EAV hingga lebih dari 1 TB. Standarkan ukuran volume per storage group agar manajemen kapasitas sederhana.

## 7.1 Analisis: Merencanakan Kapasitas Copy Services

Salinan data di storage terlihat gratis karena dibuat dalam hitungan detik, tetapi setiap salinan memakan kapasitas fisik. Tanpa perencanaan, array bisa penuh justru pada malam EOD ketika FlashCopy pra-EOD, snapshot Safeguarded Copy, dan pertumbuhan data terjadi bersamaan.

| Jenis salinan | Kapasitas yang dibutuhkan | Catatan |
|---|---|---|
| FlashCopy penuh | Sama dengan ukuran sumber | Salinan independen; cocok untuk data uji dan backup jangka panjang |
| FlashCopy space-efficient | Sebanding dengan data yang berubah setelah salinan dibuat | Hemat untuk salinan pra-EOD yang dibuang keesokan harinya |
| Safeguarded Copy | Sebanding dengan perubahan antar-snapshot dikali jumlah snapshot yang disimpan | Direncanakan dari laju perubahan dan retensi |
| Metro Mirror / Global Mirror | Kapasitas penuh di array sekunder | Plus kapasitas journal untuk Global Mirror |

Perkiraan kasar kapasitas Safeguarded Copy:

$$C_{\text{SGC}} \approx r_{\text{ubah harian}} \times C_{\text{produksi}} \times D_{\text{retensi}}$$

Contoh: 50 TB data produksi dengan laju perubahan 3% per hari, disimpan 7 hari, membutuhkan sekitar 50 × 0,03 × 7 = 10,5 TB. Angka ini batas atas yang aman, karena perubahan berulang pada track yang sama di antara dua snapshot hanya dihitung sekali. Laju perubahan nyata diukur dari statistik array selama beberapa pekan, termasuk akhir bulan ketika perubahan jauh lebih besar.

Pantau kapasitas array seperti storage group: tren bulanan, ambang peringatan, dan proyeksi. Kapasitas yang habis di array dapat membuat FlashCopy atau snapshot gagal diam-diam, dan kegagalan itu baru terlihat saat salinan dibutuhkan.

# Bab 8. Sistem Operasi di IBM Z

Satu CPC dapat menjalankan beberapa sistem operasi sekaligus di LPAR yang berbeda. Di perbankan, kombinasi yang paling umum adalah z/OS untuk core banking dan z/VM dengan Linux untuk beban pendukung.

| Sistem operasi | Kegunaan utama | Catatan |
|---|---|---|
| z/OS | Beban transaksi dan batch kritikal: CICS, IMS, Db2, MQ | Fokus buku ini |
| z/VM | Hypervisor untuk ratusan guest Linux atau z/OS uji | Berjalan di IFL atau CP |
| Linux on IBM Z | Aplikasi terbuka, container, OpenShift | Distribusi RHEL, SLES, Ubuntu |
| z/TPF | Transaksi ekstrem: maskapai, sebagian pemroses kartu | Sangat khusus |
| z/VSE (VSEn) | Beban warisan skala menengah | Pengembangannya telah dialihkan IBM ke 21st Century Software |
| KVM on IBM Z | Hypervisor Linux-native | Alternatif z/VM |

## 8.1 Analisis: Menempatkan Beban di Sistem Operasi yang Tepat

IBM Z bisa menjalankan z/OS, Linux, dan z/VM berdampingan. Pertanyaan yang sering muncul dalam proyek baru adalah di mana sebuah komponen sebaiknya ditempatkan. Jawabannya ditentukan oleh sifat beban, bukan oleh kebiasaan tim.

| Jenis beban | Tempat yang cocok | Alasan |
|---|---|---|
| Transaksi core banking COBOL/CICS | z/OS | Integritas transaksi, data sharing, dan alat operasional yang matang |
| Batch EOD dengan dataset VSAM/GDG | z/OS | JES2, scheduler, dan DFSMS dirancang untuk pola ini |
| API gateway, microservice Java/Node.js | Linux on Z atau z/OS Connect | Linux untuk ekosistem open source; z/OS Connect bila dekat dengan CICS |

| Jenis beban | Tempat yang cocok | Alasan |
|---|---|---|
| Database open source, middleware Linux | Linux on Z di IFL | Tidak menambah MSU z/OS; dekat dengan data inti |
| Banyak server kecil dengan pemakaian rendah | Linux di bawah z/VM | Overcommit dan konsolidasi (Bab 162) |
| Analitik berat di atas data inti | Linux on Z atau platform terpisah | Tergantung kebutuhan latensi dan biaya lisensi |

Tiga pertanyaan membantu memutuskan. Pertama, seberapa dekat beban ini harus dengan data core banking? Komunikasi lewat HiperSockets di CPC yang sama jauh lebih cepat daripada lewat jaringan pusat data. Kedua, model lisensi apa yang berlaku? Beban di IFL tidak menambah MSU z/OS, tetapi software Linux punya lisensinya sendiri. Ketiga, siapa yang akan mengoperasikannya? Beban Linux di mainframe membutuhkan tim yang paham Linux dan paham mainframe sekaligus.

Keputusan penempatan sebaiknya dicatat bersama alasannya. Beberapa tahun kemudian, ketika biaya atau teknologi berubah, catatan itu memungkinkan keputusan ditinjau ulang secara rasional.

## Bab 9. Komponen Inti z/OS

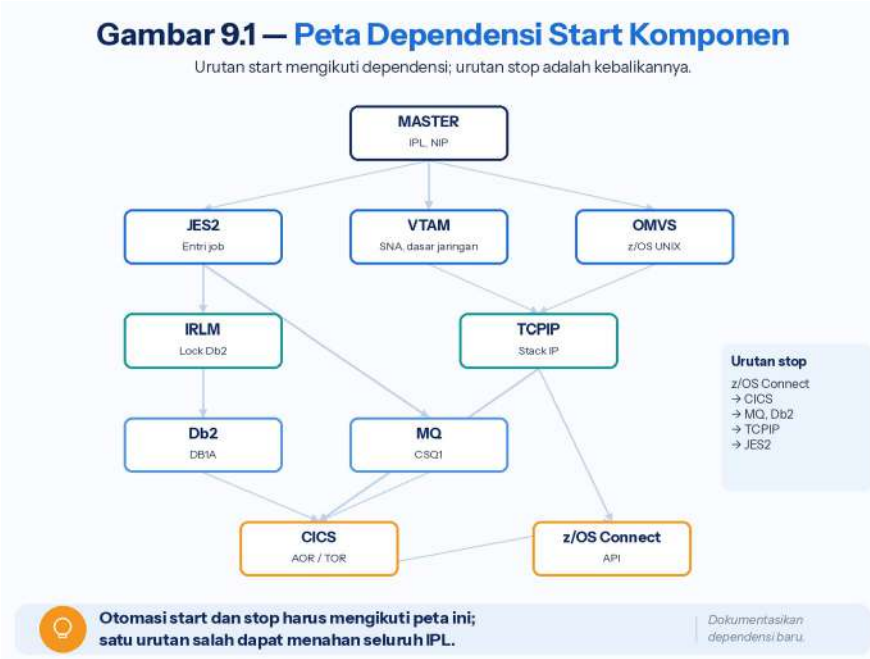
z/OS bukan satu program tunggal, melainkan kumpulan elemen dan fitur yang dipasang dan dirawat bersama lewat SMP/E. Tabel berikut memetakan elemen yang akan Anda sentuh hampir setiap hari.

| Elemen | Fungsi | Padanan kasar di IBM i |
|---|---|---|
| BCP (Base Control Program) | Kernel: dispatcher, storage manager, I/O supervisor | SLIC dan XPF |
| JES2 | Menerima, menjadwalkan, dan menampung output job | Job queue dan output queue |
| DFSMS | Manajemen dataset, katalog, dan storage | Single-level storage dan ASP |
| TSO/E dan ISPF | Antarmuka interaktif untuk pengguna teknis | Sesi 5250 dan PDM |
| SDSF | Melihat job, output, SYSLOG, | WRKACTJOB, WRKSPLF |

| Elemen | Fungsi | Padanan kasar di IBM i |
|---|---|---|
| | dan perintah console | |
| RACF (Security Server) | Otentikasi dan otorisasi | Profil pengguna dan wewenang objek |
| z/OS UNIX System Services | Lingkungan POSIX dan zFS | IFS dan PASE |
| Communications Server | TCP/IP dan VTAM/SNA | Konfigurasi TCP/IP |
| SMP/E | Instalasi dan pemeliharaan software | Manajemen PTF |
| RMF | Pengukuran kinerja | Collection Services |
| ICSF | Layanan kriptografi | Digital Certificate Manager |
| z/OSMF | Antarmuka web administrasi | Navigator for i |

## 9.1 Analisis: Dependensi Start dan Stop Komponen

Komponen z/OS tidak berdiri sendiri. CICS membutuhkan Db2 dan MQ untuk melayani transaksi yang menyentuh database dan antrean. Db2 membutuhkan IRLM. TCP/IP membutuhkan z/OS UNIX. z/OS Connect membutuhkan CICS dan TCP/IP. Urutan start yang salah tidak selalu membuat sistem gagal, tetapi hampir selalu membuat layanan terlambat siap atau berjalan dalam keadaan setengah terhubung.




*Gambar 9.1 — Peta dependensi start komponen*

Kebanyakan started task berjalan di bawah JES2, sehingga JES2 harus aktif lebih dulu. Beberapa komponen sistem dapat dijalankan langsung di bawah master scheduler dengan SUB=MSTR, sehingga tidak bergantung pada JES2. Keputusan komponen mana yang dijalankan dengan SUB=MSTR memengaruhi seberapa cepat sistem bisa pulih jika JES2 bermasalah.

| Urutan yang salah | Gejala | Pencegahan |
|---|---|---|
| CICS start sebelum Db2 siap | Transaksi Db2 abend AEY9 | DB2CONN dengan STANDBYMODE(RECONNECT); otomasi menunggu Db2 aktif |
| z/OS Connect start sebelum CICS | API mengembalikan error sampai CICS siap | Health check sebelum DVIPA diaktifkan |
| Kanal dibuka sebelum validasi | Nasabah bertransaksi saat sistem belum lengkap | Buka DVIPA atau routing paling akhir |
| TCP/IP dihentikan sebelum CICS | Transaksi di tengah jalan terputus mendadak | Urutan stop terbalik dari start |
| JES2 dihentikan dengan job | Shutdown tertahan | $P I dan tunggu job selesai |

| Urutan yang salah | Gejala | Pencegahan |
|---|---|---|
| masih aktif | | sebelum $P JES2 |

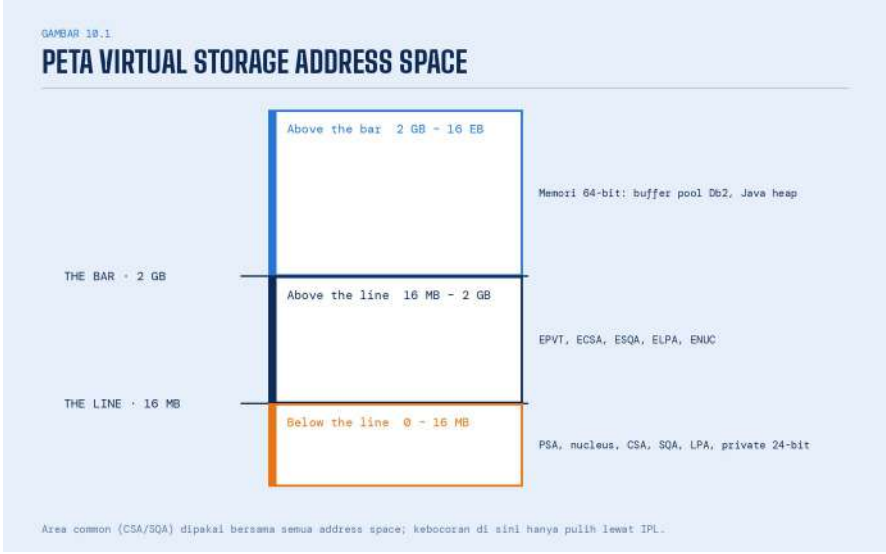
Peta dependensi ini harus hidup di otomasi, bukan hanya di dokumen. Produk otomasi seperti System Automation atau skrip yang dikelola rapi menyimpan relasi “A membutuhkan B” dan menjalankan start serta stop dengan urutan yang benar. Setiap kali komponen baru ditambahkan, misalnya produk monitoring atau gateway API baru, dependensinya harus didaftarkan saat itu juga. Dependensi yang tidak terdaftar adalah penyebab umum IPL yang “selesai” tetapi layanannya belum benar-benar siap.

# Bab 10. Address Space dan Virtual Storage

Setiap pekerjaan di z/OS berjalan dalam address space miliknya sendiri. Satu address space adalah ruang alamat virtual 64-bit yang terisolasi dari yang lain.

## 10.1 Peta virtual storage

| Area | Batas alamat | Isi |
|---|---|---|
| Above the bar | 2 GB sampai 16 EB | Memori 64-bit: buffer pool Db2, Java heap, memory object |
| The bar | 2 GB | Batas historis pengalamatan 31-bit |
| Above the line | 16 MB sampai 2 GB | Private 31-bit (EPVT), ECSA, ESQA, ELPA, ENUC |
| The line | 16 MB | Batas historis pengalamatan 24-bit |
| Below the line | 0 sampai 16 MB | PSA, nucleus, private 24-bit, CSA, SQA, LPA |




*Gambar 10.1 — Peta virtual storage address space z/OS*

Area bersama (*common*) seperti CSA dan SQA terlihat oleh semua address space. Kebocoran di area ini tidak hilang saat job selesai dan hanya pulih lewat IPL, sehingga harus dipantau ketat.

```
D VIRTSTOR,HVSHARE          Status shared memory 64-bit
D VIRTSTOR,LFAREA           Status large frame area
F HZSPROC,DISPLAY,CHECKS,CHECK=(IBMVSM,*)   Status health
check VSM
D A,L                       Daftar address space aktif
```

## 10.2 Address space sistem yang wajib dikenali

| Address space | Fungsi | Jika bermasalah |
|---|---|---|
| *MASTER* | Address space pertama, induk semua lainnya | Sistem tidak dapat berjalan |
| PCAUTH, RASP, TRACE, DUMPSRV | Layanan inti kernel, paging, trace, dump | Butuh IPL |
| XCFAS | Komunikasi antarsistem dalam sysplex | Sistem bisa dikeluarkan dari sysplex |
| GRS | Serialisasi sumber daya (ENQ) | Contention dan hang meluas |

| Address space | Fungsi | Jika bermasalah |
|---|---|---|
| CONSOLE | Manajemen pesan dan perintah | Operator kehilangan kendali |
| WLM | Workload Manager | Prioritas kerja kacau |
| SMSPDSE, SMSPDSE1 | Layanan PDSE | Akses load library gagal |
| CATALOG | Catalog Address Space | Dataset tidak dapat ditemukan |
| JES2 | Subsistem entri job | Tidak ada job baru dapat berjalan |
| OMVS | Kernel z/OS UNIX | TCP/IP dan aplikasi modern gagal |
| TCPIP, VTAM | Jaringan | Kanal transaksi terputus |

## 10.3 Unit kerja: TCB dan SRB

Pekerjaan dalam address space dijalankan sebagai TCB (*Task Control Block*) atau SRB (*Service Request Block*). TCB adalah tugas biasa yang bisa menunggu; SRB adalah rutinitas sistem berprioritas tinggi yang tidak bisa diinterupsi dengan cara yang sama.

Sebagian beban SRB dan TCB tertentu, misalnya permintaan Db2 lewat DRDA, dapat dijalankan di zIIP. Karena itu metrik “TCB time” dan “SRB time” di SMF tipe 30 penting saat menghitung biaya per aplikasi.

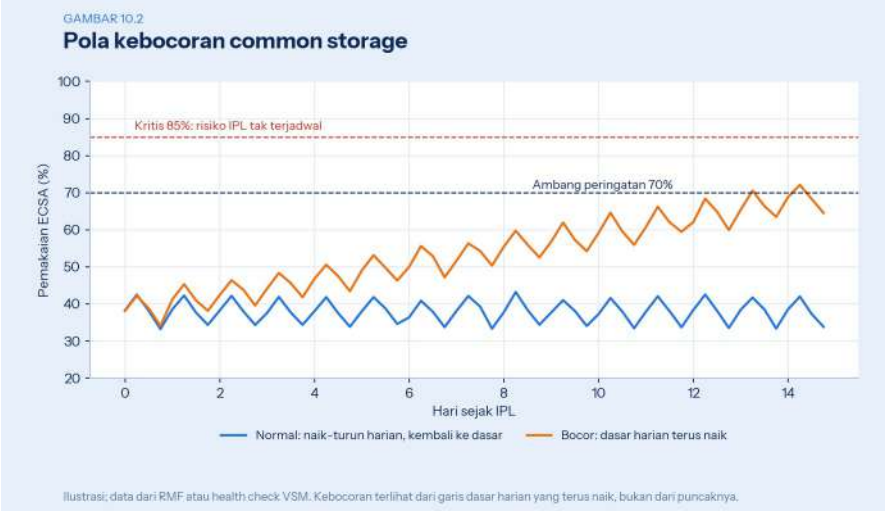
## 10.4 Analisis: Kehabisan Virtual Storage dan Kebocoran Common Area

Masalah virtual storage datang dalam dua bentuk yang sangat berbeda. Yang pertama terbatas pada satu address space: job atau region kehabisan memori privatnya dan abend. Yang kedua mengenai seluruh sistem: area common seperti CSA dan SQA terus terpakai tanpa dikembalikan, sampai sistem tidak bisa lagi memulai address space baru.

**Kehabisan private area.** Abend `S878` dan `S80A` menandakan permintaan memori yang tidak dapat dipenuhi. Penyebabnya bisa `REGION` yang terlalu kecil, batas yang ditetapkan exit `IEFUSI`, atau program yang memang

bocor memori. Untuk memori di atas 2 GB, batasnya ditentukan `MEMLIMIT` di JCL atau `SMFPRMxx`. Menaikkan batas hanya menunda masalah jika program terus meminta memori tanpa melepasnya. Bandingkan pemakaian memori di awal dan akhir job yang sama selama beberapa hari untuk membedakannya.

**Kebocoran common area.** Area common dipakai bersama oleh semua address space dan hanya dibersihkan saat IPL. Kebocoran terjadi ketika sebuah program atau subsistem mengambil CSA atau ECSA dan tidak mengembalikannya saat selesai. Pola khasnya terlihat jelas bila diplot per hari.




*Gambar 10.2 — Pola kebocoran common storage*

Pada sistem sehat, pemakaian common area naik saat jam sibuk dan turun kembali ke garis dasar yang sama setiap malam. Pada sistem yang bocor, garis dasarnya terus naik dari hari ke hari. Kapan batas akan tercapai dapat diperkirakan dari laju kenaikan garis dasar:

$$T_{\text{sisa}} = \frac{U_{\text{ambang}} - U_{\text{sekarang}}}{\Delta U_{\text{dasar per hari}}}$$

Contoh: garis dasar ECSA sekarang 64%, naik 2,1% per hari, dan ambang kritis 85%. Sisa waktu sekitar (85 − 64) / 2,1 = 10 hari. Dalam 10 hari itu

ada dua pilihan: menemukan pemilik kebocoran dan me-restart address space-nya, atau merencanakan IPL terkendali sebelum batas tercapai.

| Langkah | Alat |
|---|---|
| Deteksi dini | Health check `VSM_CSA_THRESHOLD` dan `VSM_SQA_THRESHOLD` |
| Mengaktifkan pelacakan pemilik | `VSM TRACK CSA(ON) SQA(ON)` di `DIAGxx` |
| Melihat pemakaian per job | Laporan common storage RMF Monitor III |
| Bukti untuk vendor atau IBM | SVC dump address space pemilik |
| Pemulihan | Restart address space pemilik; IPL terkendali bila tidak bisa |

SQA yang penuh akan meluber ke CSA, jadi pantau keduanya bersama. Satu address space yang me-restart dan membebaskan common storage-nya adalah bukti kuat bahwa pemiliknya sudah ditemukan.

# 🛠️ Bagian III — Perencanaan dan Implementasi

## Bab 11. Desain Topologi LPAR dan Sysplex

Topologi yang sehat memisahkan produksi, pengembangan, dan sandbox sysprog ke LPAR berbeda. Minimal dua LPAR produksi dalam Parallel Sysplex diperlukan jika Anda menargetkan layanan tanpa jeda saat IPL.

| LPAR | Sistem | Sysplex | Isi | Catatan |
|---|---|---|---|---|
| PRD1 | SYSA | PLEXP | CICS AOR/TOR, Db2 member DB1A, MQ | Produksi, data sharing |
| PRD2 | SYSB | PLEXP | CICS AOR/TOR, Db2 member DB1B, MQ | Produksi, data sharing |
| CF01, CF02 | CFCC | PLEXP | Structure lock, cache, list | Di ICF, sebaiknya di CPC berbeda |
| DEV1 | SYSD | PLEXD | Development dan SIT | Terpisah dari sysplex produksi |
| UAT1 | SYSU | PLEXD | UAT dengan data samaran | |
| SPG1 | SYST | Monoplex | Sandbox sysprog untuk uji RSU | Satu-satunya tempat uji IPL berisiko |
| DR1, DR2 | SYSA', SYSB' | PLEXP (DR) | Target GDPS | Kapasitas via CBU |

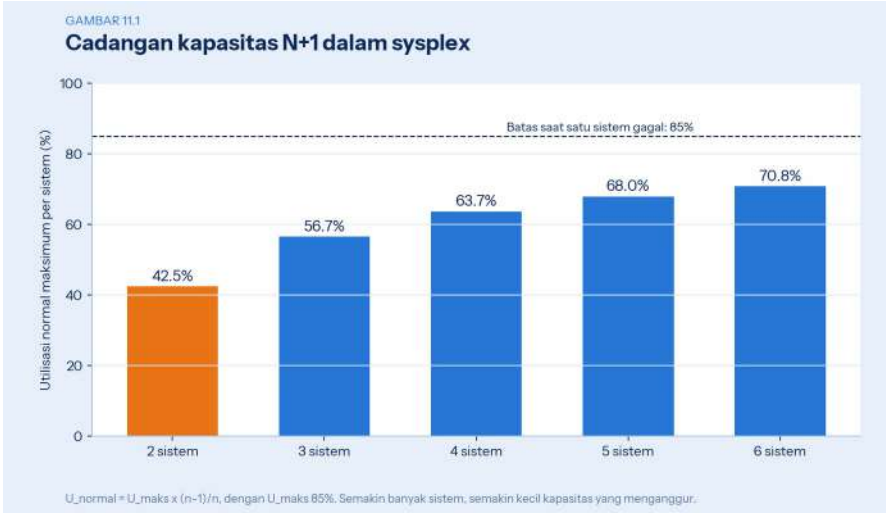
Prinsip yang saya pegang: setiap perubahan sistem harus lewat SPG1, lalu DEV, lalu UAT, lalu satu LPAR produksi, baru LPAR produksi kedua. Rolling IPL semacam ini adalah keunggulan utama sysplex.

## 11.1 Analisis: Berapa Sistem dalam Sysplex Produksi?

Pertanyaan “dua atau tiga sistem produksi?” sering dijawab dari kebiasaan, padahal jawabannya bisa dihitung. Tujuan sysplex adalah bertahan saat satu sistem berhenti, baik terencana maupun tidak. Syaratnya, sistem yang tersisa harus sanggup menampung seluruh beban.

Jika beban dibagi rata di n sistem dan setiap sistem tidak boleh melebihi utilisasi maksimum saat satu sistem hilang, maka utilisasi normal setiap sistem dibatasi:

$$U_{\text{normal}} \leq U_{\text{maks}} \times \frac{n-1}{n}$$




*Gambar 11.1 — Cadangan kapasitas N+1 dalam sysplex*

Dengan batas 85%, sysplex dua sistem hanya boleh berjalan di sekitar 42,5% per sistem pada kondisi normal. Separuh kapasitas menganggur sebagai cadangan. Dengan tiga sistem batasnya naik ke sekitar 56,7%, dan dengan empat sistem sekitar 63,7%.

| Jumlah sistem | Kelebihan | Kekurangan |
|---|---|---|
| 2 | Sederhana, overhead koordinasi kecil | Cadangan kapasitas besar; saat satu sistem di-IPL, tidak ada redundansi tersisa |

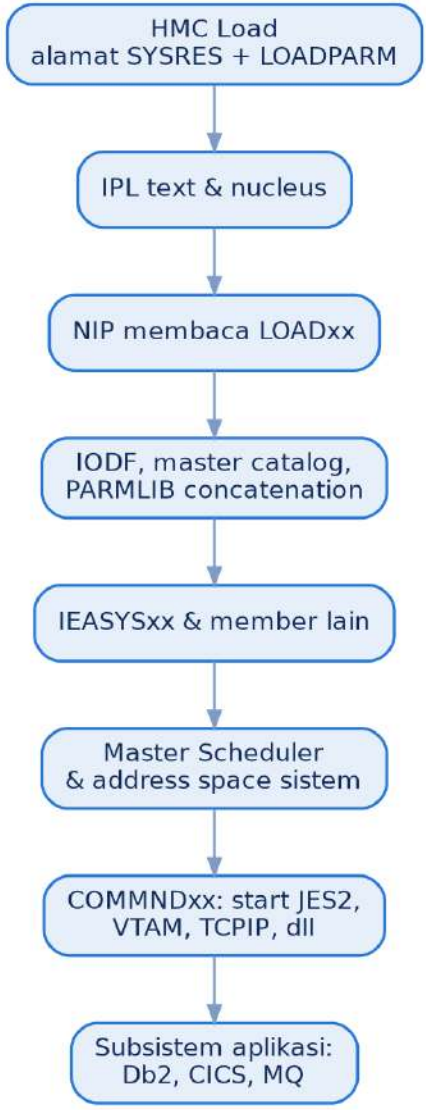
| Jumlah sistem | Kelebihan | Kekurangan |
|---|---|---|
| 3 | Rolling IPL tetap menyisakan dua sistem; cadangan lebih efisien | Lebih banyak LPAR, logical CP, dan konfigurasi yang dijaga |
| 4 atau lebih | Cadangan paling efisien | Kompleksitas operasional dan overhead data sharing meningkat |

Banyak bank memilih dua sistem dan menerima kondisi “terdegradasi” saat satu sistem berhenti: beban penting tetap berjalan, sedangkan beban discretionary ditunda oleh WLM. Pilihan ini sah, asalkan ditulis sebagai keputusan dan diuji. Yang berbahaya adalah menjalankan dua sistem masing-masing di 70% sambil berasumsi sysplex akan menyelamatkan layanan saat satu sistem gagal. Secara matematis, sistem yang tersisa akan diminta memikul 140%.

Ketersediaan sysplex juga dibatasi oleh komponen yang dipakai bersama. Dua sistem yang sama-sama bergantung pada satu CF, satu array storage, atau satu couple dataset tanpa alternatif akan gagal bersamaan. Karena itu rancangan sysplex dimulai dari daftar komponen bersama, lalu memastikan setiap komponen punya pasangan (Bab 43 dan 179).

## Bab 12. Proses IPL

IPL (*Initial Program Load*) adalah proses boot z/OS. Operator memulainya di HMC dengan dua nilai: alamat device IPL (volume SYSRES) dan LOADPARM delapan karakter.




NIP (*Nucleus Initialization Program*) membaca konfigurasi, lalu Master Scheduler menyalakan address space sistem. Otomasi seperti System Automation atau skrip COMMNDxx kemudian menyalakan subsistem sesuai urutan dependensi.

## 12.1 Struktur LOADPARM

| Posisi | Isi | Contoh |
|---|---|---|
| 1–4 | Alamat device volume berisi IODF | `9A10` |
| 5–6 | Suffix member LOADxx | `00` |
| 7 | IMSI: mode prompt saat IPL | `M` = tampilkan pesan, tanpa prompt |
| 8 | Suffix nucleus alternatif | `1` |

Contoh LOADPARM `9A1000M1` berarti IODF di `9A10`, pakai `LOAD00`, tampilkan pesan, nucleus `IEANUC01`.

## 12.2 Member PARMLIB penting

| Member | Mengatur | Bisa diubah dinamis? |
|---|---|---|
| LOADxx | IODF, master catalog, PARMLIB, sysplex, symbol | Tidak, butuh IPL |
| IEASYSxx | Parameter sistem utama, menunjuk member lain | Sebagian besar tidak |
| IEASYMxx | System symbol seperti `&SYSNAME`, `&SYSCLONE` | `SETLOAD xx,IEASYM` |
| PROGxx | APF list, LNKLST, LPA dinamis, exit | `SET PROG=xx` |
| LPALSTxx | Library LPA | Butuh IPL dengan CLPA |
| CONSOLxx | Definisi console | Sebagian lewat `VARY CN` |
| COMMNDxx | Perintah otomatis saat IPL | Tidak berlaku setelah IPL |
| SMFPRMxx | Tipe record SMF dan dataset SMF | `SET SMF=xx` |
| IGDSMSxx | Konfigurasi SMS | `SET SMS=xx` |
| COUPLExx | Couple dataset dan sysplex | `SETXCF` |
| IEAOPTxx | Parameter dispatcher dan HiperDispatch | `SET OPT=xx` |
| SCHEDxx | Program Properties Table | `SET SCH=xx` |

| Member | Mengatur | Bisa diubah dinamis? |
|---|---|---|
| BPXPRMxx | Parameter z/OS UNIX dan mount | `SET OMVS=xx` |
| MPFLSTxx | Penyaringan dan otomasi pesan | `SET MPF=xx` |
| IEFSSNxx | Definisi subsistem | `SETSSI` |

Simpan setiap perubahan PARMLIB di member baru dengan suffix berbeda. Jangan pernah menimpa member yang sedang aktif tanpa salinan kembali.

## 12.3 Jenis IPL dan start JES2

| Istilah | Arti | Kapan dipakai |
|---|---|---|
| IPL dengan CLPA | Membangun ulang LPA dari LPALST | Setelah RSU atau perubahan modul LPA |
| IPL tanpa CLPA | Memakai ulang PLPA dari page dataset | IPL rutin tanpa perubahan LPA |
| JES2 WARM start | Melanjutkan antrean job dan spool | Kondisi normal |
| JES2 HOT start | Melanjutkan setelah JES2 abend | Pemulihan JES2 |
| JES2 COLD start | Menghapus seluruh isi spool | Hanya jika benar-benar direncanakan; semua output hilang |

Peringatan keras: jawaban `COLD` pada prompt JES2 di produksi akan menghapus seluruh output batch yang belum diarsipkan. Wajibkan persetujuan dua orang untuk langkah ini.

## 12.4 Urutan shutdown yang aman

1. Hentikan akses kanal: tutup koneksi TCP/IP masuk atau alihkan ke sistem lain di sysplex.
2. Hentikan CICS dengan `F CICSxxx,CEMT PERFORM SHUTDOWN` atau prosedur standar.

3. Hentikan MQ, lalu Db2 dengan `-STOP DB2 MODE(QUIESCE)`.
4. Hentikan TCPIP dan VTAM, lalu inisiator batch dengan `$P I`.
5. Hentikan JES2 dengan `$P JES2`, lalu `Z EOD`.
6. Keluarkan sistem dari sysplex: `V XCF,SYSx,OFFLINE` dari sistem lain.
7. Baru lakukan *Deactivate* atau *Load* ulang di HMC.

## 12.5 Analisis Kegagalan IPL

IPL gagal hampir selalu di fase awal, sebelum console z/OS siap. Petunjuk satu-satunya ada di HMC: pesan sistem operasi dan wait state code. Karena itu prosedur IPL harus menyebut dengan jelas di mana petunjuk itu dibaca.


*Gambar 12.1 — Fase IPL dan titik kegagalannya*

| Fase | Gejala | Petunjuk | Penyebab umum | Tindakan pertama |
|---|---|---|---|---|
| Load | Tidak ada pesan sama sekali | HMC: status load gagal | Alamat SYSRES salah, volume offline | Periksa alamat device dan jalur ke volume |
| NIP | Sistem berhenti dengan wait | Wait state code dan reason code | LOADxx tidak ditemukan, | IPL ulang dengan |

| Fase | Gejala | Petunjuk | Penyebab umum | Tindakan pertama |
|---|---|---|---|---|
| | state | di HMC | IODF tidak cocok, master catalog tidak dapat dibuka | LOADPARM terakhir yang baik |
| PARMLIB | Pesan error sintaks, prompt operator | Operating System Messages | Member baru salah ketik, member tidak ada di konkatenasi | Pakai suffix member lama |
| Master scheduler | Address space sistem gagal start | SYSLOG awal | Perubahan LPA atau APF, dataset sistem hilang | IPL dengan CLPA dan konfigurasi lama |
| COMMNDxx | JES2 menunggu jawaban | WTOR JES2 | Operator tidak tahu harus menjawab apa | Jawaban tertulis di runbook: WARM, bukan COLD |
| Subsistem | Db2 atau CICS gagal start | Log subsistem | Urutan start salah, dependensi belum siap | Perbaiki urutan di otomasi |

Wait state code dibaca bersama reason code-nya. Misalnya wait state `064` menandakan kegagalan saat IPL dengan reason code yang menunjuk penyebab spesifik, dan `0A2` terkait sistem yang dikeluarkan dari sysplex. Rujuk *z/OS MVS System Codes* untuk arti reason code yang muncul.

## 12.6 Mengukur dan Memendekkan Durasi IPL

Durasi IPL menentukan berapa lama satu sistem sysplex tidak melayani saat rolling IPL, dan berapa lama seluruh layanan mati jika sysplex hanya satu sistem.

$$T_{\text{IPL}} = T_{\text{shutdown}} + T_{\text{load + NIP}} + T_{\text{master}} + T_{\text{subsistem}} + T_{\text{validasi}}$$

| Komponen | Cara memendekkan |
|---|---|
| Shutdown | Otomasi urutan stop; jangan menunggu |

| Komponen | Cara memendekkan |
|---|---|
| | operator mengetik perintah satu per satu |
| Load dan NIP | Gunakan System Recovery Boost yang tersedia di generasi z15 ke atas |
| Master scheduler | CLPA hanya bila LPA berubah (Bab 12.3) |
| Subsistem | Start paralel untuk subsistem yang tidak saling bergantung |
| Validasi | Transaksi uji otomatis, bukan pengecekan manual |

Catat durasi setiap komponen di setiap IPL. Data ini membuktikan manfaat perbaikan dan menunjukkan komponen mana yang memburuk dari waktu ke waktu.

# Bab 13. IODF dan HCD

Konfigurasi hardware yang dilihat z/OS dan PR/SM berasal dari IODF (*I/O Definition File*). IODF dibuat dan diubah lewat HCD, antarmuka ISPF khusus untuk konfigurasi I/O.

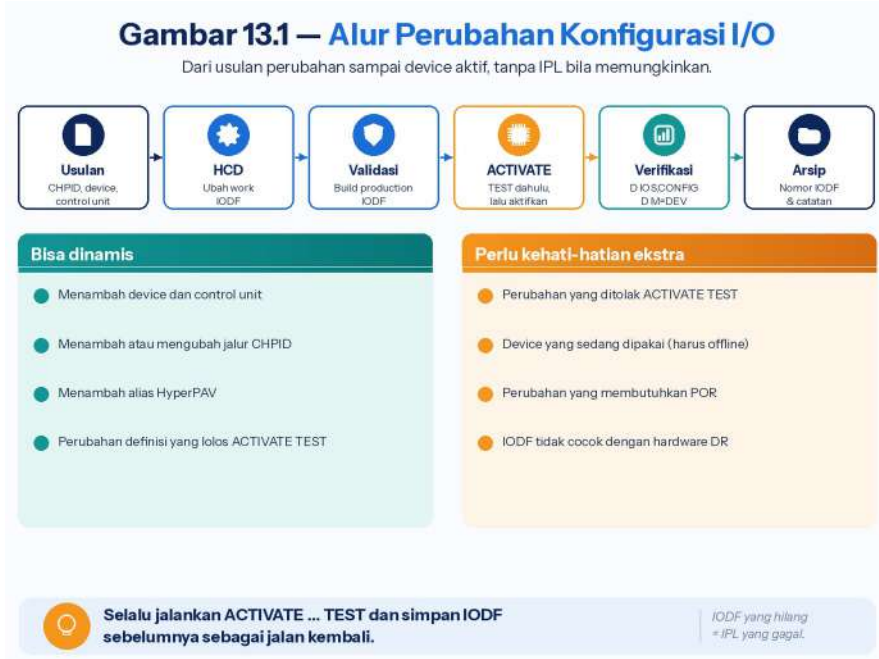
Perubahan I/O dapat diaktifkan tanpa IPL dengan *dynamic activate*:

```
ACTIVATE IODF=25,TEST        Uji dulu, tidak mengubah apa
pun
ACTIVATE IODF=25             Aktifkan IODF baru
D IOS,CONFIG                 Pastikan IODF aktif sudah
berganti
```

Simpan riwayat IODF dengan penomoran berurutan dan catat perubahan tiap nomor. Kehilangan IODF yang cocok dengan konfigurasi hardware bisa membuat IPL gagal total.

## 13.1 Analisis: Mengelola Perubahan I/O dengan Aman

Perubahan I/O terlihat kecil di atas kertas, misalnya menambah satu control unit atau dua jalur FICON. Namun IODF adalah fondasi setiap IPL. Kesalahan di sana bisa baru terasa berbulan-bulan kemudian, saat sistem di-IPL atau saat situs DR diaktifkan.




*Gambar 13.1 — Alur perubahan konfigurasi I/O*

Alurnya memisahkan tiga hal yang sering tercampur. **Work IODF** adalah tempat perubahan dirancang. **Production IODF** adalah hasil yang sudah divalidasi dan tidak dapat diubah lagi. **Aktivasi** adalah penerapan production IODF ke sistem yang berjalan. Memisahkan ketiganya membuat setiap versi konfigurasi dapat ditelusuri dan dikembalikan.

`ACTIVATE IODF=xx,TEST` adalah langkah terpenting. Perintah ini membandingkan konfigurasi aktif dengan yang baru dan melaporkan apakah perubahan dapat diterapkan dinamis, tanpa mengubah apa pun. Jika TEST menolak perubahan, jangan mencari jalan pintas. Penyebabnya harus dipahami: device yang masih online, definisi yang bertentangan, atau perubahan yang memang membutuhkan POR.

Dua hal sering terlupa setelah aktivasi dinamis berhasil. Pertama, konfigurasi yang sama harus ditulis ke IOCDS di Support Element, agar Power-on Reset berikutnya memakai definisi yang sama. Kedua, IODF untuk mesin di situs DR harus diperbarui secara paralel. Jika tidak, DR akan

berjalan dengan konfigurasi lama, dan masalahnya baru ditemukan saat uji DR atau, lebih buruk, saat bencana.

| Langkah | Bukti yang disimpan |
|---|---|
| Rancangan perubahan | Tiket, daftar CHPID/PCHID dan device yang berubah |
| Build production IODF | Nomor IODF dan laporan validasi HCD |
| `ACTIVATE ... TEST` | Hasil perintah, disimpan bersama tiket |
| Aktivasi | Waktu aktivasi dan hasil `D IOS,CONFIG` |
| IOCDS dan DR | Konfirmasi IOCDS diperbarui; IODF DR selaras |

# Bab 14. Instalasi dan Pemeliharaan dengan SMP/E

Semua software IBM di z/OS dipasang dan dirawat lewat SMP/E. SMP/E mencatat setiap modul, PTF, dan prasyarat di database bernama CSI.

## 14.1 Zona SMP/E

| Zona | Isi |
|---|---|
| Global zone | Daftar SYSMOD yang diterima dan HOLDDATA |
| Target zone | Kondisi library runtime (SYSRES, target library) |
| Distribution zone | Library distribusi, titik kembali untuk restore |

## 14.2 Siklus pemeliharaan

| Langkah | Perintah SMP/E | Tujuan |
|---|---|---|
| Terima PTF dan HOLDDATA | `RECEIVE ORDER` atau `RECEIVE` | Unduh dan catat ke global zone |
| Periksa dampak | `APPLY ... CHECK` | Simulasi tanpa mengubah library |
| Terapkan | `APPLY` | Tulis ke target library (clone SYSRES) |

| Langkah | Perintah SMP/E | Tujuan |
|---|---|---|
| Kukuhkan | `ACCEPT` | Tulis ke distribution library setelah stabil |
| Batalkan | `RESTORE` | Kembali ke kondisi sebelum APPLY yang belum di-ACCEPT |
| Cek PTF berbahaya | `REPORT ERRSYSMODS` | Temukan PTF terpasang yang kini ditandai PE |

Contoh JCL APPLY CHECK untuk satu level RSU:

```
//SMPCHK   JOB (ACCT),'APPLY CHECK',CLASS=A,MSGCLASS=X,
//            NOTIFY=&SYSUID
//SMPE     EXEC PGM=GIMSMP,REGION=0M
//SMPCSI   DD DISP=SHR,DSN=ZOS.GLOBAL.CSI
//SMPCNTL  DD *
  SET BOUNDARY(ZOSTGT).
  APPLY SOURCEID(RSU2509)
        GROUPEXTEND
        CHECK.
/*
```

## 14.3 Strategi pemeliharaan yang disarankan

- Ikuti level RSU (*Recommended Service Upgrade*) yang telah lolos pengujian CST IBM, bukan PTF acak.
- Terapkan ke SYSRES alternatif (clone), jangan ke SYSRES yang sedang dipakai.
- Gunakan FIXCAT untuk memastikan prasyarat hardware baru dan fungsi tertentu terpenuhi.
- Jalankan `REPORT ERRSYSMODS` setiap pekan setelah menerima HOLDDATA terbaru.
- Siklus realistis untuk bank: RSU dua kali setahun, ditambah HIPER dan PTF keamanan kapan pun diperlukan.

## 14.4 Analisis: Menyeimbangkan Risiko Perubahan dan Risiko Tertinggal

Pemeliharaan software selalu memilih di antara dua risiko. Memasang PTF membawa risiko perubahan: PTF baru bisa cacat. Tidak memasangnya membawa risiko tertinggal: sistem tetap rentan terhadap masalah yang sudah diketahui dan sudah ada perbaikannya.


*Gambar 14.1 — Siklus pemeliharaan SMP/E*

RSU menjadi titik tengah yang baik karena PTF di dalamnya sudah melewati pengujian terintegrasi IBM (CST). Siklus RSU dua kali setahun memberi ritme yang bisa direncanakan. Siklus itu bukan alasan untuk menunda perbaikan kritikal.

| Jenis temuan | Keputusan yang disarankan | Alasan |
|---|---|---|
| PTF biasa | Ikut siklus RSU berikutnya | Risiko rendah, manfaat pengujian terintegrasi |
| HIPER yang relevan dengan konfigurasi Anda | Pasang di luar siklus setelah uji singkat | Dampaknya tinggi bila terjadi |

| Jenis temuan | Keputusan yang disarankan | Alasan |
|---|---|---|
| PTF keamanan | Pasang secepat prosedur darurat mengizinkan | Celah keamanan dieksploitasi aktif |
| PE pada PTF yang sudah terpasang | Evaluasi segera; pasang perbaikannya atau RESTORE | Masalah sudah ada di sistem Anda |
| FIXCAT untuk hardware baru | Wajib sebelum migrasi CPC | Tanpa itu, fungsi baru gagal atau IPL gagal |

Kunci mengelola kedua risiko adalah **tidak pernah memasang langsung ke sistem yang berjalan**. APPLY ke SYSRES clone, IPL di sandbox, lalu rolling IPL ke sistem lain. Selama SYSRES lama masih ada dan PTF belum di-ACCEPT, jalan kembali selalu tersedia: cukup IPL dari SYSRES sebelumnya.

Ukur kesehatan pemeliharaan dengan dua angka sederhana: umur RSU yang terpasang di produksi, dan jumlah temuan REPORT ERRSYSMODS yang belum ditangani. Umur RSU lebih dari 12 bulan atau temuan PE yang menumpuk adalah tanda bahwa risiko tertinggal sedang tumbuh diam-diam.

## Bab 15. Baseline dan Manajemen Konfigurasi

Sysplex yang sehat memakai satu set PARMLIB bersama dengan system symbol, bukan salinan per sistem. Perbedaan antarsistem dinyatakan lewat symbol seperti &SYSNAME dan &SYSCLONE.

- Simpan PARMLIB, PROCLIB, dan definisi subsistem di Git atau SCM lain; z/OSMF dan Zowe CLI mempermudah sinkronisasi.
- Dokumentasikan baseline: level z/OS, RSU, IODF, LOADPARM, dan versi setiap subsistem.
- Bandingkan PARMLIB aktif dengan baseline sebelum setiap IPL; D PARMLIB menunjukkan konkatenasi yang aktif.
- Catat setiap perubahan dinamis (SET, SETPROG) ke member PARMLIB agar tidak hilang saat IPL berikutnya.

## 15.1 Analisis: Mendeteksi Drift Konfigurasi

*Configuration drift* terjadi ketika dua sistem yang seharusnya identik perlahan berbeda. Tidak ada satu perubahan besar yang menyebabkannya. Penyebabnya adalah puluhan perubahan kecil yang diterapkan di satu tempat dan lupa di tempat lain: perintah SETPR0G yang tidak dicatat ke PARMLIB, structure CFRM baru yang hanya didefinisikan di situs utama, atau profil TCP/IP yang diubah langsung di satu sistem.


*Gambar 15.1 — Baseline dan deteksi drift*

Drift berbahaya karena tidak terlihat sampai saat terburuk: saat IPL, saat failover, atau saat uji DR. Narasi keempat di Bab 121 adalah contohnya. CFRM policy di situs DR tertinggal dua tahun, dan kesalahannya baru terlihat ketika CICS gagal membuka file VSAM di tengah uji.

Pencegahannya adalah memperlakukan konfigurasi seperti kode. Baseline disimpan di Git, setiap perubahan melewati review, dan kondisi aktif setiap sistem dibandingkan dengan baseline setiap hari.

```
ALGORITMA DeteksiDrift(sistem_list, baseline)
  untuk setiap sistem s:
```

```
      aktif <- kumpulkan(D PARMLIB, D SYMBOLS, D PROG,APF,
D PROG,EXIT,
                         policy CFRM/SFM/WLM aktif, profil
TCP/IP, level RSU)
      untuk setiap item i di aktif:
          jika i berbeda dari baseline[s][i] maka
              jika ada tiket perubahan yang disetujui maka
catat "perlu masuk Git"
              selain itu catat DRIFT, prioritas tinggi
bila menyangkut DR atau keamanan
  bandingkan pasangan sistem yang seharusnya identik (SYSA
vs SYSA', SYSB vs SYSB')
  kirim laporan harian; drift tanpa tiket diperlakukan
sebagai insiden
```

Perbandingan pasangan produksi dan DR adalah bagian terpenting. Situs DR jarang dipakai, jadi tidak ada yang akan melihat perbedaannya secara kebetulan. Hanya pemeriksaan otomatis yang bisa menemukannya tepat waktu.

# 💾 Bagian IV — Dataset dan Manajemen Storage

## Bab 16. Dataset

Data di z/OS disimpan sebagai dataset, bukan file dalam direktori bersarang. Nama dataset terdiri dari qualifier yang dipisah titik, maksimum 44 karakter, dengan tiap qualifier maksimum 8 karakter.

Contoh: `BANK.PROD.CIF.MASTER`. Qualifier pertama (HLQ) menentukan katalog mana yang dipakai dan profil RACF mana yang melindunginya.

### 16.1 Jenis dataset

| Organisasi | Singkatan | Kegunaan | Catatan |
|---|---|---|---|
| Sequential | PS | File transaksi, laporan, extract | Paling sederhana |
| Partitioned | PDS | Source, JCL, PARMLIB | Perlu compress (`IEBCOPY`) saat ruang habis |
| Partitioned Extended | PDSE | Load library modern, source | Tanpa compress, mendukung program objects |
| VSAM KSDS | KSDS | File master berkunci: CIF, rekening | Paling umum di aplikasi CICS lawas |
| VSAM ESDS | ESDS | Log berurutan | Tanpa kunci |
| VSAM RRDS | RRDS | Akses berdasarkan nomor record | Jarang |
| VSAM Linear | LDS | Container Db2, zFS | Dikelola subsistem |
| Generation Data Group | GDG | Versi harian file batch: `(0)`, `(+1)`, `(-1)` | Wajib untuk output EOD berulang |

## 16.2 Atribut record

| Atribut | Arti | Nilai umum |
|---|---|---|
| RECFM | Format record | FB (fixed blocked), VB (variable blocked), U (load module) |
| LRECL | Panjang record logis | 80 untuk source dan JCL |
| BLKSIZE | Ukuran blok fisik | Biarkan 0 agar sistem memilih ukuran optimal (half-track) |

## 16.3 Batas extent: sumber abend paling sering

Dataset non-VSAM biasa hanya boleh punya 16 extent per volume. Jika primary dan secondary allocation terlalu kecil, job akan abend saat extent habis.

| Abend | Makna | Tindakan cepat |
|---|---|---|
| SB37 | Tidak bisa menambah extent; volume penuh atau batas extent tercapai | Perbesar SPACE atau izinkan multivolume |
| SD37 | Tidak ada secondary allocation | Tambahkan nilai secondary |
| SE37 | Batas volume atau direktori PDS penuh | Compress PDS atau tambah volume |

Dataset extended format dan VSAM dapat memiliki hingga 123 extent per volume dan hingga 59 volume. Untuk file EOD besar, gunakan data class extended format.

## 16.4 Analisis: Standar Penamaan Dataset

Di z/OS, nama dataset bukan sekadar label. Nama menentukan katalog tempat dataset terdaftar, profil RACF yang melindunginya, konstruk SMS yang dipilih ACS routine, kebijakan migrasi dan backup HSM, bahkan laporan biaya per aplikasi. Mengubah standar penamaan di tengah jalan sangat mahal, jadi standar harus ditetapkan sejak awal dan ditegakkan.

Pola yang banyak dipakai untuk aplikasi bank:

```
BANK.<LINGKUNGAN>.<APLIKASI>.<JENIS>.<NAMA>
contoh:
BANK.PROD.CIF.MASTER.KSDS        master CIF produksi
BANK.PROD.DEP.EOD.ACCRUAL        output akrual deposito
(GDG)
BANK.UAT.CIF.MASTER.KSDS         salinan UAT dengan data
samaran
BANK.PROD.CIF.LOADLIB            load library aplikasi CIF
```

| Qualifier | Fungsi | Dipakai oleh |
|---|---|---|
| HLQ (`BANK`) | Pemilik data, alias katalog | Katalog, RACF |
| Lingkungan (`PROD`, `UAT`, `DEV`) | Memisahkan produksi dari non-produksi | RACF, ACS routine, backup |
| Aplikasi (`CIF`, `DEP`, `GL`) | Pemilik bisnis dan biaya | Laporan kapasitas, chargeback |
| Jenis (`MASTER`, `EOD`, `INTF`, `RPT`) | Pola pemakaian data | Management class, retensi |
| Nama | Identitas spesifik | Dokumentasi job |

Dengan pola seperti ini, satu baris ACS routine dapat mengatur seluruh dataset EOD produksi:

```
PROC MGMTCLAS
  FILTLIST PRODEOD INCLUDE(BANK.PROD.*.EOD.**)
  SELECT
    WHEN (&DSN = &PRODEOD) SET &MGMTCLAS = 'MCEOD35'
    OTHERWISE              SET &MGMTCLAS = 'MCSTD'
  END
END
```

Pola yang sama membuat profil RACF menjadi sederhana: `BANK.PROD.**` untuk semua data produksi, dengan akses berbeda per aplikasi lewat `BANK.PROD.CIF.**`, `BANK.PROD.GL.**`, dan seterusnya.

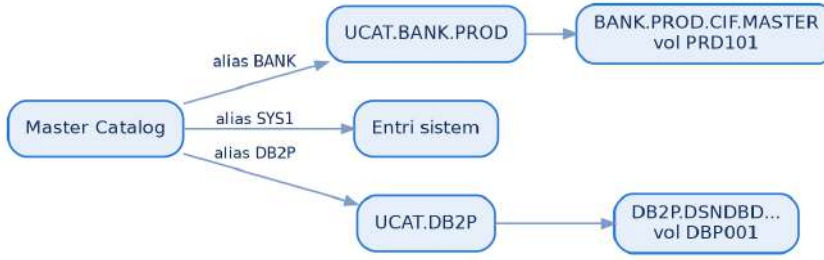
| Pola buruk | Mengapa berbahaya |
|---|---|
| HLQ pribadi di produksi (`USER01.CIF.MASTER`) | Hak akses mengikuti orang, bukan fungsi; hilang saat orangnya pindah |
| Lingkungan tidak ada di nama | Job UAT bisa menulis ke data produksi tanpa |

| Pola buruk | Mengapa berbahaya |
|---|---|
| | ditolak RACF |
| Dataset “sementara” yang hidup bertahun-tahun | Tidak masuk kebijakan backup, tetapi ternyata dipakai EOD |
| HLQ baru tanpa didaftarkan ke backup | Kasus Bab 120: enam bulan tanpa backup |

Standar penamaan paling efektif bila ditegakkan otomatis: ACS routine menolak alokasi dengan pola yang tidak dikenal di storage group produksi, dan pipeline deploy memeriksa nama dataset di JCL baru sebelum dipromosikan.

## Bab 17. Katalog

Katalog adalah indeks yang memetakan nama dataset ke volume tempatnya berada. Tanpa katalog, dataset hanya bisa ditemukan jika nama volumenya disebutkan.




Master catalog sebaiknya hanya berisi entri sistem dan alias. Seluruh dataset aplikasi harus berada di user catalog melalui alias HLQ.


*Gambar 17.1 — Hierarki HLQ dan katalog*

## 17.1 Perintah katalog

```
//LISTC    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  LISTCAT ENTRIES(BANK.PROD.CIF.MASTER) ALL
  LISTCAT ALIAS CATALOG(CATALOG.MASTER)
  DEFINE ALIAS (NAME(BANKX) RELATE(UCAT.BANK.PROD))
  EXAMINE NAME(UCAT.BANK.PROD) INDEXTEST DATATEST
/*
```

Perintah console untuk Catalog Address Space:

```
F CATALOG,LIST              Permintaan katalog yang sedang
berjalan
F CATALOG,OPEN              Katalog yang sedang terbuka
F CATALOG,CLOSE(UCAT.X)     Menutup katalog, misalnya
sebelum restore
F CATALOG,RESTART           Restart CAS jika hang
```

Backup katalog harian adalah wajib. Katalog rusak tanpa backup berarti ribuan dataset “hilang” meskipun datanya masih ada di disk.

## 17.2 Analisis: Menjaga Kesehatan Katalog

Katalog adalah salah satu komponen yang paling jarang dipikirkan, sampai bermasalah. Katalog yang rusak membuat dataset yang tercatat di dalamnya tidak dapat ditemukan, walaupun datanya masih utuh di disk. Jika katalog itu memuat dataset core banking produksi, dampaknya setara dengan kehilangan data sampai katalog dipulihkan.

| Aktivitas | Cara | Frekuensi |
|---|---|---|
| Pemeriksaan struktur | IDCAMS `EXAMINE` untuk struktur internal katalog | Mingguan atau sebelum perubahan besar |
| Pemeriksaan konsistensi | IDCAMS `DIAGNOSE` untuk kecocokan katalog dan isi volume | Bulanan |
| Backup | Salinan katalog dengan IDCAMS atau DFSMSdss | Harian, termasuk sebelum EOD |
| Pemantauan kinerja | `F CATALOG,REPORT,PERFORMANCE` | Saat ada keluhan dan dalam tinjauan bulanan |
| Rencana pemulihan | Backup terakhir ditambah perubahan sejak backup | Diuji setahun sekali |

Pemulihan katalog ke kondisi terkini membutuhkan dua bahan: backup katalog terakhir dan catatan perubahan katalog sejak backup itu. Perubahan katalog tercatat di record SMF tertentu, dan utilitas pemulihan katalog memakai record tersebut untuk menerapkan ulang perubahan di atas backup. Karena itu, retensi record SMF yang relevan harus cukup panjang untuk menjembatani jarak antar-backup katalog (Bab 170.1).

**Kontensi katalog** adalah masalah kinerja yang sering disalahpahami. Job yang lambat membuka dataset bisa jadi bukan karena disk, tetapi karena katalog sedang sibuk melayani permintaan lain, misalnya skrip yang menjalankan `LISTCAT` massal (kasus K-10). Laporan kinerja katalog menunjukkan jenis permintaan yang paling banyak memakan waktu.

Pisahkan data aplikasi ke beberapa user catalog berdasarkan HLQ (Bab 16.4). Satu katalog raksasa untuk semua data adalah titik kegagalan tunggal dan titik kontensi tunggal sekaligus. Beberapa katalog yang lebih

kecil membatasi dampak bila salah satunya bermasalah, dan mempercepat pemeriksaan serta pemulihannya.

# Bab 18. DFSMS dan Storage Group

DFSMS mengotomasi penempatan dan pengelolaan dataset. Pengguna tidak memilih volume; ACS routine yang memutuskan berdasarkan nama dataset, pemilik, dan jenis.

| Konstruk SMS | Menentukan | Contoh |
|---|---|---|
| Data Class | Atribut fisik: RECFM, LRECL, extended format | `DCEXTFB` |
| Storage Class | Kinerja dan ketersediaan | `SCPROD`, `SCDB2` |
| Management Class | Retensi, migrasi, backup oleh HSM | `MCEOD30` |
| Storage Group | Kumpulan volume tempat data ditaruh | `SGPROD`, `SGBATCH`, `SGDB2` |

## 18.1 Memantau storage group

```
D SMS,SG(SGPROD),LISTVOL    Pemakaian tiap volume
D SMS,STORGRP(ALL)          Ringkasan semua storage group
D SMS,VOL(PRD101)           Status satu volume
V SMS,VOL(PRD120,*),D,N     Nonaktifkan alokasi baru ke
volume
```

## 18.2 Ambang batas yang disarankan

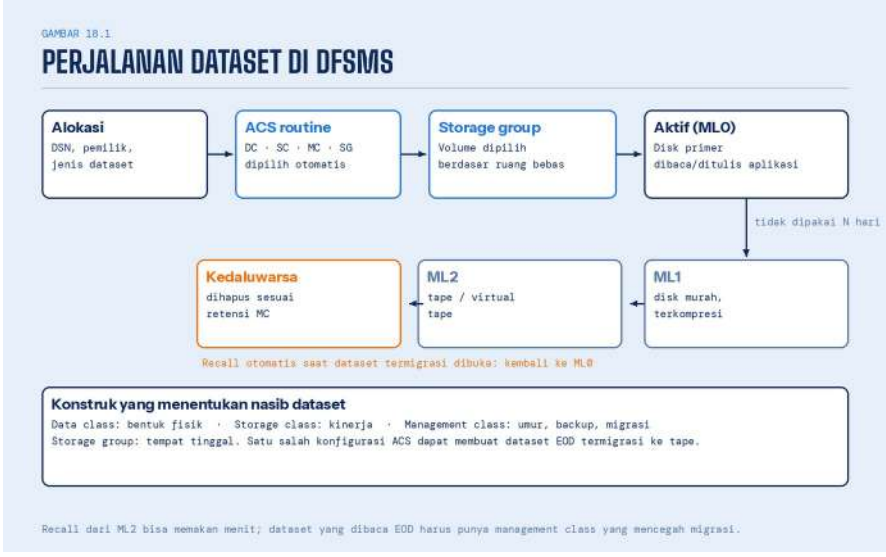
| Pemakaian storage group | Status | Tindakan |
|---|---|---|
| Di bawah 70% | Normal | Pantau tren bulanan |
| 70–80% | Perhatian | Rencanakan penambahan volume |
| 80–85% | Peringatan | Jalankan migrasi HSM, hapus GDG usang |
| 85–90% | Kritis | Tambah volume ke storage group segera |

| Pemakaian storage group | Status | Tindakan |
|---|---|---|
| Di atas 90% | Darurat | Risiko SB37 massal di batch EOD |

Seperti pada IBM i, lingkungan core banking sebaiknya memasang ambang peringatan di 75%. Output akhir bulan dan akhir tahun dapat menambah pemakaian belasan persen dalam satu malam.

## 18.3 Analisis: Menghitung Ruang dan Membaca Kesehatan Dataset

Abend SB37 di tengah EOD hampir selalu berarti SPACE yang ditebak, bukan dihitung. Perhitungan berikut cukup dilakukan sekali per dataset EOD penting dan dicatat di dokumentasi job.




*Gambar 18.1 — Perjalanan dataset di DFSMS*

Satu track 3390 menampung sekitar 56.664 byte, dan blok yang optimal adalah setengah track (hingga 27.998 byte). Dari situ jumlah record per track dapat dihitung:

$$\text{record per blok} = \left\lfloor \frac{27998}{LRECL} \right\rfloor \qquad \text{track} = \left\lceil \frac{N_{\text{record}}}{2 \times \text{record per blok}} \right\rceil \qquad \text{silinder} = \left\lceil \frac{\text{track}}{15} \right\rceil$$

| LRECL (FB) | BLKSIZE optimal | Record per track | Record per silinder |
|---|---|---|---|
| 80 | 27.920 | 698 | 10.470 |
| 133 | 27.930 | 420 | 6.300 |
| 400 | 27.600 | 138 | 2.070 |
| 1.000 | 27.000 | 54 | 810 |

Contoh: file transaksi harian 5 juta record dengan LRECL 400 membutuhkan sekitar 36.232 track atau 2.416 silinder. Dataset non-extended hanya boleh 16 extent per volume, sehingga kapasitas maksimumnya:

$$K_{\text{maks per volume}} = P + 15 \times S$$

Dengan `SPACE=(CYL,(1000,100))`, kapasitas maksimum di satu volume adalah 2.500 silinder: cukup untuk hari biasa, tetapi tidak untuk akhir bulan yang volumenya 30% lebih besar. Pilihan yang lebih aman adalah primary 2.500 silinder dengan secondary 500, atau data class extended format yang mengizinkan hingga 123 extent per volume.

**Kesehatan VSAM KSDS.** File master yang sering disisipi akan mengalami *CI split* dan *CA split*, terlihat di `LISTCAT` sebagai `SPLITS-CI` dan `SPLITS-CA`.

| Indikator LISTCAT | Arti | Tindakan |
|---|---|---|
| `SPLITS-CA` tumbuh cepat | Sisipan terkonsentrasi di rentang kunci tertentu | Tambah `FREESPACE` CA; reorganisasi berkala |
| `SPLITS-CI` tinggi, CA stabil | Sisipan tersebar | Tambah `FREESPACE` CI |
| `EXTENTS` mendekati batas | Pertumbuhan melebihi rencana | Reorganisasi ke alokasi baru yang lebih besar |
| `EXCPS` tinggi relatif terhadap record | Buffer kurang atau urutan akses acak | Tambah buffer (`BUFND`/`BUFNI`) atau pertimbangkan RLS/LSR |

**Biaya migrasi HSM.** Dataset yang dimigrasi ke ML2 membutuhkan recall dari tape sebelum dapat dibaca. Untuk dataset yang dibaca EOD, satu recall yang lambat dapat menunda seluruh rantai job. Karena itu management class dataset EOD harus menetapkan umur migrasi lebih

panjang dari siklus pemakaiannya, misalnya lebih dari 35 hari untuk data bulanan.

# ⚙️ Bagian V — Batch: JCL dan JES2

## Bab 19. Anatomi JCL

Setiap pekerjaan batch di z/OS dideskripsikan dengan JCL (*Job Control Language*). JCL hanya punya tiga statement utama: `JOB`, `EXEC`, dan `DD`.

| Statement | Fungsi | Parameter yang sering dipakai |
|---|---|---|
| `JOB` | Identitas job dan atribut umum | `CLASS`, `MSGCLASS`, `REGION`, `NOTIFY`, `RESTART`, `TYPRUN` |
| `EXEC` | Menjalankan program atau procedure | `PGM`, `PROC`, `PARM`, `COND`, `REGION` |
| `DD` | Menghubungkan nama file di program ke dataset | `DSN`, `DISP`, `SPACE`, `DCB`, `DATACLAS`, `SYSOUT` |
| `IF/THEN/ELSE/ENDIF` | Logika bersyarat antar step | `stepname.RC`, `ABEND`, `ABENDCC` |
| `SET`, `INCLUDE`, `JCLLIB` | Symbol dan pustaka procedure | Standarisasi JCL |

### 19.1 Contoh JCL tahap EOD

```
//EODACR01 JOB (BANK,EOD),'AKRUAL
BUNGA',CLASS=E,MSGCLASS=X,
//             MSGLEVEL=(1,1),REGION=0M
//*-------------------------------------------------------
----
//* STEP010 : SORT TRANSAKSI HARIAN BERDASARKAN NO
REKENING
//*-------------------------------------------------------
----
//STEP010  EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DISP=SHR,DSN=BANK.PROD.TRX.DAILY(0)
//SORTOUT  DD DSN=BANK.PROD.TRX.SORTED(+1),
//            DISP=(NEW,CATLG,DELETE),
//            DATACLAS=DCEXTFB,SPACE=(CYL,(500,100),RLSE)
//SYSIN    DD *
```

```
  SORT FIELDS=(1,16,CH,A)
/*
//*-------------------------------------------------------------
----
//* STEP020 : HITUNG AKRUAL, HANYA JIKA SORT BERHASIL
//*-------------------------------------------------------------
----
//CHK010   IF (STEP010.RC = 0) THEN
//STEP020  EXEC PGM=ACRBUNGA
//STEPLIB  DD DISP=SHR,DSN=BANK.PROD.LOADLIB
//TRXIN    DD DISP=SHR,DSN=BANK.PROD.TRX.SORTED(+1)
//ACCTMST  DD DISP=SHR,DSN=BANK.PROD.ACCT.MASTER
//RPTOUT   DD SYSOUT=R
//SYSOUT   DD SYSOUT=*
//CHK010E  ENDIF
```

## 19.2 Parameter DISP

| DISP | Status awal | Normal | Abnormal |
|---|---|---|---|
| SHR | Dataset ada, boleh dibaca bersama | — | — |
| OLD | Dataset ada, akses eksklusif | — | — |
| MOD | Tambah di akhir, atau buat jika belum ada | — | — |
| (NEW,CATLG,DELETE) | Buat baru | Katalogkan | Hapus |
| (OLD,DELETE,KEEP) | Ada | Hapus | Simpan untuk rerun |

Pola `(NEW,CATLG,DELETE)` membuat rerun lebih aman: jika step abend, dataset setengah jadi terhapus otomatis.

## 19.3 Return code dan abend

| Kode | Arti umum |
|---|---|
| RC 0 | Sukses |
| RC 4 | Peringatan, biasanya boleh lanjut |

| Kode | Arti umum |
|---|---|
| RC 8 | Error, output mungkin tidak lengkap |
| RC 12, 16 | Error serius |
| `Sxxx` | System abend, misalnya `S0C7` data numerik tidak valid |
| `Unnnn` | User abend yang dipanggil program atau utilitas |
| JCL ERROR | Job gagal sebelum dijalankan, misalnya dataset tidak ditemukan |

## 19.4 Analisis Kesalahan JCL yang Paling Sering

Sebagian besar kegagalan batch malam bukan abend program, melainkan JCL yang tidak cocok dengan kondisi dataset saat itu. Membaca JCL dengan pertanyaan yang tepat mencegah hampir semuanya.


*Gambar 19.1 — Anatomi JCL beranotasi*

| Pesan | Arti | Akar masalah tipikal | Pencegahan |
|---|---|---|---|
| `IEFC452I ... JOB NOT RUN - JCL ERROR` | JCL ditolak sebelum berjalan | Sintaks salah, PROC atau symbol tidak ada | Validasi JCL di pipeline sebelum promosi |

| Pesan | Arti | Akar masalah tipikal | Pencegahan |
|---|---|---|---|
| `IEF212I ... DATA SET NOT FOUND` | Dataset input tidak ada di katalog | Job pendahulu belum selesai atau gagal | Dependensi scheduler eksplisit |
| `IEF253I ... DUPLICATE NAME ON DIRECT ACCESS VOLUME` | Dataset baru bentrok dengan yang lama di volume | Rerun tanpa menghapus output lama | `DISP=(NEW,CATLG,DELETE)` dan step pembersih |
| `IEF287I ... NOT CATLGD 2` | Dataset dibuat tetapi gagal dikatalogkan | Nama sudah ada di katalog | Hapus atau pakai GDG untuk output berulang |
| `IEF453I ... JOB FAILED - JCL ERROR` | Kesalahan JCL terdeteksi saat alokasi | Parameter DD tidak cocok dengan dataset | Standarisasi PROC dan data class |

## 19.5 Analisis DISP untuk Keamanan Rerun

Parameter `DISP` menentukan apakah rerun aman. Tabel berikut menganalisis kombinasi yang paling sering dipakai.

| Pola DISP output | Aman di-rerun? | Alasan |
|---|---|---|
| `(NEW,CATLG,DELETE)` | Ya, setelah output lama dihapus | Abend menghapus output setengah jadi |
| `(NEW,CATLG,CATLG)` | Tidak | Output rusak tetap terkatalog; rerun gagal `NOT CATLGD 2` atau membaca data rusak |
| `MOD` | Tidak | Rerun menggandakan isi (lihat Praktikum 4) |
| GDG `(+1)` dengan `(NEW,CATLG,DELETE)` | Ya | Setiap run membuat generasi baru; rerun cukup dari step yang gagal |
| `OLD` pada master file | Hanya dengan backup | Isi sudah berubah sebagian saat abend |

## 19.6 Urutan Pencarian Program

Saat step menjalankan `PGM=`, z/OS mencari modul secara berurutan: job pack area, library task (`STEPLIB` atau `JOBLIB`), LPA, lalu LNKLST. Modul pertama yang ditemukan yang dipakai.


*Gambar 19.2 — Urutan pencarian program*

Karena urutan ini, `STEPLIB` yang menunjuk library lama akan memanggil versi program lama meskipun versi baru sudah ada di library lain. Untuk produksi, kendalikan library lewat PROC standar dan LNKLST, dan verifikasi dengan pesan alokasi di job log (RB-016, K-14).

# Bab 20. Utilitas Standar

| Program | Fungsi | Contoh kegunaan |
|---|---|---|
| `IEFBR14` | Tidak melakukan apa-apa; dipakai untuk alokasi atau hapus lewat DD | Hapus dataset sebelum rerun |
| `IEBGENER` | Menyalin dataset sequential | Salin file interface |

| Program | Fungsi | Contoh kegunaan |
|---|---|---|
| IEBCOPY | Salin, gabung, dan compress PDS/PDSE | Promosi load library |
| IDCAMS | Kelola VSAM, katalog, GDG | DEFINE CLUSTER, REPRO, LISTCAT |
| SORT (DFSORT) / ICETOOL | Sort, merge, filter, reformat, laporan | Hampir setiap job EOD |
| ADRDSSU | Backup dan restore dataset atau volume (DFSMSdss) | Backup sebelum EOD |
| IKJEFT01 | Menjalankan TSO dan REXX dalam batch | Otomasi, perintah Db2 DSN |
| BPXBATCH | Menjalankan perintah shell z/OS UNIX | Skrip transfer file |
| DSNUTILB | Utilitas Db2: REORG, RUNSTATS, COPY | Pemeliharaan database |
| AMASPZAP | Menambal modul secara biner | Hanya atas arahan IBM Support |

Contoh mendefinisikan GDG dengan 30 generasi dan VSAM KSDS:

```
//DEFINE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE GDG (NAME(BANK.PROD.TRX.SORTED) LIMIT(30) SCRATCH
NOEMPTY)
  DEFINE CLUSTER (NAME(BANK.PROD.CIF.MASTER) -
         INDEXED KEYS(12 0) RECORDSIZE(400 1200) -
         CYLINDERS(200 50) SHAREOPTIONS(2 3) -
         STORAGECLASS(SCPROD))
/*
```

## 20.1 Analisis: DFSORT dan ICETOOL sebagai Alat Operasional

DFSORT sering dianggap hanya pengurut data. Dalam praktik operasional, DFSORT dan ICETOOL adalah pisau lipat system engineer: memecah file, menghitung total kontrol, mencari duplikat, dan membuat laporan ringkas tanpa menulis program.

**Memecah file transaksi per wilayah dengan trailer jumlah record:**

```
//SPLIT    EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DISP=SHR,DSN=BANK.PROD.TRX.DAILY(0)
//JKT      DD
DSN=BANK.PROD.TRX.JKT(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(100,20),RLSE),DATACLAS=DCEXTFB
//SBY      DD
DSN=BANK.PROD.TRX.SBY(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(50,10),RLSE),DATACLAS=DCEXTFB
//LAIN     DD
DSN=BANK.PROD.TRX.LAIN(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(50,10),RLSE),DATACLAS=DCEXTFB
//SYSIN    DD *
  OPTION COPY
  OUTFIL FNAMES=JKT,INCLUDE=(17,3,CH,EQ,C'JKT'),
         TRAILER1=(1:C'JUMLAH RECORD:
',COUNT=(M11,LENGTH=10))
  OUTFIL FNAMES=SBY,INCLUDE=(17,3,CH,EQ,C'SBY'),
         TRAILER1=(1:C'JUMLAH RECORD:
',COUNT=(M11,LENGTH=10))
  OUTFIL FNAMES=LAIN,SAVE
/*
```

`SAVE` menampung semua record yang tidak cocok dengan filter lain. Dengan begitu tidak ada record yang hilang diam-diam, dan jumlah record di tiga file selalu sama dengan jumlah input.

**Total per rekening:** `SORT FIELDS=(1,16,CH,A)` diikuti `SUM FIELDS=(30,8,PD)` menghasilkan satu record per rekening dengan total nominal, berguna untuk rekonsiliasi dengan GL.

**Statistik dan pencarian duplikat dengan ICETOOL:**

```
//CEK      EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//IN       DD DISP=SHR,DSN=BANK.PROD.TRX.DAILY(0)
//DUP      DD
DSN=BANK.PROD.TRX.DUPREF(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(5,5),RLSE)
//TOOLIN   DD *
  COUNT  FROM(IN)
  STATS  FROM(IN) ON(30,8,PD)
```

```
  SELECT FROM(IN) TO(DUP) ON(40,20,CH) ALLDUPS
/*
```

`COUNT` menghitung record, `STATS` menampilkan nilai minimum, maksimum, rata-rata, dan total field nominal, sedangkan `SELECT ... ALLDUPS` mengambil semua record dengan referensi transaksi ganda. Langkah seperti ini cocok dijadikan step verifikasi di awal EOD: jika ada referensi ganda, EOD ditahan sebelum transaksi diposting dua kali.

Semua keluaran DFSORT dan ICETOOL masuk ke SYSOUT, sehingga ringkasannya bisa dibaca operator dan dibandingkan dengan total kontrol dari sistem pengirim (Bab 147).

# Bab 21. JES2

JES2 menerima job, menaruhnya dalam antrean berdasarkan class dan prioritas, menyerahkannya ke initiator, lalu menampung output di spool. Satu initiator menjalankan satu job pada satu waktu.

## 21.1 Perintah JES2 harian

| Perintah | Fungsi |
|---|---|
| `$D I` | Tampilkan semua initiator dan class-nya |
| `$T I5,C=EA` | Ubah initiator 5 agar melayani class E lalu A |
| `$S I5` / `$P I5` | Mulai atau hentikan initiator 5 |
| `$DA` | Job yang sedang aktif |
| `$D J1234` | Status job nomor 1234 |
| `$H J1234` / `$A J1234` | Tahan atau lepaskan job |
| `$C J1234` | Batalkan job (untuk yang sedang berjalan pakai `C jobname`) |
| `$D SPOOL` | Pemakaian spool |
| `$D JOBCLASS(E)` | Atribut class E |
| `$P JES2` | Hentikan JES2 setelah semua kerja selesai |

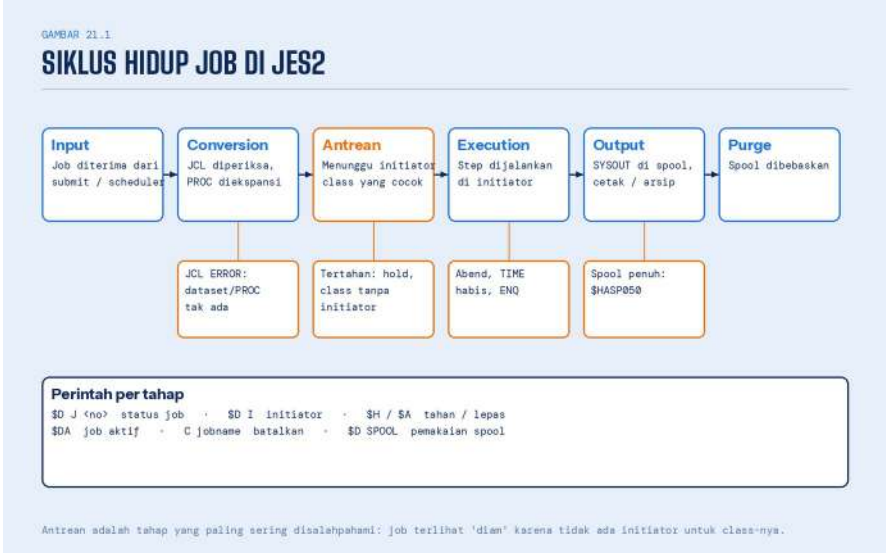
## 21.2 Ambang spool

Spool yang penuh menghentikan seluruh sistem batch dan bisa mengganggu subsistem online. Pesan `$HASP050` menandakan kekurangan sumber daya JES2.

| Pemakaian spool | Tindakan |
|---|---|
| Di bawah 60% | Normal |
| 60–75% | Periksa output yang tidak terarsip atau job dengan output raksasa |
| 75–85% | Purge output lama sesuai kebijakan, arsipkan ke sistem output management |
| Di atas 85% | Darurat: `$P` output besar yang sudah diarsip, tambah spool volume dengan `$S SP00L` |

## 21.3 Analisis Antrean Job dan Jumlah Initiator

Job yang “diam” di antrean adalah keluhan batch yang paling sering. Untuk menjawabnya dengan angka, gunakan hukum Little dari teori antrean.




*Gambar 21.1 — Siklus hidup job di JES2*

$$L = \lambda \times W \qquad \rho = \frac{\lambda \times S}{N}$$

L adalah rata-rata job aktif, λ laju kedatangan job per jam, W waktu rata-rata di sistem, S waktu eksekusi rata-rata, N jumlah initiator untuk class tersebut, dan ρ tingkat kesibukan initiator.

Contoh: class E menerima 120 job per jam di jendela EOD, masing-masing rata-rata berjalan 6 menit (0,1 jam). Rata-rata ada λ × S = 12 job yang berjalan bersamaan. Dengan 12 initiator, ρ = 1: antrean akan terus memanjang setiap ada sedikit variasi. Dengan 16 initiator, ρ = 0,75: antrean pendek dan stabil.

| ρ | Perilaku antrean |
|---|---|
| Di bawah 0,7 | Waktu tunggu kecil dan stabil |
| 0,7–0,85 | Waktu tunggu mulai terasa di puncak |
| Di atas 0,85 | Waktu tunggu naik tajam; sedikit lonjakan membuat antrean panjang |
| 1 atau lebih | Antrean tumbuh tanpa batas |

Menambah initiator hanya membantu jika sumber daya lain tersedia. Jika CPU atau I/O sudah jenuh, initiator tambahan hanya memindahkan antrean dari JES2 ke dispatcher. Cek PI service class batch (Bab 168) sebelum menambah initiator.

## Bab 22. Penjadwalan Batch dan Restart

Batch EOD bank biasanya diatur scheduler seperti IBM Z Workload Scheduler, Broadcom CA 7, atau BMC Control-M. Scheduler menyimpan dependensi, kalender, dan kondisi sukses setiap job.

Prinsip restart yang sama dengan buku IBM i berlaku di sini: setiap step harus aman diulang. Pada z/OS, restart dari step tertentu dilakukan dengan parameter `RESTART=stepname` di kartu JOB atau lewat fungsi rerun scheduler.

- Pastikan step yang mengubah master file selalu didahului backup (FlashCopy atau `ADRDSSU`).

- Gunakan GDG agar input dan output setiap run terpisah dan dapat dirujuk ulang.
- Hindari `DISP=MOD` pada file output yang dipakai step berikutnya; rerun akan menggandakan isinya.
- Catat status fase EOD di tabel kontrol Db2 atau dataset kontrol, bukan hanya di scheduler.

## 22.1 Analisis: Jalur Kritis Batch EOD

Durasi EOD bukan jumlah durasi semua job. Durasinya ditentukan oleh jalur terpanjang dalam grafik dependensi, yang disebut **jalur kritis**. Memahami jalur ini membedakan optimasi yang benar-benar memperpendek EOD dari optimasi yang hanya menghabiskan waktu engineer.


*Gambar 22.1 — Jalur kritis batch EOD*

$$T_{\text{EOD}} = \max_{p \in \text{jalur}} \sum_{j \in p} d_j \qquad \text{slack}_j = LS_j - ES_j$$

d adalah durasi job, ES waktu mulai paling awal, dan LS waktu mulai paling lambat tanpa menunda akhir EOD. Job dengan slack nol berada di jalur kritis.

Dalam contoh Gambar 22.1, EOD berlangsung 265 menit. Tim yang tidak memahami jalur kritis mungkin menghabiskan dua pekan mempercepat job Kliring dari 40 menjadi 20 menit, dan EOD tidak berubah sedetik pun karena jalur Kliring memiliki slack 35 menit. Sebaliknya, keputusan bisnis sederhana untuk menjalankan laporan setelah layanan dibuka memangkas 45 menit.

Langkah praktis membangun analisis jalur kritis:

1. Ekspor grafik dependensi dari scheduler, termasuk dependensi tersembunyi seperti dataset bersama.
2. Ambil durasi median setiap job selama 30 hari dari SMF tipe 30 atau riwayat scheduler.
3. Hitung jalur kritis dan slack setiap job; alat scheduler modern sering sudah menyediakannya.
4. Ulangi untuk durasi akhir bulan, karena jalur kritisnya bisa berbeda.
5. Fokuskan optimasi hanya pada job dengan slack nol, dan hitung ulang setelah setiap perubahan besar.

Jalur kritis juga berguna saat insiden. Jika job di luar jalur kritis gagal, ada waktu sebesar slack-nya untuk memperbaiki tanpa menunda layanan. Jika job di jalur kritis gagal, setiap menit keterlambatan langsung menjadi menit keterlambatan pembukaan layanan.

# 🖥️ Bagian VI — Antarmuka dan Otomasi

## Bab 23. TSO/E dan ISPF

TSO/E adalah lingkungan interaktif berbasis baris perintah, sedangkan ISPF adalah antarmuka menu layar penuh di atasnya. Hampir semua pekerjaan sysprog tradisional dilakukan dari ISPF lewat emulator 3270.

| Opsi ISPF | Fungsi | Padanan di IBM i |
|---|---|---|
| =2 | Edit dataset atau member | SEU / RDi |
| =3.4 | DSLIST: cari dan kelola dataset dengan pola nama | WRKOBJ, WRKLIB |
| =3.3 | Move/copy member atau dataset | CPYF, CRTDUPOBJ |
| =6 | Command shell TSO | Command line |
| =S atau =SD | SDSF (tergantung konfigurasi menu) | WRKACTJOB, WRKSPLF |
| =M.x | Menu produk: SMP/E, HCD, IPCS | Menu GO |

Perintah TSO yang sering dipakai sysprog:

```
LISTCAT ENT('BANK.PROD.CIF.MASTER') ALL   Info katalog
dataset
LISTDS 'SYS1.PARMLIB' MEMBERS             Daftar member
LU OPSUSR1                                Lihat profil
RACF user
LD DA('BANK.PROD.**') GENERIC             Lihat profil
dataset RACF
SEND 'EOD SELESAI' USER(OPSUSR1)          Kirim pesan ke
user
OMVS                                      Masuk ke shell
z/OS UNIX
```

## 23.1 Analisis: Produktivitas di ISPF

Sysprog berpengalaman bisa terlihat bekerja dua kali lebih cepat daripada rekan baru, padahal perbedaannya sering hanya pada penguasaan beberapa perintah ISPF. Bagian ini merangkum perintah yang paling menghemat waktu dalam pekerjaan sehari-hari.

| Kebutuhan | Perintah di ISPF Edit | Kegunaan |
|---|---|---|
| Melihat hanya baris yang relevan | `X ALL` lalu `F ALL 'teks'` | Menyembunyikan semua baris, lalu menampilkan yang cocok saja |
| Mengganti di seluruh member | `C ALL 'lama' 'baru'` | Mengubah semua kemunculan sekaligus |
| Membandingkan dua versi | `COMPARE nama-dataset` | Menandai baris yang berbeda antara member ini dan member lain |
| Mewarnai sintaks | `HILITE JCL` atau `HILITE REXX` | Kesalahan sintaks lebih mudah terlihat |
| Kembali tanpa menyimpan | `CAN` | Membatalkan semua perubahan sejak member dibuka |

Di daftar dataset (opsi 3.4), perintah baris seperti `B` (browse), `E` (edit), `M` (daftar member), `I` (informasi), dan `Z` (compress PDS) mempercepat navigasi. Filter dengan pola seperti `BANK.PROD.*.EOD.**` langsung menampilkan semua dataset EOD produksi.

**Edit macro** mengotomasi pemeriksaan yang berulang. Contoh berikut menampilkan hanya baris yang merujuk data produksi, berguna saat meninjau JCL yang akan dipromosikan:

```
/* REXX - PRODCHK: tampilkan hanya baris yang merujuk data
produksi */
ADDRESS ISREDIT
"MACRO"
"EXCLUDE ALL"
"FIND ALL 'BANK.PROD.'"
"FIND ALL 'DISP=OLD'"
"FIND ALL 'DISP=MOD'"
EXIT 0
```

Simpan macro di library yang ada di konkatenasi SYSEXEC atau SYSPROC, lalu ketik PRODCHK di baris perintah Edit. Dalam sekejap terlihat apakah JCL "UAT" ternyata masih menulis ke data produksi.

ISPF tetap relevan, tetapi tidak harus menjadi satu-satunya alat. Pengembang yang terbiasa dengan editor modern bisa memakai VS Code dengan Zowe Explorer (Bab 55) untuk menyunting member dan menjalankan job, sementara ISPF tetap tersedia untuk pekerjaan operasional yang cepat di console.

## Bab 24. SDSF

SDSF adalah jendela utama untuk melihat apa yang terjadi di sistem. Dari sini sysprog dan operator memantau job, membaca output, dan memberi perintah console.

| Panel SDSF | Isi |
|---|---|
| DA | Address space aktif beserta CPU, EXCP, dan real storage |
| ST | Status semua job di antrean |
| O / H | Output yang siap dicetak atau ditahan |
| LOG / OPERLOG | SYSLOG satu sistem atau log gabungan sysplex |
| SR | Pesan yang menunggu balasan operator (WTOR) |
| INIT | Initiator JES2 |
| SP | Volume spool |
| ENC | Enclave WLM, termasuk transaksi DDF dan zIIP |
| CK | Health Checker |
| / | Memberi perintah console dari baris perintah SDSF |

Akses SDSF dikendalikan lewat class RACF SDSF. Pastikan pengembang tidak punya akses perintah / di produksi.

## 24.1 Analisis: SDSF sebagai Pintu Kendali

SDSF sering dianggap alat melihat output saja, padahal lewat SDSF seseorang dapat membatalkan job, membaca output job orang lain, dan memberi perintah console. Dari sudut pandang keamanan, SDSF adalah salah satu pintu kendali terkuat di sistem, dan harus diatur seperti itu.

Hak akses SDSF dikendalikan lewat profil RACF di class SDSF. Profil `ISFCMD.*` mengatur panel mana yang boleh dibuka, sedangkan `ISFOPER.SYSTEM` mengatur hak memberi perintah console lewat `/`. Akses ke output job lain diatur lewat class `JESSPOOL`. Kombinasi ketiganya menentukan siapa bisa melihat apa dan melakukan apa.

| Peran | Panel yang wajar | Perintah / | Output job lain |
|---|---|---|---|
| Pengembang | ST, O, H untuk job sendiri | Tidak | Tidak |
| Application support | ST, DA, O, H, LOG | Terbatas, misalnya D saja | Job aplikasinya |
| Operator | Semua panel operasional | Ya, dibatasi profil OPERCMDS | Ya |
| Sysprog | Semua panel | Ya | Ya, diaudit |
| Auditor | LOG, OPERLOG | Tidak | Hanya baca bila diperlukan |

Perintah yang diberikan lewat SDSF tercatat di SYSLOG bersama user ID pemberinya. Gunakan catatan ini dalam review bulanan: siapa memberi perintah perubahan seperti C, FORCE, V, dan SET, dan apakah semuanya punya tiket.

**Alur kerja yang efisien.** Operator berpengalaman bekerja dengan urutan yang konsisten saat ada keluhan: SR untuk pesan yang menunggu jawaban, DA untuk address space yang memakan sumber daya, ST untuk job yang tertahan, lalu LOG atau OPERLOG untuk pesan beberapa menit terakhir. Urutan ini menjawab tiga pertanyaan pertama dalam hampir setiap insiden: ada yang menunggu, ada yang rakus, atau ada yang gagal?

# Bab 25. z/OS UNIX System Services

z/OS UNIX menyediakan lingkungan POSIX dengan filesystem hierarkis berbasis zFS. TCP/IP, z/OSMF, Java, WebSphere Liberty, dan z/OS Connect semuanya bergantung padanya.

| Kebutuhan | Tempat |
|---|---|
| Konfigurasi TCP/IP modern, sertifikat, skrip | zFS, misalnya `/etc`, `/u/<user>` |
| Aplikasi Java, Node.js, Python | zFS, biasanya `/usr/lpp` untuk produk IBM |
| File interface CSV dari/ke sistem lain | zFS atau dataset sequential, tergantung aplikasi |
| Program COBOL dan data master | Dataset MVS |


*Gambar 25.1 — Dataset MVS dan zFS*

Perintah pemantauan z/OS UNIX:

```
D OMVS,F                    Filesystem yang ter-mount
D OMVS,A=ALL                Proses UNIX aktif
D OMVS,O                    Opsi BPXPRMxx aktif
F ZFS,QUERY,ALL             Statistik zFS
```

```
zfsadm aggrinfo -aggregate OMVS.ZFS.USR   (shell) sisa
ruang agregat
```

zFS yang penuh adalah penyebab umum kegagalan aplikasi Java dan log TCP/IP. Atur `aggrgrow` agar agregat tumbuh otomatis dan pantau pemakaiannya seperti storage group.

## 25.2 Analisis: Kapasitas dan Kinerja zFS

zFS terasa seperti filesystem Unix biasa, sehingga pengelolaannya sering diserahkan tanpa aturan yang jelas. Padahal di bawahnya, setiap filesystem zFS adalah dataset VSAM linear yang disebut *aggregate*, lengkap dengan batas extent, storage group, dan backup seperti dataset lainnya.

| Aspek | Yang perlu diatur | Akibat bila diabaikan |
|---|---|---|
| Ukuran awal dan secondary | Secondary allocation yang cukup untuk pertumbuhan | Filesystem penuh walau storage group masih lega |
| Pertumbuhan otomatis | `aggrgrow` aktif untuk aggregate aplikasi | Aplikasi Java gagal menulis log saat puncak |
| Pemisahan filesystem | Log, aplikasi, dan data pertukaran di aggregate terpisah | Log yang meledak memenuhi filesystem aplikasi |
| Cache | Parameter `user_cache_size` dan `meta_cache_size` di `IOEFSPRM` | I/O zFS tinggi, aplikasi lambat |
| Backup | Aggregate ikut rencana backup DFSMS | File konfigurasi dan sertifikat hilang saat restore |
| Sysplex | Mount sysplex-aware untuk filesystem bersama | Akses dari sistem lain lambat atau gagal |

**Pola insiden paling umum** adalah filesystem log yang penuh karena rotasi log tidak dikonfigurasi. Satu aplikasi yang menulis log debug di produksi bisa menghabiskan beberapa gigabyte dalam semalam. Jika log berada di filesystem yang sama dengan aplikasi, aplikasi ikut gagal.

Aturan praktisnya sederhana. Setiap aplikasi punya aggregate log sendiri dengan rotasi yang terkonfigurasi. Pemakaian zFS dipantau dengan ambang yang sama seperti storage group (Bab 18.2). Aggregate yang penting dimasukkan ke laporan cakupan backup (Bab 120).

# Bab 26. REXX untuk System Engineer

REXX di z/OS memegang peran yang sama dengan CL di IBM i: bahasa otomasi harian sysprog. REXX bisa berjalan interaktif di TSO, dalam batch lewat `IKJEFT01`, atau di z/OS UNIX.

## 26.1 Memeriksa extent dataset kritikal

Skrip berikut memeriksa dataset yang mendekati batas 16 extent sebelum batch EOD dimulai.

```
/* REXX - CEKEXT: peringatan dataset mendekati batas
extent */
daftar = "'BANK.PROD.CIF.EXTRACT' 'BANK.PROD.GL.EXTRACT'"
/* nama lengkap; LISTDSI tidak menerima (0) */
do i = 1 to words(daftar)
  dsn = word(daftar, i)
  rc  = LISTDSI(dsn)
  if rc <> 0 then do
    say dsn 'LISTDSI gagal, reason' sysreason
    iterate
  end
  if sysextents >= 12 then
    say 'PERINGATAN:' dsn 'sudah' sysextents 'extent'
  else
    say 'OK        :' dsn sysextents 'extent,' sysused
sysunits 'terpakai'
end
exit 0
```

## 26.2 Menjalankan perintah console dari REXX

SDSF menyediakan antarmuka REXX sehingga skrip dapat membaca panel dan memberi perintah console.

```
/* REXX - CEKSPOOL: tampilkan pemakaian spool lewat SDSF
*/
rc = isfcalls('ON')
Address SDSF "ISFSLASH '$D SPOOL' (WAIT)"
do i = 1 to isfulog.0
  say isfulog.i
end
Address SDSF "ISFEXEC DA"
say 'Address space aktif:' jname.0
rc = isfcalls('OFF')
exit 0
```

## 26.3 Menjalankan REXX dalam batch

```
//CEKEXT   EXEC PGM=IKJEFT01
//SYSEXEC  DD DISP=SHR,DSN=BANK.SYSPROG.REXX
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  %CEKEXT
/*
```

Simpan skrip REXX di library bersama dengan kontrol versi, dan jadwalkan lewat scheduler batch. Gaya ini membangun pustaka utilitas yang bisa diwariskan ke generasi berikutnya.

## 26.4 Analisis: Pagar Pengaman untuk Otomasi

Otomasi mempercepat pekerjaan yang benar, tetapi juga mempercepat kesalahan. Skrip yang membatalkan job “looping” berdasarkan CPU bisa membatalkan job EOD yang memang berat. Skrip yang memurge output lama bisa menghapus bukti audit. Karena itu setiap otomasi perlu tingkat kewenangan yang jelas.

| Tingkat | Perilaku | Contoh yang cocok |
|---|---|---|
| 1. Mengamati | Mengumpulkan data dan melaporkan | Health check harian (Bab 86) |
| 2. Menyarankan | Mengusulkan tindakan, manusia yang menjalankan | Kandidat REORG, kandidat purge spool |
| 3. Bertindak dengan persetujuan | Menjalankan setelah dikonfirmasi operator | Menambah volume ke storage group |

| Tingkat | Perilaku | Contoh yang cocok |
|---|---|---|
| 4. Otonom | Bertindak sendiri dalam batas yang ketat | Membalas WTOR rutin yang terdaftar, restart subsistem lewat ARM |

Setiap otomasi di tingkat 3 dan 4 wajib punya pagar pengaman berikut:

- **Mode uji (*dry-run*)** yang menampilkan apa yang akan dilakukan tanpa melakukannya.
- **Batas laju**, misalnya tidak lebih dari tiga tindakan per jam, agar kesalahan logika tidak menjadi bencana beruntun.
- **Daftar pengecualian** untuk job dan address space kritikal yang tidak boleh disentuh.
- **Tombol mati** yang bisa menghentikan semua otomasi dari satu tempat.
- **Log setiap tindakan** dengan alasan, waktu, dan data yang dipakai untuk memutuskan.

Naikkan tingkat kewenangan secara bertahap. Otomasi baru dimulai di tingkat 1 atau 2 selama beberapa pekan. Setelah sarannya terbukti benar secara konsisten, barulah dinaikkan ke tingkat berikutnya.

# 🔐 Bagian VII — Keamanan

## Bab 27. Konsep RACF

RACF adalah pengelola keamanan bawaan z/OS yang melakukan otentikasi pengguna dan memutuskan setiap akses ke sumber daya. Beberapa bank memakai produk pihak ketiga seperti ACF2 atau Top Secret, tetapi prinsipnya serupa.

| Konsep RACF | Arti | Padanan di IBM i |
|---|---|---|
| User ID | Identitas pengguna, maksimum 8 karakter | User profile |
| Group | Kelompok pengguna dan pemilik profil | Group profile |
| Dataset profile | Melindungi dataset berdasarkan pola nama, misalnya `BANK.PROD.**` | Wewenang objek pada library/file |
| General resource profile | Melindungi sumber daya non-dataset lewat class | Authorization list, function usage |
| UACC | Akses default bagi semua orang | `*PUBLIC` |
| Access list | Daftar user/group dan level aksesnya | Private authority |

Level akses dari rendah ke tinggi adalah `NONE`, `EXECUTE`, `READ`, `UPDATE`, `CONTROL`, dan `ALTER`. Untuk dataset produksi, UACC seharusnya selalu `NONE`.

### 27.1 Atribut user berisiko tinggi

| Atribut | Kekuatan | Kebijakan yang disarankan |
|---|---|---|
| `SPECIAL` | Mengelola seluruh database RACF | Hanya tim security, maksimal 2–3 orang |
| `OPERATIONS` | Akses ke hampir semua dataset | Hanya user teknis storage yang terbatas |

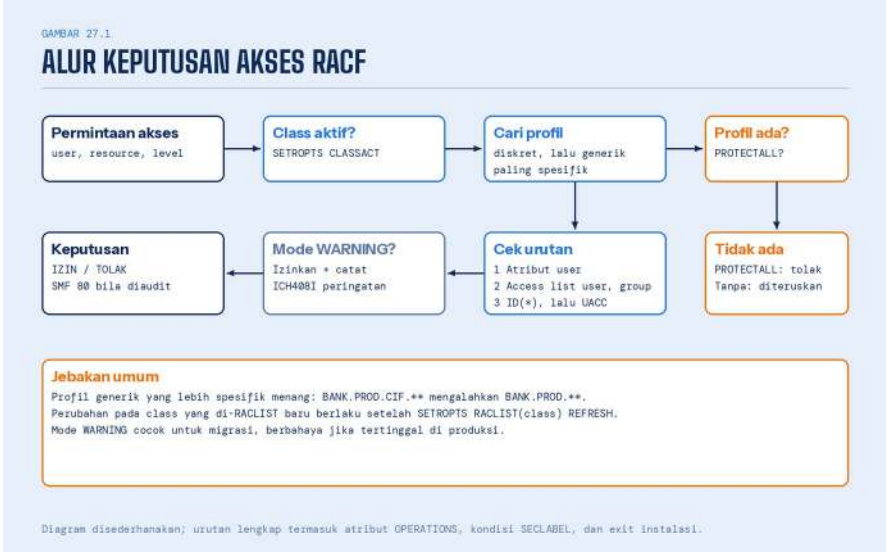
| Atribut | Kekuatan | Kebijakan yang disarankan |
|---|---|---|
| AUDITOR | Membaca konfigurasi dan mengatur audit | Tim audit TI |
| PROTECTED | Tidak bisa login interaktif | Wajib untuk user started task dan batch |
| REVOKE | Nonaktif | Status default user yang tidak dipakai 90 hari |

## 27.2 Class general resource yang sering dipakai

| Class | Melindungi |
|---|---|
| FACILITY | Fungsi sistem: BPX.SUPERUSER, STGADMIN.*, IRR.* |
| OPERCMDS | Perintah console, misalnya MVS.CANCEL.**, JES2.** |
| STARTED | User ID yang dipakai started task |
| SURROGAT | Hak submit job atas nama user lain |
| TSOAUTH, ACCTNUM | Hak TSO |
| SDSF | Panel dan perintah SDSF |
| UNIXPRIV | Hak superuser granular di z/OS UNIX |
| PROGRAM | Program yang dikendalikan |
| TCICSTRN, GCICSTRN | Transaksi CICS |
| DSNR, MDSNxx | Koneksi dan objek Db2 |

## 27.3 Analisis Keputusan Akses

Akses yang ditolak padahal “seharusnya boleh”, atau diizinkan padahal “seharusnya tidak”, hampir selalu bisa dijelaskan dengan urutan keputusan RACF.




*Gambar 27.1 — Alur keputusan akses RACF*

```
ALGORITMA KeputusanAksesDataset(user, dataset, level)   --
disederhanakan
  jika class atau proteksi tidak aktif maka kembalikan
sesuai kebijakan default
  profil <- profil diskret yang cocok, atau generik paling
spesifik
  jika tidak ada profil:
      kembalikan TOLAK bila PROTECTALL aktif, selain itu
diteruskan tanpa RACF
  jika HLQ dataset = user ID maka IZIN
  jika user ada di access list dengan level cukup maka
IZIN
  jika salah satu group user ada di access list dengan
level cukup maka IZIN
  jika ID(*) atau UACC memberi level cukup maka IZIN
  jika user punya atribut OPERATIONS dan tidak dibatasi
access list maka IZIN
  jika profil dalam mode WARNING maka IZIN dan catat
peringatan
  selain itu TOLAK dan tulis ICH408I (serta SMF 80 bila
diaudit)
```

| Keluhan | Analisis | Perbaikan |
|---|---|---|
| “Sudah di-PERMIT, tetap | Class di-RACLIST belum di- | SETROPTS ... |

| Keluhan | Analisis | Perbaikan |
|---|---|---|
| ditolak” | refresh, atau profil lain yang lebih spesifik yang dipakai | `REFRESH`; `LISTDSD DATASET(...) GENERIC` untuk melihat profil yang berlaku |
| “User baru bisa membaca data sensitif” | UACC atau ID(*) terlalu longgar, atau user mewarisi group yang terlalu luas | UACC `NONE`, akses lewat group khusus |
| “Tidak ada ICH408I, tapi akses terlalu luas” | Profil dalam mode WARNING | Keluarkan dari mode WARNING setelah masa transisi |
| “Operator bisa membaca semua dataset” | Atribut OPERATIONS | Batasi OPERATIONS; beri akses spesifik lewat profil |

Prinsip pengelolaan yang mencegah sebagian besar masalah: satu group per fungsi kerja, akses diberikan ke group bukan ke user, dan profil generik yang konsisten per aplikasi.

## Bab 28. Administrasi RACF Sehari-hari

```
ADDUSER TLR0123 NAME('SITI RAHMA') DFLTGRP(TELLER)
OWNER(SECADM) -
       PASSWORD(TEMP2026) TSO(PROC(IKJACCNT))
CONNECT TLR0123 GROUP(CABJKT01) AUTH(USE)
ALTUSER TLR0123 RESUME                      Buka user
terkunci
ALTUSER TLR0123 PASSWORD(BARU2026) EXPIRED   Reset kata
sandi
LISTUSER TLR0123 ALL                        Lihat profil
user

ADDSD  'BANK.PROD.**' UACC(NONE) OWNER(BANKADM)
PERMIT 'BANK.PROD.**' ID(BATCHGRP) ACCESS(UPDATE)
PERMIT 'BANK.PROD.**' ID(APPSUPP) ACCESS(READ)
LISTDSD DATASET('BANK.PROD.**') ALL

RDEFINE OPERCMDS MVS.CANCEL.** UACC(NONE)
PERMIT  MVS.CANCEL.** CLASS(OPERCMDS) ID(OPERGRP)
ACCESS(UPDATE)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

```
SEARCH CLASS(USER) REVOKE                    Daftar user
ter-revoke
```

Setiap perubahan pada class yang di-RACLIST baru berlaku setelah `SETROPTS RACLIST(class) REFRESH`. Ini adalah penyebab umum "perubahan izin tidak berlaku".

## 28.1 Menjaga database RACF

- Aktifkan database RACF primer dan backup di volume berbeda; periksa dengan `RVARY LIST`.
- Jalankan `IRRDBU00` secara berkala untuk mengekspor database ke format flat yang bisa dianalisis dengan SQL atau alat lain.
- Gunakan kata sandi berbentuk *password phrase* dan MFA (misalnya IBM Z Multi-Factor Authentication) untuk user teknis berprivilegi.
- Gunakan PassTicket untuk aplikasi yang harus login atas nama user tanpa menyimpan kata sandi.

## 28.2 Analisis: Siklus Hidup User ID

Sebagian besar temuan audit akses bukan tentang profil yang salah, tetapi tentang user ID yang tidak lagi sesuai dengan pemiliknya. Contohnya karyawan yang sudah pindah bagian tetapi masih memegang akses lama, atau karyawan yang sudah keluar tetapi user ID-nya masih aktif. Karena itu user ID dikelola sebagai siklus hidup: masuk (*joiner*), pindah (*mover*), dan keluar (*leaver*).

| Tahap | Pemicu | Tindakan di RACF | Risiko bila terlewat |
|---|---|---|---|
| Masuk | Data karyawan baru dan permintaan atasan | `ADDUSER` dengan group sesuai jabatan, kata sandi sementara kedaluwarsa | Akses diberi per orang, bukan per peran |
| Pindah | Perubahan jabatan atau unit | `CONNECT` ke group baru, `REMOVE` dari group lama | Akses menumpuk dari jabatan-jabatan lama |
| Keluar | Data karyawan keluar | `ALTUSER ... REVOKE` pada hari | User ID yatim yang bisa disalahgunakan |

| Tahap | Pemicu | Tindakan di RACF | Risiko bila terlewat |
|---|---|---|---|
| | | terakhir, hapus setelah masa retensi | |
| Tidak aktif | Tidak login selama periode tertentu | Revoke otomatis dengan `SETROPTS INACTIVE(n)` | Akun tidur menjadi target |

```
ALGORITMA ProsesKaryawanKeluar(daftar_keluar_dari_HR)
  untuk setiap karyawan k:
      ids <- semua user ID milik k (TSO, CICS, user teknis
yang dia pegang)
      untuk setiap id: ALTUSER id REVOKE
      untuk setiap user teknis yang kata sandinya dia
ketahui: ganti kata sandi
      cari dataset dan profil yang dimiliki (OWNER) oleh
k; alihkan ke group
      catat bukti dengan waktu dan pelaksana
  bandingkan: user ID aktif tanpa pasangan di data HR =
temuan untuk ditindaklanjuti
```

Recertification, yaitu peninjauan akses oleh atasan setiap kuartal atau semester, menutup celah yang lolos dari proses otomatis. Yang ditinjau adalah keanggotaan group, bukan ribuan profil satu per satu. Inilah alasan akses harus diberikan lewat group per fungsi kerja: tinjauan menjadi masuk akal dan bisa benar-benar dilakukan.

User teknis, seperti user started task, batch, dan koneksi aplikasi, dikelola terpisah. User ini ditandai `PROTECTED`, punya pemilik tim yang jelas, dan ditinjau bersama perubahan aplikasinya.

## Bab 29. Integritas Sistem

Keamanan z/OS bergantung pada integritas sistem, bukan hanya profil RACF. Program yang berjalan dalam kondisi *authorized* dapat melewati semua kontrol keamanan.

| Area sensitif | Risiko | Kontrol |
|---|---|---|
| Library APF (`PROGxx`) | Program di dalamnya bisa berjalan authorized | Batasi `UPDATE` ke library APF hanya untuk sysprog |
| PPT (`SCHEDxx`) | Program bisa melewati | Tinjau setiap entri baru |

| Area sensitif | Risiko | Kontrol |
|---|---|---|
| | pemeriksaan tertentu | |
| SVC dan exit sistem | Kode kernel buatan sendiri | Inventaris dan tanda tangan kode |
| `SYS1.PARMLIB`, `SYS1.PROCLIB` | Mengubah perilaku sistem saat IPL | `UPDATE` hanya untuk sysprog, diaudit |
| `BPX.SUPERUSER`, UID 0 | Superuser z/OS UNIX | Gunakan `UNIXPRIV` yang granular |
| Akses HMC | Kendali penuh hardware | Akun personal, MFA, log |

## 29.1 Analisis: Review Library APF secara Berkala

Setiap library APF adalah pintu menuju mode authorized. Program yang dapat ditaruh seseorang di library APF dapat melewati hampir semua kontrol keamanan. Karena itu daftar APF harus sependek mungkin dan setiap entri harus dapat dipertanggungjawabkan.

```
ALGORITMA ReviewAPF()
  daftar <- hasil D PROG,APF
  untuk setiap library L di daftar:
      jika L tidak ada di volume yang disebut maka catat
KRITIS ("APF menunjuk dataset yang tidak ada")
      profil <- profil RACF yang melindungi L
      jika tidak ada profil atau UACC di atas READ maka
catat KRITIS
      penulis <- user dan group dengan akses UPDATE atau
lebih tinggi ke L
      jika penulis bukan hanya group sysprog atau pipeline
deploy maka catat TINGGI
      jika audit UPDATE tidak aktif maka catat SEDANG
      jika L tidak dipakai produk atau aplikasi yang
terdaftar maka usulkan penghapusan
  bandingkan daftar dengan baseline di Git; entri baru
tanpa tiket = insiden (RB-050)
```

Temuan “APF menunjuk dataset yang tidak ada” terdengar sepele, padahal berbahaya. Siapa pun yang berhak membuat dataset dengan nama itu di volume tersebut otomatis mendapat library authorized. Entri

seperti ini sering tertinggal setelah produk dicopot, dan harus dihapus dari PR0Gxx dan dari daftar aktif.

| Tingkat temuan | Contoh | Tenggat perbaikan |
|---|---|---|
| Kritis | APF tanpa profil RACF, APF menunjuk dataset yang tidak ada | Segera |
| Tinggi | Pengembang punya UPDATE ke library APF produksi | Dalam pekan yang sama |
| Sedang | Audit update tidak aktif | Dalam siklus review berikutnya |
| Rendah | Library APF tidak terpakai | Dijadwalkan bersama perubahan berikutnya |

# Bab 30. Audit dan SMF

Setiap keputusan keamanan dapat dicatat ke SMF. Record tipe 80 berisi event RACF, tipe 81 berisi inisialisasi RACF, dan tipe 83 berisi event audit tambahan seperti perubahan label dataset.

```
SETROPTS AUDIT(DATASET USER GROUP)     Audit perubahan
profil
SETROPTS LOGOPTIONS(FAILURES(DATASET)) Catat akses gagal
ALTDSD 'BANK.PROD.**' AUDIT(ALL(UPDATE))  Audit semua
update
```

Unload SMF tipe 80 dengan utilitas IRRADU00, lalu kirimkan ke SIEM. Banyak bank memakai produk seperti IBM zSecure atau alat pihak ketiga untuk mengirim event mainframe secara real time.

## 30.1 Analisis: Merancang Audit yang Berguna

Audit keamanan sering gagal di dua ujung. Di satu ujung, event penting tidak dicatat. Di ujung lain, semua dicatat tetapi tidak ada yang membacanya, sehingga log hanya menjadi biaya penyimpanan.


*Gambar 30.1 — Alur audit keamanan*

Perencanaan kapasitas audit dimulai dari volume event:

$$V_{\text{harian}} \approx N_{\text{event}} \times \bar{s}_{\text{record}} \qquad V_{\text{retensi}} = V_{\text{harian}} \times D_{\text{retensi}}$$

Contoh: 2 juta event RACF per hari dengan rata-rata ukuran record sekitar 600 byte menghasilkan sekitar 1,2 GB per hari, atau sekitar 438 GB untuk retensi satu tahun. Angka ini perlu diukur di sistem Anda sendiri, tetapi urutan besarnya membantu merencanakan penyimpanan dan lisensi SIEM.

Alert yang baik diukur dari presisinya, yaitu porsi alert yang benar-benar membutuhkan tindakan:

$$\text{Presisi} = \frac{\text{alert yang benar}}{\text{total alert}}$$

Jika presisi di bawah sekitar 50%, tim mulai mengabaikan alert, termasuk yang benar. Tinjau alert palsu setiap bulan dan setel ulang ambangnya. Lebih baik sepuluh alert bermutu tinggi daripada seratus yang diabaikan.

| Kesalahan umum | Akibat | Perbaikan |
|---|---|---|
| Audit `ALL(READ)` untuk semua dataset | Volume SMF meledak, event penting tenggelam | Audit baca hanya untuk data sensitif |
| SMF 80 tidak dikirim ke SIEM | Serangan terlihat hanya setelah dianalisis manual | Streaming atau unload terjadwal |
| Alert tanpa pemilik | Tidak ada yang bertindak | Setiap aturan punya pemilik dan runbook |
| Retensi terlalu pendek | Investigasi insiden lama tidak bisa dilakukan | Retensi mengikuti kebijakan dan regulasi |

# Bab 31. Enkripsi dan Kriptografi

IBM Z memiliki kriptografi hardware di setiap core (CPACF) dan kartu Crypto Express sebagai HSM bersertifikat. Keduanya diakses aplikasi lewat ICSF.

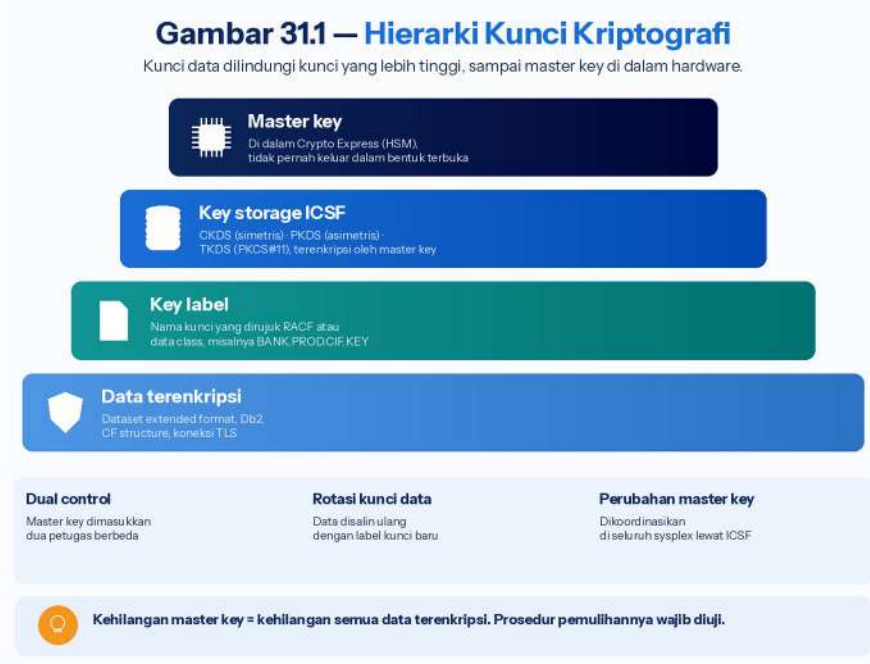
| Kemampuan | Yang dilindungi | Cara mengaktifkan |
|---|---|---|
| Pervasive encryption dataset | Dataset extended format, termasuk VSAM dan data Db2 | Key label di segmen DFP profil dataset atau data class |
| AT-TLS | Koneksi TCP/IP tanpa mengubah aplikasi | Policy Agent (PAGENT) |
| Coupling Facility encryption | Data dalam structure CF | Parameter `ENCRYPT` di CFRM policy |
| Kunci PIN dan kartu | Transaksi ATM dan kartu | Crypto Express dalam mode CCA |

Catatan kepatuhan: di Indonesia, penyelenggaraan TI bank diatur oleh POJK Nomor 11/POJK.03/2022 dan surat edaran turunannya. Pemrosesan kartu juga tunduk pada PCI DSS. Petakan setiap kontrol di bab ini ke pasal atau requirement yang relevan dalam dokumen kepatuhan internal Anda.

## 31.1 Analisis: Manajemen Kunci Kriptografi

Enkripsi hanya sekuat pengelolaan kuncinya. Di IBM Z, kunci diatur dalam hierarki: kunci data dilindungi kunci yang lebih tinggi, sampai ke master

key yang tidak pernah meninggalkan kartu Crypto Express dalam bentuk terbuka.




*Gambar 31.1 — Hierarki kunci kriptografi*

ICSF menyimpan kunci operasional dalam tiga dataset: CKDS untuk kunci simetris, PKDS untuk kunci asimetris, dan TKDS untuk kunci PKCS#11. Isinya terenkripsi oleh master key, sehingga dataset ini tidak berguna jika dicuri tanpa kartu kriptonya.

Untuk pervasive encryption dataset, kunci data dirujuk lewat **key label**. Label ini didefinisikan di segmen DFP profil dataset RACF atau di data class. Aplikasi tidak pernah melihat kuncinya. Hak untuk memakai label dikendalikan RACF, sehingga pemisahan tugas bisa ditegakkan: storage admin dapat memindahkan dataset terenkripsi tanpa bisa membaca isinya.

| Proses | Yang terjadi | Risiko bila salah |
|---|---|---|
| Memuat master key | Bagian kunci dimasukkan oleh dua atau lebih petugas (dual control), umumnya | Satu orang menguasai seluruh kunci |

| Proses | Yang terjadi | Risiko bila salah |
|---|---|---|
| | lewat TKE | |
| Mengganti master key | CKDS/PKDS dienkripsi ulang dengan master key baru; dikoordinasikan di seluruh sysplex | Sistem lain tidak dapat membaca kunci; aplikasi gagal |
| Rotasi kunci data dataset | Label baru dibuat, data disalin ulang agar terenkripsi dengan kunci baru | Data lama tetap memakai kunci lama bila tidak disalin |
| Backup CKDS/PKDS | Disimpan terpisah dari data yang dienkripsinya | Kehilangan CKDS berarti kehilangan akses ke data |
| Pemulihan di DR | Kartu kripto DR memuat master key yang sama | Data tersedia di DR tetapi tidak dapat dibaca |

Baris terakhir tabel adalah kegagalan yang paling sering terlewat. Replikasi data ke DR berjalan sempurna, tetapi kartu Crypto Express di situs DR tidak memuat master key yang sama. Uji DR harus selalu mencakup pembacaan data terenkripsi, bukan hanya IPL dan start subsistem.

# 🏦 Bagian VIII — Middleware dan Core Banking

## Bab 32. CICS Transaction Server

CICS adalah monitor transaksi yang menjalankan hampir semua layanan online core banking di mainframe. Satu region CICS adalah satu address space yang melayani ribuan transaksi pendek secara bersamaan.

| Peran region | Fungsi |
|---|---|
| TOR (Terminal-Owning Region) | Menerima permintaan dari terminal, TCP/IP, atau web service |
| AOR (Application-Owning Region) | Menjalankan program aplikasi COBOL/Java |
| FOR (File-Owning Region) | Memiliki file VSAM; kini sering digantikan VSAM RLS |
| CICSPlex SM | Mengelola dan menyeimbangkan beban banyak region |

### 32.1 Perintah CEMT penting

```
F CICSPRD1,CEMT I TASK                      Task yang
sedang berjalan
F CICSPRD1,CEMT I FILE(ACCTMST)             Status file
F CICSPRD1,CEMT S FILE(ACCTMST) CLO         Tutup file
sebelum batch
F CICSPRD1,CEMT S FILE(ACCTMST) OPE ENA     Buka kembali
setelah batch
F CICSPRD1,CEMT S PROG(ACRINQ) PHASEIN      Muat versi
program baru
F CICSPRD1,CEMT I DB2CONN                   Status koneksi
ke Db2
```

### 32.2 Abend CICS yang paling sering

| Abend | Makna | Langkah awal |
|---|---|---|
| ASRA | Program check, misalnya | Cari offset di transaction |

| Abend | Makna | Langkah awal |
|---|---|---|
| | data desimal rusak | dump |
| AEI0 | Program tidak ditemukan (PGMIDERR) | Periksa definisi PROGRAM dan DFHRPL |
| AICA | Task berjalan terlalu lama tanpa melepas CPU | Periksa loop di program |
| AKCS | Timeout menunggu sumber daya (deadlock) | Periksa enqueue dan record lock |
| AEY9 | Db2 tidak tersedia untuk CICS | Periksa DB2CONN dan status Db2 |
| AFCR/AFCW | Masalah akses file VSAM | Periksa status file dan RLS |

Kondisi SOS (*Short on Storage*) dan batas MXT (*max tasks*) adalah dua penyebab utama CICS lambat di jam sibuk. SOS ditandai pesan seri DFHSM013x di log, sedangkan kondisi MXT terlihat di statistik dispatcher CICS.

## 32.3 Analisis: Berapa AOR yang Dibutuhkan?

Jumlah region CICS sering bertambah secara historis, misalnya satu AOR baru setiap ada aplikasi baru, bukan dari perhitungan. Padahal kebutuhan AOR bisa diperkirakan dari tiga batas: CPU di QR TCB, jumlah task (MXT), dan storage.

**Batas QR TCB.** Program yang tidak threadsafe berjalan bergantian di satu TCB utama, QR. Satu TCB hanya bisa memakai satu prosesor pada satu waktu, sehingga throughput satu AOR dibatasi CPU yang dipakai setiap transaksi di QR:

$$TPS_{\text{maks per AOR}} \approx \frac{1}{t_{\text{CPU di QR per transaksi}}} \times U_{\text{target}}$$

Contoh: jika setiap transaksi memakai 1,5 ms CPU di QR, satu AOR secara teori hanya sanggup sekitar 666 transaksi per detik. Dengan target utilisasi QR 70%, kapasitas amannya sekitar 460 TPS. Untuk puncak 1.500 TPS dibutuhkan minimal empat AOR, dan lima atau enam bila ingin tetap aman saat satu AOR di-restart.

**Batas task.** Hukum Little di Bab 35.2 menghitung jumlah task bersamaan dari TPS dan waktu respons. Jika waktu respons naik karena Db2 lambat, jumlah task naik mengikuti, dan AOR bisa mencapai MXT sebelum mencapai batas CPU.

**Batas storage.** Setiap task memakai storage di DSA. Jumlah task dikali storage per task harus muat di EDSA dengan cadangan untuk lonjakan.

| Kondisi | Tanda di statistik | Solusi |
|---|---|---|
| QR TCB jenuh | CPU QR mendekati 100% sementara CPU LPAR masih tersisa | Jadikan program threadsafe agar berpindah ke open TCB; tambah AOR |
| Mencapai MXT | Antrean task, tanpa SOS | Cari penyebab tunggu (Db2, file); baru naikkan MXT |
| SOS | Pesan `DFHSM013x` | Cari task boros storage; naikkan `EDSALIM` bila REGION cukup |
| Satu AOR bermasalah memengaruhi aplikasi lain | Banyak aplikasi di satu region | Pisahkan aplikasi kritikal ke AOR sendiri |

Membuat program threadsafe sering memberi hasil lebih besar daripada menambah AOR. Setelah program threadsafe, panggilan Db2 dan sebagian besar logika berjalan di open TCB yang bisa memakai banyak prosesor sekaligus, sehingga batas QR tidak lagi menjadi leher botol.

# Bab 33. Db2 for z/OS

Db2 for z/OS adalah basis data relasional utama di mainframe. Setiap subsistem Db2 terdiri dari beberapa address space.

| Address space | Fungsi |
|---|---|
| `ssnmMSTR` | Log, recovery, dan kontrol sistem |
| `ssnmDBM1` | Mesin database: buffer pool, eksekusi SQL |
| `ssnmDIST` | DDF untuk akses jarak jauh (JDBC, ODBC, DRDA) |
| `ssnmIRLM` | Lock manager |

## 33.1 Perintah Db2 harian

```
-DIS THREAD(*)                          Thread aktif
-DIS DB(BANKDB) SPACENAM(*) RESTRICT    Objek dalam status
terbatas
-DIS BUFFERPOOL(BP1) DETAIL             Statistik buffer
pool
-DIS LOG                                Status log aktif
-DIS UTIL(*)                            Utilitas yang
sedang berjalan
-DIS DDF                                Status DDF
-STOP DB2 MODE(QUIESCE)                 Hentikan Db2
dengan aman
```

## 33.2 Status terbatas yang wajib dikenali

| Status | Arti | Pemulihan umum |
|---|---|---|
| COPY | Copy pending, butuh image copy | Jalankan COPY |
| CHKP | Check pending, integritas referensial belum pasti | Jalankan CHECK DATA |
| RECP | Recover pending | Jalankan RECOVER |
| RBDP | Rebuild pending pada index | Jalankan REBUILD INDEX |
| STOP | Objek dihentikan | -START DB(...) SPACENAM(...) setelah penyebab jelas |
| LPL/GRECP | Halaman atau objek perlu dipulihkan (data sharing) | -START DB(...) memicu pemulihan otomatis |

Jangan memakai ACCESS(FORCE) untuk menghapus status terbatas kecuali Anda benar-benar paham akibatnya. Opsi ini bisa menyembunyikan data yang tidak konsisten.

## 33.3 Pemeliharaan rutin

- RUNSTATS setelah perubahan volume data besar agar optimizer memilih akses yang tepat.
- REORG terjadwal untuk tabel transaksi dengan banyak insert dan delete.

- `COPY` penuh mingguan dan incremental harian, ditambah FlashCopy image copy bila tersedia.
- `REBIND` terkendali setelah upgrade, dengan `APREUSE` agar access path tidak berubah tiba-tiba.

## 33.4 Analisis: Menemukan SQL yang Mahal

Di kebanyakan sistem core banking, sebagian kecil SQL memakan sebagian besar CPU Db2. Menemukan SQL itu dan memperbaikinya adalah cara paling murah untuk menurunkan MSU dan waktu respons sekaligus.

**Metrik per transaksi.** Data akuntansi Db2 (SMF 101) menyediakan angka per paket dan per transaksi. Empat angka yang paling berguna:

| Metrik | Arti | Tanda masalah |
|---|---|---|
| Getpage per transaksi | Jumlah halaman yang disentuh | Naik tajam berarti access path berubah atau data tumbuh |
| Sync I/O per transaksi | Halaman yang harus dibaca dari disk | Buffer pool kurang atau pola akses acak |
| CPU per transaksi | Biaya langsung | Dibandingkan antarrilis aplikasi |
| Lock/latch wait | Waktu menunggu kunci | Kontensi dengan batch atau transaksi lain |

**Membaca EXPLAIN.** Setelah SQL mahal ditemukan, `EXPLAIN` menunjukkan cara Db2 mengeksekusinya. Kolom yang paling sering dibaca di `PLAN_TABLE`:

| Kolom | Nilai yang dicari | Arti |
|---|---|---|
| `ACCESSTYPE` | `R` di transaksi online | Tablespace scan, hampir selalu masalah untuk OLTP |
| `MATCHCOLS` | `0` padahal ada index | Index dipakai tanpa pencocokan kolom kunci |
| `INDEXONLY` | `Y` | Bagus: data diambil dari index saja |
| `PREFETCH` | `S` atau `L` di OLTP | Membaca banyak halaman |

| Kolom | Nilai yang dicari | Arti |
|---|---|---|
| | | untuk satu transaksi |
| SORTN_*, SORTC_* | Y | Ada sort, periksa apakah bisa dihindari dengan index |

**Contoh.** Transaksi inquiry saldo biasanya menyentuh 8 getpage. Setelah rilis baru, angkanya menjadi 4.000 getpage, dan CPU per transaksi naik puluhan kali. EXPLAIN menunjukkan ACCESSTYPE = R pada tabel riwayat transaksi: predikat baru ditulis dengan fungsi pada kolom tanggal, sehingga index tidak bisa dipakai. Setelah predikat ditulis ulang menjadi rentang tanggal biasa, access path kembali memakai index dan getpage turun ke kisaran semula.

Jadikan “getpage per transaksi untuk 20 transaksi teratas” sebagai bagian review kinerja mingguan. Kenaikan angka ini adalah peringatan paling awal bahwa sebuah rilis aplikasi atau pertumbuhan data mulai merusak access path.

## Bab 34. IMS dan IBM MQ

IMS adalah database hierarkis dan transaction manager yang masih dipakai sebagian bank untuk data nasabah dan rekening lama. Jika lingkungan Anda memakai IMS, perlakukan IMS TM seperti CICS dan IMS DB seperti Db2 dari sudut pandang operasional.

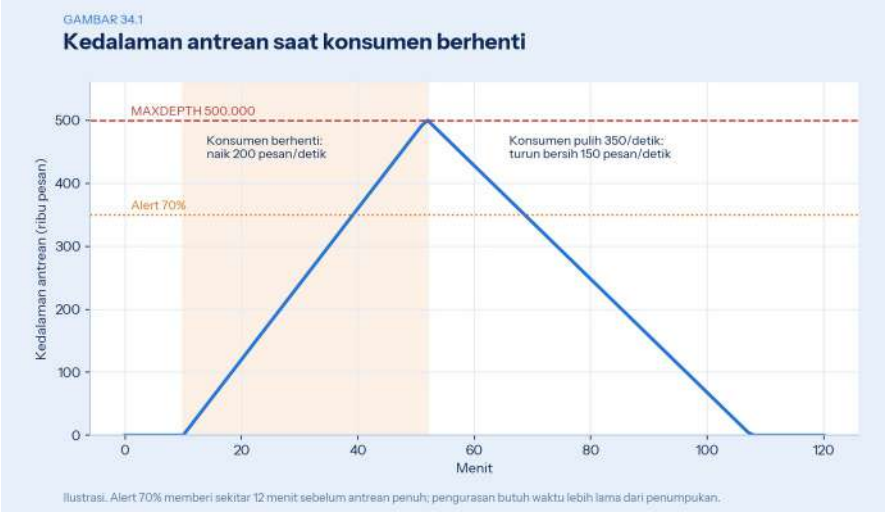
IBM MQ menjadi tulang punggung integrasi antara core banking dan sistem lain. Setiap queue manager adalah subsistem dengan command prefix sendiri, misalnya +CSQ1.

```
+CSQ1 DISPLAY QLOCAL(BANK.*) CURDEPTH MAXDEPTH   Kedalaman
antrean
+CSQ1 DISPLAY CHSTATUS(*)                        Status
channel
+CSQ1 START CHANNEL(TO.SWITCHING)                Mulai
channel
+CSQ1 DISPLAY QMGR                               Info
queue manager
```

Antrean yang terus menumpuk hampir selalu berarti konsumen di sisi lain berhenti. Pasang alert saat `CURDEPTH` melewati 70% dari `MAXDEPTH`.

## 34.1 Analisis: Kapasitas Antrean dan Waktu Pengurasan

Antrean MQ adalah penyangga antara sistem yang bekerja dengan kecepatan berbeda. Penyangga ini hanya aman jika ukurannya dan waktu pemulihannya dihitung. Dua pertanyaan penting: berapa lama antrean bisa menampung pesan saat konsumen berhenti, dan berapa lama waktu yang dibutuhkan untuk mengurasnya kembali?




*Gambar 34.1 — Kedalaman antrean saat konsumen berhenti*

$$T_{\text{penuh}} = \frac{\text{MAXDEPTH} - D_{\text{sekarang}}}{\lambda_{\text{masuk}}} \qquad T_{\text{kuras}} = \frac{D}{\mu_{\text{konsumsi}} - \lambda_{\text{masuk}}}$$

λ adalah laju pesan masuk dan μ laju konsumen memproses. Dalam ilustrasi, pesan masuk 200 per detik. Saat konsumen berhenti, antrean berkapasitas 500.000 pesan penuh dalam sekitar 42 menit. Alert di 70% berbunyi sekitar 12 menit sebelum penuh, cukup untuk bertindak jika runbook-nya jelas. Setelah konsumen pulih dengan kecepatan 350 pesan per detik, laju pengurasan bersih hanya 150 per detik, sehingga butuh sekitar 56 menit untuk kembali kosong. Pengurasan hampir selalu lebih lama daripada penumpukan, karena pesan baru tetap berdatangan.

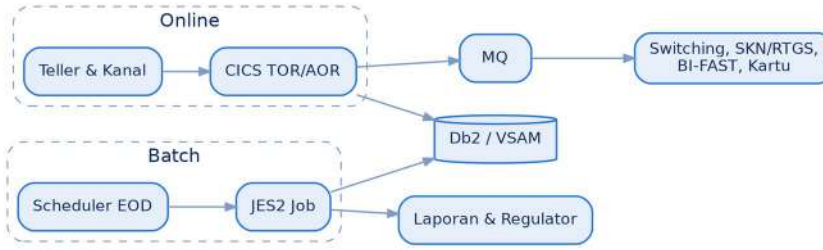
Saat antrean mencapai MAXDEPTH, aplikasi pengirim menerima error dan harus memutuskan: mencoba lagi, menyimpan sementara, atau menolak transaksi. Perilaku ini harus dirancang dan diuji, bukan ditemukan saat insiden.

| Keputusan | Pertimbangan |
|---|---|
| Ukuran MAXDEPTH | Cukup untuk menampung gangguan konsumen yang realistis, misalnya 1 jam puncak |
| Ukuran page set atau CF structure | Harus muat MAXDEPTH dikali ukuran pesan rata-rata, dengan cadangan |
| Ambang alert | Berdasarkan waktu tersisa, bukan hanya persentase |
| Jumlah instance konsumen | Laju pengurasan harus jauh di atas laju masuk puncak |
| Perilaku pengirim saat penuh | Retry dengan jeda, atau tolak dengan pesan yang jelas ke nasabah |

Alert berbasis waktu lebih berguna daripada alert berbasis persentase. “Antrean akan penuh dalam 10 menit” langsung memberi tahu seberapa mendesak situasinya, sedangkan “antrean 70%” masih harus diterjemahkan oleh petugas jaga.

## Bab 35. Lanskap Core Banking di IBM Z

Core banking di mainframe umumnya berupa aplikasi COBOL yang dikembangkan sendiri atau paket vendor yang telah lama beroperasi. Pola arsitekturnya hampir selalu sama: online di CICS, data di Db2 atau VSAM, integrasi lewat MQ, dan batch EOD di JES2.



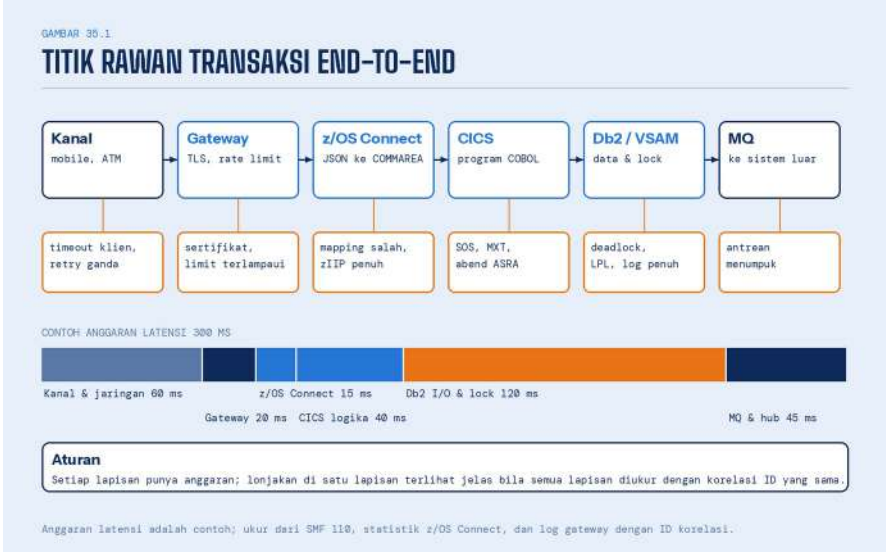

## 35.1 Fase EOD tipikal

| Urutan | Fase | Syarat sebelum lanjut |
|---|---|---|
| 1 | Cut-off online, tutup file CICS atau alihkan ke mode stand-in | Tidak ada transaksi posting tertunda |
| 2 | Backup pra-EOD (FlashCopy atau image copy Db2) | Backup terverifikasi |
| 3 | Posting transaksi tertunda dan kliring | Total batch cocok dengan total kontrol |
| 4 | Akrual bunga dan biaya | Tabel kontrol fase ditandai selesai |
| 5 | Posting ke General Ledger | GL seimbang |
| 6 | Laporan harian dan regulator | Laporan lengkap |
| 7 | Pergantian tanggal sistem | Tanggal bisnis bergeser satu hari |
| 8 | Backup pasca-EOD dan buka online (BOD) | CICS file terbuka, kanal aktif |

Prinsip rancangan yang sama dengan edisi IBM i berlaku penuh: setiap fase harus aman diulang, dan status fase dicatat di tabel kontrol, bukan hanya di scheduler. Banyak bank memakai pola “stand-in” agar transaksi kartu dan ATM tetap berjalan selama fase batch kritikal.

## 35.2 Analisis: Kegagalan End-to-End dan Anggaran Latensi

Nasabah tidak melihat CICS atau Db2; ia melihat transfer yang lambat atau gagal. Analisis yang tajam memetakan setiap lapisan yang dilewati transaksi, mode kegagalannya, dan porsi waktunya.




*Gambar 35.1 — Titik rawan transaksi end-to-end*

| Lapisan | Mode kegagalan | Dampak ke nasabah | Cara mendeteksi | Mitigasi |
|---|---|---|---|---|
| Kanal | Timeout klien lalu retry | Transaksi ganda bila tidak idempoten | Log kanal, referensi ganda di core | Referensi unik dan idempotensi (Bab 102) |
| Gateway | Sertifikat kedaluwarsa, rate limit | Semua transaksi gagal | Alert handshake dan kode 429 | Inventaris sertifikat (Bab 149) |
| z/OS Connect | Mapping salah, zIIP penuh | Error format atau lambat | Statistik z/OS Connect, RMF zIIP | Uji kontrak API, kapasitas zIIP |
| CICS | SOS, MXT, abend ASRA | Lambat atau gagal sebagian | Statistik CICS, SMF 110 | RB-013, RB-014 |
| Db2 / VSAM | Deadlock, LPL, log penuh | Timeout, SQLCODE | `-DIS` perintah, pesan DSNT | RB-004, RB-018, RB-019 |

| Lapisan | Mode kegagalan | Dampak ke nasabah | Cara mendeteksi | Mitigasi |
|---|---|---|---|---|
| | | negatif | | |
| MQ | Antrean menumpuk | Konfirmasi tertunda | CURDEPTH, alert kedalaman | RB-038, RB-057 |

**Anggaran latensi.** Waktu respons total adalah jumlah waktu di setiap lapisan:

$$T_{\text{total}} = \sum_{i=1}^{n} t_i \qquad \text{dengan target misalnya } T_{\text{total, p95}} \leq 300 \text{ ms}$$

Anggaran yang ditetapkan per lapisan membuat regresi kinerja langsung terlihat pemiliknya. Pantau persentil 95 atau 99, bukan rata-rata: rata-rata 150 ms bisa menyembunyikan 5% transaksi yang melewati 2 detik.

**Hubungan TPS dengan MXT.** Hukum Little yang dipakai untuk initiator (Bab 21.3) juga menentukan kebutuhan task CICS:

$$\text{task bersamaan} = \text{TPS} \times \text{waktu respons di CICS}$$

Contoh: 1.500 transaksi per detik dengan 0,12 detik di CICS berarti rata-rata 180 task aktif di seluruh AOR. Jika delapan AOR berbagi beban secara merata, masing-masing butuh sekitar 23 task rata-rata. Dengan cadangan untuk lonjakan dan task yang menunggu Db2, MXT per AOR di kisaran 60–80 masuk akal. Jika waktu respons Db2 berlipat dua saat ada contention, kebutuhan task ikut berlipat dan AOR bisa mencapai MXT. Karena itu masalah Db2 sering tampil sebagai masalah CICS.

# 🌐 Bagian IX — Jaringan

## Bab 36. z/OS Communications Server

Jaringan z/OS ditangani oleh Communications Server, yang terdiri dari stack TCP/IP dan VTAM untuk SNA. Hampir semua kanal modern kini masuk lewat TCP/IP, tetapi VTAM tetap wajib aktif karena TCP/IP dan banyak subsistem bergantung padanya.

| Komponen | Fungsi | Konfigurasi |
|---|---|---|
| TCPIP address space | Stack TCP/IP | `PROFILE.TCPIP` dan `TCPIP.DATA` |
| OSA-Express | Kartu jaringan fisik (mode OSD untuk Ethernet) | IODF dan `INTERFACE` di profile |
| HiperSockets | Jaringan memori antar-LPAR dalam satu CPC | IODF dan profile |
| SMC-R / SMC-D | Komunikasi TCP dipercepat lewat RDMA atau memori bersama | Profile TCP/IP |
| VTAM | SNA, APPN, dan Enterprise Extender | `ATCSTRxx`, `ATCCONxx` di VTAMLST |
| Policy Agent (PAGENT) | Kebijakan AT-TLS, IPSec, QoS | File policy di zFS |
| Resolver | Resolusi DNS | `RESOLVER` started task |

### 36.1 Analisis: Memilih Jalur Komunikasi

Komunikasi antar-komponen di IBM Z bisa lewat beberapa jalur, dan pilihannya berpengaruh besar pada latensi dan pemakaian CPU. Aturan umumnya: semakin dekat letak dua komponen, semakin pendek jalur yang sebaiknya dipakai.

| Jalur | Cocok untuk | Karakteristik |
|---|---|---|
| OSA-Express | Komunikasi keluar CPC: kanal, mitra, pusat data | Jaringan fisik standar; latensi jaringan pusat data |
| HiperSockets | Antar-LPAR dalam satu CPC | Transfer lewat memori; tidak |

| Jalur | Cocok untuk | Karakteristik |
|---|---|---|
| | | keluar ke jaringan fisik |
| SMC-D | Koneksi TCP antar-LPAR dalam satu CPC | Memakai memori bersama internal; aplikasi tetap memakai TCP biasa |
| SMC-R | Koneksi TCP antar-CPC dengan adapter RoCE | Mengurangi overhead TCP untuk lalu lintas berat antar-mesin |

Contoh penerapan: gateway API di Linux on Z yang memanggil z/OS Connect di LPAR z/OS pada CPC yang sama sebaiknya memakai HiperSockets atau SMC-D. Jalur ini menghindari perjalanan keluar ke switch dan kembali lagi, sehingga latensi turun dan beban adapter OSA berkurang. Sebaliknya, komunikasi dengan payment hub di server terpisah tetap memakai OSA.

Keunggulan SMC adalah transparansinya. Aplikasi tetap membuka koneksi TCP seperti biasa, dan stack TCP/IP yang memutuskan memakai jalur memori bila kedua ujung mendukung. Karena itu SMC dapat diaktifkan bertahap tanpa mengubah aplikasi, dengan pengukuran sebelum dan sesudah.

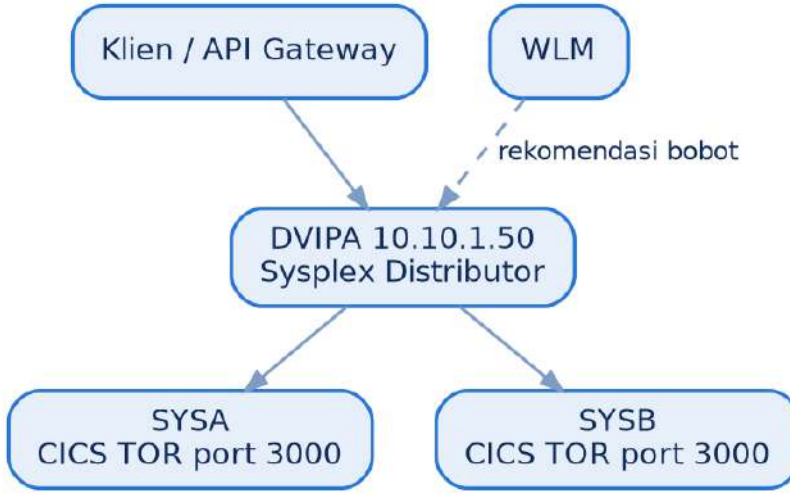
Pilihan jalur juga menyangkut keamanan. Lalu lintas lewat HiperSockets tidak melewati jaringan fisik, tetapi tetap perlu TLS bila datanya sensitif dan kebijakan bank mewajibkan enkripsi di semua jalur.

## Bab 37. VIPA dan Sysplex Distributor

VIPA (*Virtual IP Address*) adalah alamat IP yang tidak terikat ke satu kartu fisik. Jika satu OSA gagal, lalu lintas tetap mengalir lewat OSA lain tanpa klien menyadarinya.

| Jenis | Kegunaan |
|---|---|
| Static VIPA | Ketahanan terhadap kegagalan kartu dalam satu sistem |
| Dynamic VIPA (DVIPA) | Alamat pindah otomatis ke sistem lain saat sistem asal gagal |

| Jenis | Kegunaan |
|---|---|
| Distributed DVIPA (Sysplex Distributor) | Satu alamat IP dilayani beberapa sistem dengan penyeimbangan beban berbasis WLM |




Sysplex Distributor membagi koneksi baru ke sistem yang paling sehat menurut WLM. Inilah yang memungkinkan rolling IPL tanpa memutus kanal digital.

## 37.1 Analisis: Takeover DVIPA dari Sisi Klien

DVIPA sering dijelaskan dari sisi mainframe: alamat IP pindah ke sistem lain saat sistem asal gagal. Dari sisi nasabah, pertanyaannya berbeda: apa yang terjadi pada transfer yang sedang diproses tepat saat sistem berhenti?


*Gambar 37.1 — Kronologi takeover DVIPA*

Koneksi TCP yang sedang aktif ke sistem yang gagal tidak ikut pindah. Koneksi itu terputus, dan klien harus membuat koneksi baru, yang kali ini diarahkan ke sistem pengganti. Artinya, keandalan layanan saat takeover ditentukan oleh dua pihak sekaligus.

| Pihak | Tanggung jawab | Jika diabaikan |
|---|---|---|
| z/OS dan jaringan | Mendeteksi kegagalan cepat, memindahkan DVIPA, menerima koneksi baru | Klien terus mencoba ke alamat yang tidak menjawab |
| Payment hub dan gateway | Retry dengan jeda bertahap dan batas percobaan | Badai reconnect membebani sistem pengganti |
| Aplikasi | Referensi unik per transaksi dan status inquiry | Transaksi ganda atau status tidak diketahui |
| Operasi | Rekonsiliasi transaksi di sekitar waktu takeover | Selisih baru ditemukan keesokan harinya |

Timeout di sisi klien harus lebih panjang dari waktu deteksi dan takeover. Jika klien menyerah dalam 5 detik sementara takeover butuh 20 detik,

semua transaksi di jendela itu akan gagal di mata nasabah, meskipun mainframe pulih dengan sempurna. Sebaliknya, timeout yang terlalu panjang membuat nasabah menunggu tanpa kabar. Nilai yang tepat didapat dari uji takeover terencana, bukan dari tebakan.

**Uji yang disarankan setiap kuartal:** hentikan satu sistem produksi secara terencana di jam sepi, catat waktu dari berhentinya sistem sampai transaksi pertama sukses lewat sistem pengganti, lalu hitung transaksi yang gagal dan pastikan semuanya terselesaikan lewat rekonsiliasi. Hasilnya adalah angka RTO jaringan yang nyata, bukan asumsi.

## Bab 38. Diagnosis Jaringan

```
D TCPIP,,NETSTAT,HOME          Alamat IP lokal
D TCPIP,,NETSTAT,DEVLINKS      Status interface dan OSA
D TCPIP,,NETSTAT,ROUTE         Tabel routing
D TCPIP,,NETSTAT,CONN          Koneksi aktif
D TCPIP,,NETSTAT,VIPADCFG      Konfigurasi DVIPA
D TCPIP,,NETSTAT,VDPT          Distribusi Sysplex
Distributor
V TCPIP,,OBEYFILE,DSN=TCPIP.OBEY(ADDRTE)   Ubah
konfigurasi dinamis
D NET,MAJNODES                 Major node VTAM aktif
D NET,EE                       Status Enterprise Extender
```

Dari TSO atau shell z/OS UNIX tersedia `PING`, `TRACERTE`, `NETSTAT`, dan `nslookup`. Untuk analisis paket, gunakan `CTRACE` komponen `SYSTCPDA` atau packet trace dengan perintah `V TCPIP,,PKTTRACE`.

| Gejala | Kemungkinan penyebab | Pemeriksaan |
|---|---|---|
| Klien tidak bisa konek sama sekali | Port tidak listen, firewall, OSA mati | `NETSTAT CONN`, `DEVLINKS`, log firewall |
| Konek lalu putus saat handshake TLS | Sertifikat kedaluwarsa atau policy AT-TLS salah | Log PAGENT, `SYSLOGD`, masa berlaku sertifikat di RACF keyring |
| Lambat hanya di satu sistem | Distribusi DVIPA tidak seimbang | `NETSTAT VDPT`, laporan WLM |
| Transfer file lambat | Ukuran window atau MTU tidak cocok | Profil `TCPCONFIG`, trace |

| Gejala | Kemungkinan penyebab | Pemeriksaan |
|---|---|---|
| Koneksi SNA lama terputus | Link EE atau major node tidak aktif | `D NET,EE`, `D NET,ID=...` |

## 38.1 Analisis: Mengubah Profil TCP/IP dengan Aman

Profil TCP/IP menentukan alamat, rute, port, dan kebijakan jaringan z/OS. Perubahannya bisa diterapkan tanpa restart stack, yang sangat membantu, tetapi juga berarti satu kesalahan ketik bisa langsung memutus kanal nasabah.

| Praktik | Penerapan |
|---|---|
| Perubahan dinamis lewat file terpisah | `VARY TCPIP,,OBEYFILE` dengan file yang hanya berisi perubahan, bukan seluruh profil |
| Profil utama tetap diperbarui | Perubahan yang sudah diterapkan dinamis juga dimasukkan ke profil utama, agar restart berikutnya tidak menghapusnya |
| Reservasi port | Pernyataan `PORT` mengikat port ke job atau started task tertentu, sehingga program lain tidak dapat memakainya |
| Kontrol akses port | Opsi SAF pada reservasi port untuk membatasi siapa yang boleh membuka port penting |
| Uji di non-produksi | Profil yang sama diuji di sistem uji sebelum diterapkan |
| Rencana kembali | File berisi kebalikan perubahan sudah siap sebelum perubahan diterapkan |

Reservasi port sering diremehkan. Tanpa reservasi, program apa pun yang berjalan lebih dulu dapat membuka port yang seharusnya milik listener CICS atau z/OS Connect. Akibatnya, setelah restart, listener yang benar gagal membuka port-nya, dan kanal nasabah tidak dapat terhubung. Reservasi menutup kemungkinan ini.

```
ALGORITMA PerubahanProfilTCPIP(perubahan)
  siapkan file OBEY berisi perubahan dan file OBEY berisi
kebalikannya
```

```
  uji keduanya di sistem uji dengan profil yang setara
  catat kondisi sebelum: NETSTAT CONN, HOME, ROUTE,
PORTLIST
  terapkan di satu sistem produksi; verifikasi koneksi
baru dan yang ada
  jika bermasalah: terapkan file kebalikan segera
  terapkan ke sistem lain secara bergilir
  masukkan perubahan ke profil utama dan ke baseline di
Git (Bab 15.1)
```

Langkah terakhir yang paling sering terlupa adalah memasukkan perubahan ke profil utama. Perubahan dinamis yang tidak dicatat akan hilang saat stack TCP/IP berikutnya di-restart atau sistem di-IPL. Masalah yang dulu sudah diperbaiki pun muncul kembali berbulan-bulan kemudian tanpa ada yang ingat penyebabnya.

# Bab 39. Transfer File dan Interface

Pertukaran file dengan sistem lain umumnya memakai FTP dengan TLS, SFTP lewat OpenSSH di z/OS UNIX, atau produk managed file transfer seperti IBM Sterling Connect:Direct. Pilih satu standar dan terapkan konsisten di seluruh interface.

- Selalu gunakan TLS atau SSH; FTP polos tidak boleh dipakai untuk data nasabah.
- Perhatikan konversi EBCDIC ke ASCII: file teks dikonversi otomatis dalam mode ASCII, tetapi field packed decimal akan rusak jika dikonversi.
- Tetapkan code page secara eksplisit, misalnya IBM-1047 atau IBM-037 di sisi mainframe dan UTF-8 di sisi terbuka.
- Simpan setiap file interface masuk dan keluar sebagai GDG agar bisa ditelusuri dan dikirim ulang.

## 39.1 Analisis: Diagnosis Berlapis dan Kinerja Transfer

Diagnosis jaringan yang efisien bergerak dari lapisan paling bawah ke atas. Memeriksa sertifikat saat interface OSA belum aktif hanya membuang waktu.


*Gambar 39.1 — Diagnosis jaringan per lapisan*

**Mengapa transfer ke DR atau mitra lambat padahal link 1 Gbps.**
Throughput satu koneksi TCP dibatasi ukuran window dan waktu pulang-pergi (RTT), bukan hanya kapasitas link:

$$\text{Throughput}_{\text{maks}} \approx \frac{\text{Window}}{RTT} \qquad \text{Window}_{\text{perlu}} = \text{Bandwidth} \times RTT$$

Contoh: window 64 KB dengan RTT 20 ms hanya menghasilkan sekitar 3,3 MB per detik (sekitar 26 Mbps), berapa pun kapasitas link-nya. Untuk memenuhi 1 Gbps pada RTT yang sama, window yang dibutuhkan sekitar 2,5 MB.

| Situasi | Analisis | Tindakan |
|---|---|---|
| Transfer besar ke situs jauh lambat | Window lebih kecil dari bandwidth × RTT | Naikkan batas buffer TCP di `TCPCONFIG` (`TCPRCVBUFRSIZE`, `TCPMAXRCVBUFRSIZE`) sesuai kebijakan |
| Kecepatan naik-turun, banyak retransmisi | Kehilangan paket di jalur | Periksa statistik retransmisi di `NETSTAT` dan perangkat jaringan |
| Hanya satu mitra yang | Masalah di sisi mitra atau | Bandingkan RTT dan |

| Situasi | Analisis | Tindakan |
|---|---|---|
| lambat | jalur khusus | retransmisi dengan mitra lain |
| Lambat hanya saat EOD | Kontensi dengan replikasi dan batch | Jadwalkan transfer di luar puncak replikasi; QoS |

Untuk transfer antar-LPAR di CPC yang sama, HiperSockets atau SMC-D menghindari jaringan fisik sepenuhnya dan hampir selalu merupakan jalur tercepat.

# 📈 Bagian X — Kinerja dan Kapasitas

## Bab 40. Workload Manager

WLM mengalokasikan CPU, memori, dan I/O berdasarkan tujuan bisnis yang Anda tetapkan, bukan prioritas statis. Anda menyatakan “transaksi teller harus selesai 90% di bawah 0,3 detik”, lalu WLM mengatur sumber daya untuk mencapainya.

| Konsep WLM | Arti |
|---|---|
| Service class | Kelompok kerja dengan tujuan yang sama |
| Goal: response time | Target waktu respons, rata-rata atau persentil |
| Goal: velocity | Seberapa jarang kerja tertunda karena menunggu sumber daya (1–99) |
| Goal: discretionary | Hanya memakai sisa kapasitas |
| Importance | Prioritas 1 (tertinggi) sampai 5 saat tujuan tidak tercapai |
| Classification rule | Aturan memetakan transaksi atau job ke service class |
| Report class | Pengelompokan untuk pelaporan tanpa memengaruhi prioritas |
| Performance Index (PI) | PI di bawah 1 berarti tujuan tercapai; di atas 1 berarti meleset |

### 40.1 Contoh rancangan service class bank

| Service class | Isi | Goal | Importance |
|---|---|---|---|
| `SYSTEM`, `SYSSTC` | Address space sistem | Ditetapkan sistem | — |
| `STCHI` | Db2 MSTR/DBM1, IRLM, MQ, TCPIP | Velocity 70 | 1 |
| `CICSTLR` | Transaksi teller dan kanal digital | 90% < 0,3 detik | 1 |
| `DDFHI` | Akses JDBC dari API | 90% < 0,5 detik | 2 |

| Service class | Isi | Goal | Importance |
|---|---|---|---|
| BATEOD | Job kritikal EOD | Velocity 50 | 2 |
| BATNORM | Batch umum | Velocity 20 | 4 |
| TSOSYS | Sesi TSO sysprog | Periode 1 response time pendek | 3 |
| DISCR | Job development dan laporan ad hoc | Discretionary | — |

Jangan membuat terlalu banyak service class dengan importance 1. Jika semuanya penting, tidak ada yang benar-benar diprioritaskan.

```
D WLM                          Policy aktif
D WLM,SYSTEMS                  Status WLM di seluruh sysplex
V WLM,POLICY=BANKPOL           Aktifkan policy lain
E jobname,SRVCLASS=BATEOD      Pindahkan job ke service
class lain (reset)
```

## 40.2 Analisis: Menurunkan Tujuan WLM dari SLA

Tujuan WLM yang baik tidak ditebak. Tujuan itu diturunkan dari kesepakatan layanan bisnis, lalu diterjemahkan menjadi angka yang dapat diukur WLM. Kesalahan paling umum adalah memakai rata-rata, padahal nasabah merasakan ekor distribusi.


*Gambar 40.1 — Distribusi waktu respons dan tujuan persentil*

Dalam ilustrasi di atas, rata-rata waktu respons hanya sekitar 0,14 detik, tetapi 10% transaksi paling lambat memakan lebih dari 0,24 detik, dan ekornya memanjang hingga lebih dari setengah detik. Tujuan “rata-rata 0,2 detik” akan selalu tercapai dan tidak pernah memicu WLM bertindak, meskipun sebagian nasabah menunggu lama. Tujuan persentil seperti “90% di bawah 0,3 detik” mengukur pengalaman sebagian besar nasabah dan tetap sensitif terhadap perlambatan.

Langkah menurunkan tujuan:

1. Ambil SLA bisnis, misalnya “layar saldo tampil dalam 1 detik”.
2. Kurangi porsi di luar mainframe: jaringan seluler, gateway, dan aplikasi mobile. Sisanya adalah anggaran untuk z/OS (Bab 35.2).
3. Ukur distribusi aktual transaksi selama beberapa pekan dari SMF 72 atau 110.
4. Tetapkan tujuan persentil yang sedikit lebih longgar dari kondisi normal, agar PI di bawah 1 saat sehat dan naik di atas 1 saat benar-benar ada masalah.
5. Tetapkan importance berdasarkan dampak bisnis, bukan berdasarkan siapa yang paling keras meminta.

| SLA bisnis | Anggaran di z/OS | Tujuan WLM contoh | Importance |
|---|---|---|---|
| Layar teller dalam 1 detik | 0,4 detik | 90% < 0,3 detik | 1 |
| API saldo mobile dalam 2 detik | 0,5 detik | 90% < 0,4 detik | 2 |
| EOD selesai sebelum pukul 04.00 | Durasi jalur kritis | Velocity 50 untuk job jalur kritis | 2 |
| Laporan harian sebelum pukul 08.00 | Jam-jam longgar | Velocity 20 | 4 |

Tinjau tujuan setiap kali aplikasi berubah besar. Tujuan yang tidak pernah meleset selama setahun mungkin terlalu longgar. Tujuan yang meleset setiap hari mungkin tidak realistis, atau menandakan masalah kapasitas yang nyata.

# Bab 41. SMF dan RMF

SMF adalah catatan sistem untuk hampir semua kejadian di z/OS. RMF adalah pengumpul data kinerja yang menulis ke SMF dan menyediakan laporan interaktif.

| Tipe SMF | Isi | Dipakai untuk |
|---|---|---|
| 14, 15 | Dataset dibaca atau ditulis | Audit akses data |
| 30 | Aktivitas job, step, dan address space | Chargeback dan analisis batch |
| 42 | Statistik SMS dan PDSE | Kinerja storage |
| 70–79 | Data RMF: CPU, workload, I/O, CF | Analisis kinerja dan kapasitas |
| 80 | Event RACF | Audit keamanan |
| 89 | Pemakaian produk | Laporan SCRT untuk lisensi |
| 100–102 | Statistik dan akuntansi Db2 | Kinerja Db2 |
| 110 | Statistik dan monitoring CICS | Kinerja transaksi |
| 113 | Hardware counter (CPU MF) | Analisis cache dan efisiensi prosesor |
| 115, 116 | IBM MQ | Kinerja antrean |
| 119 | TCP/IP | Koneksi dan transfer file |

SMF sebaiknya ditulis ke logstream (bukan dataset `SYS1.MANx`) dan diarsip harian ke GDG. Kehilangan data SMF berarti kehilangan bukti audit dan dasar perhitungan lisensi.

| Monitor RMF | Fungsi |
|---|---|
| Monitor I | Pengumpulan berkala jangka panjang ke SMF 70–78 |
| Monitor II | Snapshot interaktif per address space |
| Monitor III | Analisis delay real-time dan historis pendek: siapa menunggu apa |

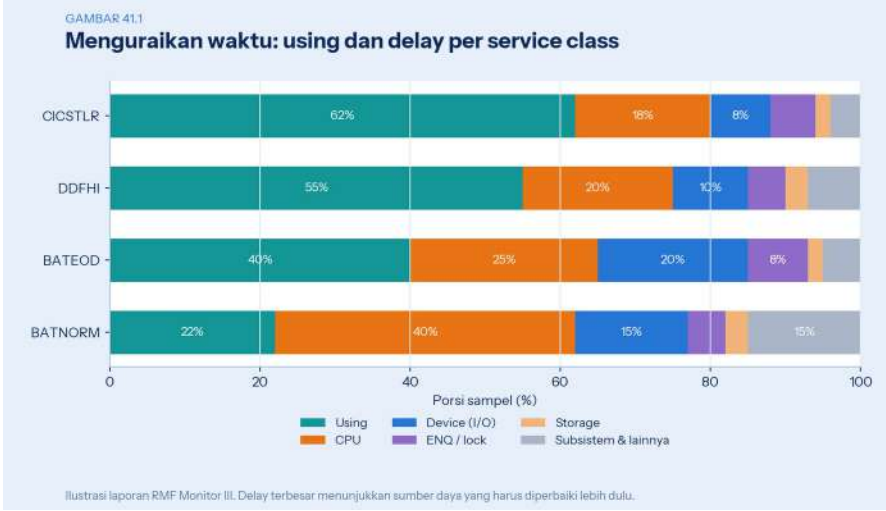
## 41.1 Metrik kunci dan ambang awal

| Metrik | Sumber | Ambang perhatian |
|---|---|---|
| CPU busy LPAR | RMF CPU Activity | Rata-rata di atas 90% selama jam sibuk |
| PI service class importance 1 | RMF Workload Activity | Di atas 1,0 lebih dari beberapa interval |
| DASD response time | RMF DASD Activity | Di atas 1 ms untuk volume produksi |
| CF service time sinkron | RMF CF Activity | Naik tajam dari baseline |
| Page fault rate | RMF Paging Activity | Paging ke disk yang berkelanjutan |
| zIIP-eligible on CP | RMF Workload | Terus meningkat |

Ambang ini hanya titik awal. Bangun baseline dari data sistem Anda sendiri selama minimal satu siklus akhir bulan.

## 41.2 Analisis: Membaca Delay dengan RMF Monitor III

RMF Monitor III menjawab pertanyaan yang paling penting saat sistem terasa lambat: pekerjaan ini sedang menunggu apa? Monitor III mengambil sampel keadaan setiap unit kerja secara berkala, lalu mengelompokkan sampel itu menjadi *using* (sedang memakai sumber daya) dan *delay* (menunggu sumber daya tertentu).




*Gambar 41.1 — Menguraikan waktu: using dan delay per service class*

| Kategori delay | Arti | Arah penyelidikan |
|---|---|---|
| CPU | Siap jalan tetapi menunggu prosesor | Kapasitas, capping, prioritas WLM, short engine |
| Device | Menunggu I/O selesai | Waktu respons I/O per volume (Bab 6.3) |
| ENQ / lock | Menunggu serialisasi dataset atau resource | `D GRS,C`, lock Db2, record lock RLS |
| Storage | Menunggu paging atau frame memori | Pemakaian real storage dan paging |
| Subsistem | Menunggu JES, HSM, atau subsistem lain | Status subsistem terkait, misalnya recall HSM |
| Operator | Menunggu mount tape atau jawaban WTOR | `D R,L`, antrean mount |

Dalam ilustrasi di atas, transaksi teller (`CICSTLR`) menghabiskan 62% sampel dalam keadaan *using*, jadi sebagian besar waktunya produktif. Batch umum (`BATNORM`) hanya 22% *using* dan 40% menunggu CPU. Artinya batch umum sedang dikorbankan WLM demi pekerjaan yang lebih penting. Itu perilaku yang benar selama batch masih selesai tepat waktu, bukan masalah yang perlu diperbaiki.

Urutan analisis yang disarankan:

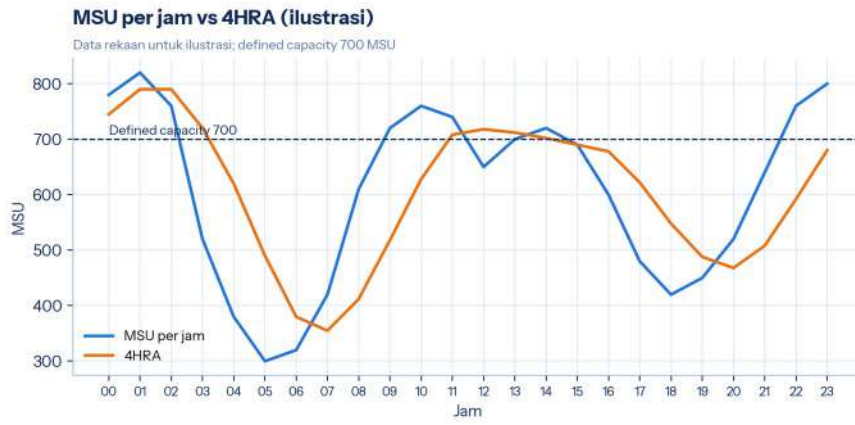
1. Mulai dari laporan ringkas sistem untuk melihat service class mana yang meleset dari tujuannya (PI di atas 1).
2. Buka laporan delay untuk service class tersebut dan cari kategori delay terbesar.
3. Masuk ke laporan detail sesuai kategori: device, ENQ, storage, atau prosesor.
4. Temukan pemilik sumber daya yang ditunggu, misalnya job pemegang ENQ atau volume yang sibuk.
5. Setelah perbaikan, ukur ulang untuk memastikan delay berpindah atau hilang, bukan sekadar berganti kategori.

## Bab 42. Capacity Planning dan Biaya

Biaya software IBM Z dihitung dari pemakaian MSU, sehingga capacity planning di mainframe selalu juga perencanaan anggaran. Kesalahan tuning dapat langsung menaikkan tagihan bulanan.

| Istilah | Arti |
|---|---|
| MSU | Satuan kapasitas untuk lisensi |
| 4HRA | Rata-rata bergulir empat jam pemakaian MSU per LPAR |
| Puncak bulanan | Nilai 4HRA tertinggi dalam satu bulan; dasar tagihan model sub-capacity |
| SCRT | Sub-Capacity Reporting Tool; laporan resmi ke IBM dari data SMF 70 dan 89 |
| Tailored Fit Pricing | Model berbasis konsumsi total, bukan puncak |

Grafik berikut menunjukkan mengapa 4HRA, bukan puncak per jam, yang menentukan tagihan. Lonjakan batch EOD dini hari mendorong 4HRA ke sekitar 790 MSU, melewati defined capacity, sehingga LPAR akan di-cap tepat saat EOD berjalan.




*Data rekaan untuk ilustrasi · 24 jam*

## 42.1 Cara menurunkan biaya tanpa mengorbankan layanan

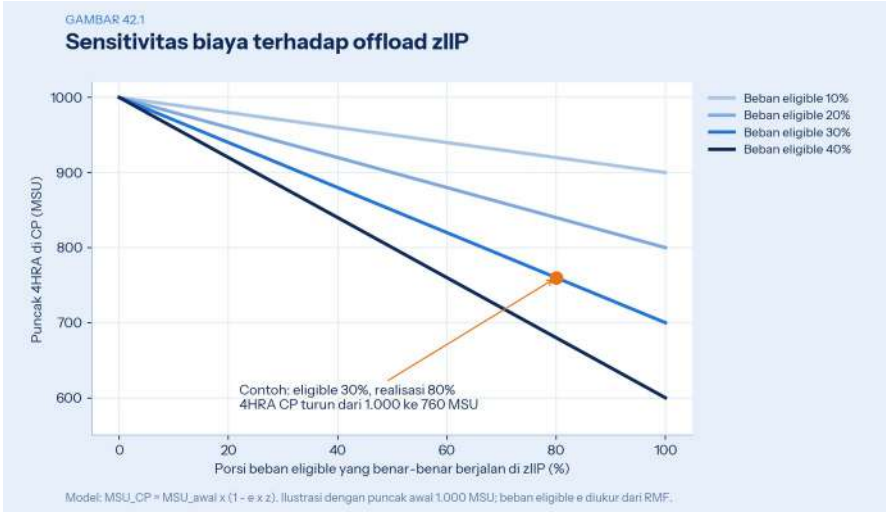
- Pindahkan puncak batch keluar dari jam puncak online agar 4HRA tidak bertumpuk.
- Maksimalkan beban eligible ke zIIP: Java, akses DDF, dan utilitas Db2 tertentu.
- Pasang defined capacity atau group capacity dengan hati-hati, dan pantau dampaknya ke batch EOD.
- Tuning SQL dan program COBOL yang paling boros; kompilasi ulang COBOL dengan versi compiler baru (Enterprise COBOL 6.x) sering menghemat CPU secara signifikan.
- Tinjau model harga dengan IBM setiap perpanjangan kontrak.

## 42.2 Analisis Sensitivitas Offload zIIP

Penghematan dari zIIP sering dijanjikan dalam persentase besar, tetapi yang menentukan tagihan adalah dua angka yang diukur sendiri: porsi beban yang eligible, dan porsi yang benar-benar berjalan di zIIP.

$$MSU_{\text{CP, sesudah}} = MSU_{\text{CP, awal}} \times (1 - e \times z)$$

e adalah porsi beban puncak yang eligible zIIP (dari RMF Workload Activity), dan z porsi beban eligible yang benar-benar dijalankan zIIP. Sisa beban eligible yang tumpah ke CP terlihat sebagai “zIIP-eligible on CP”.




*Gambar 42.1 — Sensitivitas biaya terhadap offload zIIP*

Contoh: puncak 4HRA 1.000 MSU, 30% eligible, dan 80% di antaranya berjalan di zIIP. Puncak CP turun menjadi 1.000 × (1 − 0,3 × 0,8) = 760 MSU. Jika zIIP terlalu sedikit sehingga realisasi hanya 40%, puncak hanya turun ke 880 MSU. Selisih 120 MSU itu adalah biaya dari kekurangan zIIP.

Parameter `IIPHONORPRIORITY` di `IEAOPTxx` menentukan apakah beban eligible boleh berjalan di CP saat zIIP sibuk. Nilai YES melindungi waktu respons, tetapi menaikkan MSU saat zIIP kurang. Pantau keduanya bersamaan.

| Tuas penghematan | Dampak ke 4HRA | Usaha | Risiko |
|---|---|---|---|
| Menambah zIIP agar z mendekati 100% | Tinggi bila e besar | Rendah–menengah (biaya hardware) | Kecil |
| Memindahkan logika ke Java atau API eligible | Tinggi dalam jangka panjang | Tinggi | Perubahan aplikasi |
| Menggeser batch keluar dari jam puncak | Tinggi | Rendah | Tenggat batch |

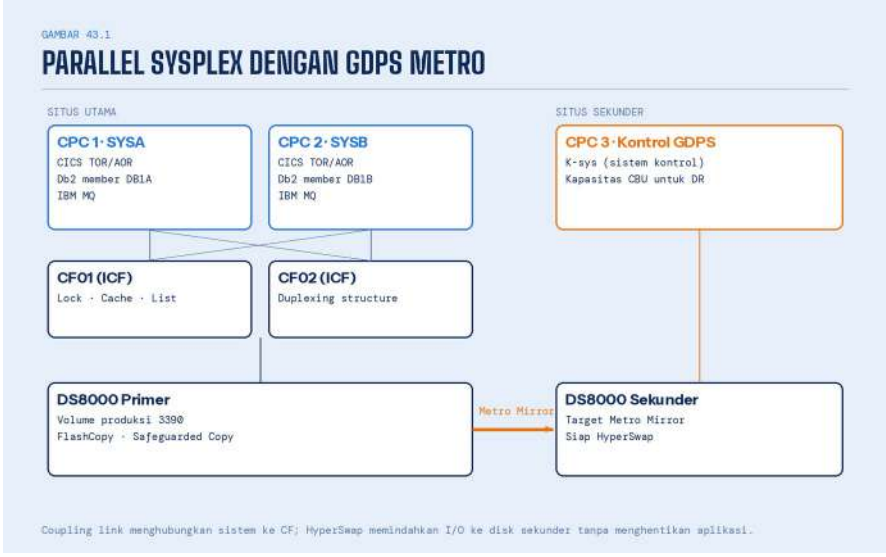
| Tuas penghematan | Dampak ke 4HRA | Usaha | Risiko |
|---|---|---|---|
| Rekompilasi COBOL dengan compiler baru | Menengah | Menengah | Pengujian regresi |
| Tuning SQL dan access path | Menengah–tinggi pada query boros | Menengah | Kecil |
| Defined capacity lebih rendah | Langsung | Rendah | Kinerja tercekik saat puncak |

Urutkan tuas dari rasio dampak terhadap usaha, dan ukur setiap perubahan dari SMF 70 dan 72 sebelum dan sesudahnya.

# 🛡️ Bagian XI — Ketersediaan, Backup, dan Pemulihan Bencana

## Bab 43. Parallel Sysplex

Parallel Sysplex menggabungkan hingga 32 sistem z/OS menjadi satu citra sistem yang berbagi data secara bersamaan. Dengan data sharing Db2 dan VSAM RLS, satu sistem bisa di-IPL sementara sistem lain tetap melayani transaksi.




*Gambar 43.1 — Parallel Sysplex dengan GDPS Metro*

| Komponen | Fungsi |
|---|---|
| XCF | Komunikasi dan pemantauan antarsistem |
| Coupling Facility (CF) | LPAR khusus berisi structure bersama: lock, cache, list |
| Couple dataset (CDS) | Konfigurasi sysplex: Sysplex, CFRM, ARM, SFM, LOGR, WLM |
| CFRM policy | Definisi structure dan CF tempatnya |
| SFM policy | Tindakan otomatis saat sistem berhenti |

| Komponen | Fungsi |
|---|---|
| | merespons |
| ARM | Restart otomatis subsistem di sistem yang sama atau lain |
| STP | Sinkronisasi waktu antar-CPC |

```
D XCF,SYSPLEX,ALL            Anggota sysplex dan statusnya
D XCF,CF,CFNAME=ALL          Status Coupling Facility
D XCF,STR                    Daftar structure
D XCF,STR,STRNAME=DSNDB0G_LOCK1   Detail satu structure
D XCF,COUPLE,TYPE=ALL        Couple dataset primer dan
alternatif
D ETR                        Status sinkronisasi waktu STP
SETXCF START,REBUILD,STRNAME=xxx,LOCATION=OTHER
Pindahkan structure ke CF lain
```

Aturan desain: minimal dua CF di CPC atau lokasi berbeda, dan setiap couple dataset memiliki alternatif di volume berbeda. Tanpa itu, sysplex hanya menambah titik kegagalan.

## 43.1 Analisis: Biaya Data Sharing

Data sharing tidak gratis. Setiap kali dua sistem mungkin memperbarui data yang sama, mereka harus berkoordinasi lewat Coupling Facility: meminta lock global, memeriksa validitas buffer, dan menulis halaman yang berubah ke group buffer pool. Semua itu memakan CPU dan waktu, dan besarnya menentukan apakah data sharing layak untuk sebuah beban.

Permintaan ke CF dapat diproses secara sinkron atau asinkron. Permintaan sinkron membuat prosesor menunggu jawaban CF, jadi sangat cepat bila CF dekat dan tidak sibuk, tetapi memakan CPU selama menunggu. Bila waktu layanan CF memburuk, z/OS dapat mengubah permintaan menjadi asinkron. Itu menghemat CPU, tetapi menambah waktu respons. Karena itu waktu layanan CF adalah metrik yang dipantau ketat (Bab 41.1).

$$\text{Overhead CF} \approx N_{\text{permintaan CF per detik}} \times t_{\text{layanan sinkron}}$$

Sebagai gambaran, 200.000 permintaan sinkron per detik dengan waktu layanan 5 mikrodetik memakan sekitar 1 detik CPU per detik, setara satu

prosesor penuh. Jika jarak CF bertambah atau CF sibuk sehingga waktu layanan naik menjadi 15 mikrodetik, biayanya naik tiga kali lipat.

**Kontensi lock.** Dua jenis kontensi global perlu dibedakan. Kontensi nyata terjadi ketika dua sistem benar-benar menginginkan resource yang sama. Kontensi palsu terjadi ketika dua resource berbeda kebetulan jatuh ke entri tabel lock yang sama di CF. Kontensi palsu murni biaya, dan biasanya dikurangi dengan memperbesar lock structure.

| Gejala | Kemungkinan penyebab | Tindakan |
|---|---|---|
| Waktu layanan CF naik | CF sibuk, link bermasalah, jarak bertambah | Periksa utilisasi CF dan link; pertimbangkan CF dedicated |
| Kontensi palsu tinggi | Lock structure terlalu kecil | Perbesar lock structure di CFRM policy |
| Kontensi nyata tinggi | Dua sistem memperbarui baris yang sama terus-menerus | Routing afinitas atau perubahan desain aplikasi |
| Castout GBP tertinggal | Beban tulis tinggi | Sesuaikan ambang castout dan ukuran GBP |

Patokan umum yang sering dipakai adalah menjaga kontensi global dalam beberapa persen dari total permintaan lock, dan kontensi palsu jauh lebih kecil lagi. Angka yang tepat untuk lingkungan Anda sebaiknya ditetapkan dari baseline sendiri dan dokumentasi terbaru IBM.

## Bab 44. HyperSwap dan GDPS

GDPS adalah solusi otomasi IBM untuk ketersediaan dan pemulihan bencana di atas replikasi DS8000. HyperSwap memindahkan I/O ke disk sekunder dalam hitungan detik tanpa menghentikan aplikasi.

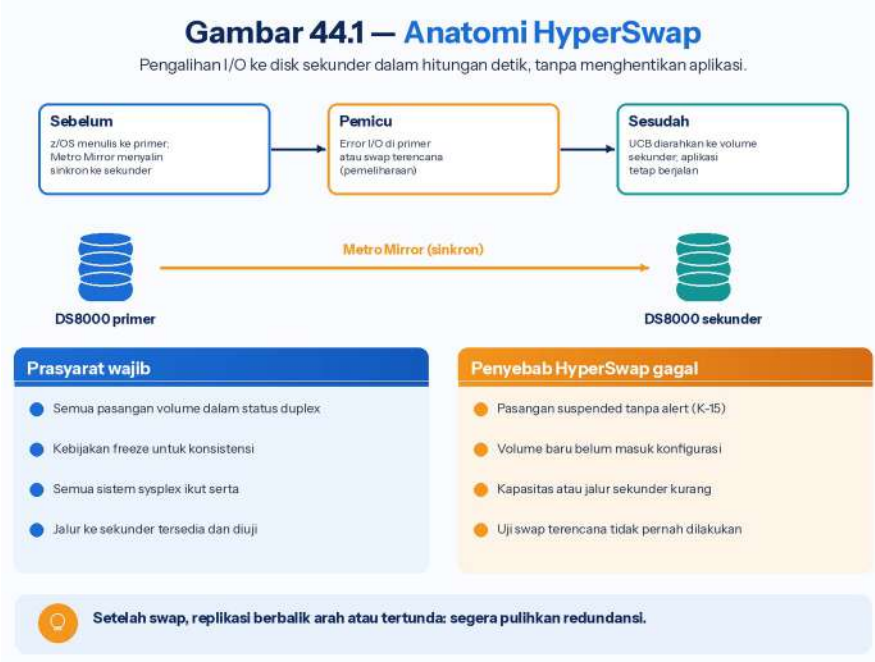
| Solusi | Replikasi | RPO tipikal | RTO tipikal | Jarak |
|---|---|---|---|---|
| GDPS Metro + HyperSwap | Metro Mirror (sinkron) | 0 | Detik untuk kegagalan disk; menit–jam untuk situs | Hingga sekitar 200 km, praktisnya jauh lebih dekat |
| GDPS Global – GM | Global Mirror (asinkron) | Detik | Kurang dari 1 jam | Tidak terbatas |

| Solusi | Replikasi | RPO tipikal | RTO tipikal | Jarak |
|---|---|---|---|---|
| GDPS Metro Global (3-site) | Sinkron + asinkron | 0 lokal, detik jarak jauh | Menit hingga di bawah 1 jam | Dua situs dekat + satu jauh |
| GDPS Continuous Availability | Replikasi software Db2/IMS/VSAM | Detik | Detik–menit | Tidak terbatas |

Angka di atas adalah kisaran umum. RPO dan RTO nyata harus dibuktikan lewat uji DR, bukan diambil dari brosur.

## 44.1 Analisis: Mengapa HyperSwap Bisa Gagal

HyperSwap adalah salah satu fitur ketersediaan terkuat IBM Z: saat disk primer bermasalah, semua I/O dialihkan ke salinan sekunder dalam hitungan detik tanpa menghentikan aplikasi. Namun fitur ini hanya bekerja jika seluruh prasyaratnya terpenuhi setiap saat, bukan hanya saat dikonfigurasi.




*Gambar 44.1 — Anatomi HyperSwap*

HyperSwap bekerja pada seluruh kelompok volume sekaligus, bukan per volume. Satu pasangan volume yang suspended atau satu volume baru yang belum dimasukkan ke konfigurasi cukup untuk membuat HyperSwap dinonaktifkan atau gagal. Karena itu status replikasi harus dipantau seketat status disk itu sendiri. Kasus K-15 di Bab 188 menunjukkan akibatnya: pasangan volume sudah suspended berhari-hari tanpa alert, dan saat disk primer bermasalah, HyperSwap tidak terjadi.

| Pemeriksaan harian | Tujuan |
|---|---|
| Semua pasangan dalam status duplex | Tidak ada volume yang tertinggal |
| HyperSwap dalam status aktif dan siap | Tidak dinonaktifkan diam-diam |
| Volume baru sudah masuk konfigurasi replikasi | Ekspansi storage tidak membuka celah |
| Jalur dan kapasitas sekunder memadai | Sekunder sanggup melayani beban penuh |

Setelah HyperSwap terjadi, sistem berjalan di disk yang dulunya sekunder, dan replikasi berada dalam keadaan terbalik atau tertunda. Pada saat itu tidak ada lagi perlindungan terhadap kegagalan kedua. Memulihkan redundansi, yaitu memperbaiki disk asal lalu menyinkronkan kembali, harus menjadi prioritas utama setelah insiden, bukan pekerjaan untuk pekan depan.

Uji HyperSwap terencana setidaknya sekali setahun, idealnya bersamaan dengan pemeliharaan storage yang memang membutuhkan swap. Uji ini membuktikan konfigurasi benar dan memberi angka nyata berapa lama aplikasi merasakan jeda.

## Bab 45. Backup dengan DFSMShsm dan DFSMSdss

DFSMShsm mengelola migrasi, backup, dan recall dataset secara otomatis berdasarkan management class. DFSMSdss (`ADRDSSU`) dipakai untuk dump dan restore volume atau dataset dalam jumlah besar.

| Fungsi | Alat | Kegunaan |
|---|---|---|
| Migrasi ML1/ML2 | DFSMShsm | Memindahkan dataset jarang dipakai ke disk murah atau |

| Fungsi | Alat | Kegunaan |
|---|---|---|
| | | tape |
| Backup incremental dataset | DFSMShsm | Backup harian dataset yang berubah |
| Dump volume | DFSMSdss / HSM autodump | Backup penuh volume, termasuk volume sistem |
| FlashCopy | DS8000 + DFSMSdss | Salinan instan konsisten sebelum EOD |
| Safeguarded Copy | DS8000 | Salinan tak dapat diubah untuk serangan siber |
| Image copy Db2 | `COPY` / FlashCopy image copy | Pemulihan tabel ke titik waktu |

```
HSEND QUERY ACTIVE                          Aktivitas HSM
saat ini
HSEND QUERY WAITING                         Permintaan yang
antre
HRECALL 'BANK.PROD.GL.2025DEC'              Recall dataset
termigrasi
HBACKDS 'BANK.PROD.CIF.MASTER'              Backup satu
dataset
HRECOVER 'BANK.PROD.CIF.MASTER' REPLACE     Restore dari
backup HSM
```

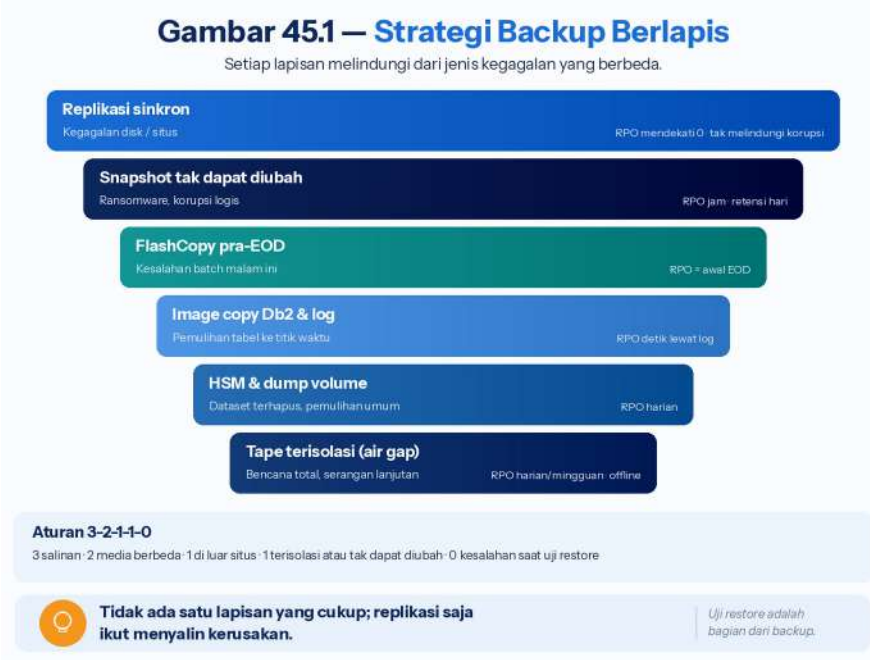
Contoh FlashCopy dataset konsisten sebelum EOD:

```
//FCPRE    EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  COPY DATASET(INCLUDE(BANK.PROD.**)) -
       RENAMEU((BANK.PROD.**,BANK.PREEOD.**)) -
       FASTREPLICATION(REQUIRED) -
       CONCURRENT TOLERATE(ENQFAILURE)
/*
```

Pastikan file CICS sudah ditutup atau di-quiesce sebelum menyalin dataset VSAM. Salinan yang diambil saat file masih diperbarui tidak dijamin konsisten.

## 45.1 Analisis: Backup Berlapis dan Memilih Sumber Pemulihan

Satu mekanisme backup tidak pernah cukup, karena setiap mekanisme melindungi dari jenis kegagalan yang berbeda. Replikasi sinkron melindungi dari kerusakan disk, tetapi ikut menyalin korupsi. Tape terisolasi melindungi dari hampir semuanya, tetapi pemulihannya lambat. Strategi yang baik menumpuk beberapa lapisan dan tahu lapisan mana yang dipakai untuk skenario apa.




*Gambar 45.1 — Strategi backup berlapis*

| Skenario | Sumber pemulihan pertama | Alasan |
|---|---|---|
| Disk atau array gagal | Replikasi (HyperSwap) | Otomatis, tanpa kehilangan data |
| Job EOD merusak data malam ini | FlashCopy pra-EOD | Titik konsisten tepat sebelum kerusakan |
| Satu tabel Db2 terhapus atau rusak | Image copy dan log Db2 | Pemulihan ke titik waktu yang presisi |
| Satu dataset terhapus | Backup HSM | Cepat, per dataset |
| Ransomware atau korupsi | Snapshot tak dapat diubah | Salinan yang tidak bisa |

| Skenario | Sumber pemulihan pertama | Alasan |
|---|---|---|
| luas | (Bab 113) | dijangkau penyerang |
| Situs dan salinan online hilang | Tape terisolasi | Garis pertahanan terakhir |

Aturan 3-2-1-1-0 merangkum prinsipnya: tiga salinan data, di dua jenis media, satu di luar situs, satu terisolasi atau tidak dapat diubah, dan nol kesalahan saat uji pemulihan. Angka nol di akhir adalah bagian yang paling sering dilupakan. Backup yang tidak pernah diuji hanyalah harapan.

```
ALGORITMA PilihSumberPemulihan(kejadian)
  tentukan waktu kerusakan paling awal T_rusak dan cakupan
(dataset, tabel, volume, situs)
  kandidat <- semua salinan yang dibuat sebelum T_rusak
dan mencakup objek terdampak
  buang kandidat yang mungkin ikut tercemar (misalnya
replikasi setelah T_rusak)
  urutkan kandidat berdasarkan: kehilangan data terkecil,
lalu waktu pemulihan tercepat
  pulihkan ke lokasi terpisah lebih dulu bila ada keraguan
tentang kebersihan data
  validasi: jumlah record, total kontrol, integritas Db2,
rekonsiliasi GL
  catat keputusan, sumber yang dipakai, dan RPO/RTO yang
benar-benar tercapai
```

Kesalahan paling mahal adalah langsung menimpa data produksi dengan salinan pertama yang ditemukan. Jika salinan itu ternyata ikut rusak, jalan kembali hilang. Pulihkan ke lokasi terpisah, validasi, baru gantikan data produksi.

## Bab 46. Merancang Latihan DR yang Bermakna

Uji DR yang baik membuktikan bahwa bisnis bisa berjalan di situs cadangan, bukan hanya bahwa sistem bisa di-IPL. Rancang latihan dengan skenario, kriteria sukses, dan pengukuran waktu yang jelas.

1. Tetapkan skenario: kehilangan situs total, kehilangan storage, atau korupsi logis.

2. Ukur RTO dari keputusan deklarasi bencana hingga transaksi teller pertama sukses.
3. Ukur RPO dengan membandingkan transaksi terakhir di situs utama dan di situs DR.
4. Jalankan EOD penuh di situs DR minimal satu kali setahun.
5. Libatkan tim aplikasi, jaringan, dan bisnis, bukan hanya sysprog.
6. Dokumentasikan setiap penyimpangan dan tindak lanjutnya sebelum latihan berikutnya.

Simpan salinan cetak prosedur pemulihan, daftar kontak, alamat IPL, dan LOADPARM di lokasi DR. Saat bencana, akses ke wiki internal belum tentu tersedia.

## 46.1 Analisis: Memilih Solusi DR Berdasarkan Risiko

Tidak semua sistem membutuhkan RPO nol. Memilih solusi DR adalah menyeimbangkan biaya solusi dengan kerugian yang dihindarinya.


*Gambar 46.1 — Spektrum solusi DR: RPO, RTO, dan biaya relatif*

| Tier layanan | Contoh | Target RPO / RTO | Solusi yang sepadan |
|---|---|---|---|
| Tier 0 | Core banking online, pembayaran | Mendekati nol / di bawah 1 jam | Metro Mirror + HyperSwap, ditambah replikasi |

| Tier layanan | Contoh | Target RPO / RTO | Solusi yang sepadan |
|---|---|---|---|
| | | | jarak jauh |
| Tier 1 | Kanal digital, kartu | Menit / 1–4 jam | Global Mirror atau GDPS Global |
| Tier 2 | Pelaporan, analitik | Jam / hingga 24 jam | Backup disk atau replikasi berjadwal |
| Tier 3 | Arsip, pengembangan | Hari / beberapa hari | Backup tape terisolasi |

Perbandingan ekonomis memakai perkiraan kerugian tahunan (*annualized loss expectancy*):

$$ALE = \sum_{s} p_s \times (RTO_s \times K_{\text{per jam}} + RPO_s \times K_{\text{data per jam}})$$

p adalah peluang tahunan skenario s, K per jam adalah kerugian per jam layanan berhenti, dan K data per jam adalah biaya merekonstruksi transaksi yang hilang.

Contoh ilustratif: kerugian Rp2 miliar per jam layanan berhenti, dan peluang kehilangan situs 1% per tahun. Solusi tape dengan RTO 48 jam menghasilkan ALE sekitar Rp0,96 miliar per tahun hanya dari downtime. Solusi dengan RTO 1 jam menurunkan angka itu menjadi sekitar Rp0,02 miliar. Selisih sekitar Rp0,94 miliar per tahun adalah batas atas yang wajar untuk biaya tambahan solusi yang lebih cepat, sebelum memperhitungkan risiko reputasi dan regulasi.

Angka dalam contoh ini rekaan. Yang penting adalah caranya: ganti dengan estimasi bisnis bank Anda, dan pastikan target RPO/RTO memenuhi ketentuan OJK tentang penyelenggaraan teknologi informasi.

# 📋 Bagian XII — Operasi Harian dan Referensi Perintah

## Bab 47. Irama Operasional

Operasi mainframe yang stabil lahir dari pemeriksaan rutin yang membosankan tetapi konsisten. Checklist berikut adalah titik awal yang perlu disesuaikan dengan lingkungan Anda.

### 47.1 Harian

- ☐ Periksa SR di SDSF: tidak ada WTOR yang menunggu tanpa pemilik.
- ☐ Periksa pemakaian spool (`$D SPOOL`) dan storage group utama (`D SMS,SG(...)`).
- ☐ Pastikan EOD selesai sesuai jadwal dan GL seimbang.
- ☐ Periksa hasil backup malam dan migrasi HSM.
- ☐ Tinjau exception Health Checker di panel `CK`.
- ☐ Periksa status CF, structure, dan couple dataset (`D XCF,CF`).
- ☐ Tinjau pesan error di OPERLOG untuk abend sistem (`IEA995I`, `IEE` dan `IOS` berulang).

### 47.2 Mingguan

- ☐ Terima HOLDDATA terbaru dan jalankan `REPORT ERRSYSMODS`.
- ☐ Tinjau tren CPU, 4HRA, dan PI service class penting.
- ☐ Tinjau user RACF yang ter-revoke atau tidak aktif.
- ☐ Periksa masa berlaku sertifikat TLS di keyring RACF.

### 47.3 Bulanan dan triwulanan

- ☐ Kirim laporan SCRT ke IBM sesuai jadwal kontrak.
- ☐ Tinjau kapasitas storage, spool, dan zFS untuk tiga bulan ke depan.

- ☐ Uji restore acak satu dataset dan satu tabel Db2.
- ☐ Tinjau akses SPECIAL, OPERATIONS, dan library APF.
- ☐ Latihan switchover sebagian (misalnya HyperSwap terencana) setiap triwulan.

## 47.4 Analisis: Serah Terima Shift dan Kelelahan Alert

Banyak insiden malam berawal dari informasi yang tidak sampai di pergantian shift: job yang di-hold sore tadi dengan alasan yang hanya diketahui satu orang, perubahan yang dipasang pukul lima, atau peringatan kapasitas yang “nanti saja”. Serah terima yang terstruktur menutup celah itu.

```
# Serah terima shift — <tanggal> <jam>
- Insiden terbuka: <nomor, status, langkah berikutnya,
pemilik>
- Job yang ditahan atau dijalankan ulang: <nama, alasan,
instruksi>
- Perubahan hari ini: <tiket, sistem, risiko untuk malam
ini>
- Kapasitas: spool, storage group, log Db2 (angka, bukan
"aman")
- Hal yang harus dipantau: <misalnya EOM, jatuh tempo
deposito massal>
- Kontak eskalasi malam ini: <nama dan nomor>
```

Masalah kedua yang sering diremehkan adalah kelelahan alert. Petugas jaga yang menerima ratusan alert per shift akan mulai mengabaikannya, termasuk alert yang penting. Beban alert dapat diukur sederhana:

$$\text{Beban alert} = \frac{\text{jumlah alert yang butuh tindakan manusia}}{\text{jam per shift}}$$

| Beban alert per jam | Kondisi | Tindakan |
|---|---|---|
| Di bawah 2 | Sehat | Pertahankan |
| 2–5 | Mulai melelahkan | Tinjau alert yang paling sering; otomasi yang rutin |
| Di atas 5 | Berbahaya | Hentikan alert yang tidak ditindaklanjuti; perbaiki akar |

| Beban alert per jam | Kondisi | Tindakan |
|---|---|---|
| | | masalah yang memicunya |

Setiap alert harus lolos satu pertanyaan: apakah alert ini meminta manusia melakukan sesuatu sekarang? Jika tidak, alert itu lebih tepat menjadi laporan harian atau entri log, bukan panggilan ke petugas jaga.

# Bab 48. Referensi Perintah Console z/OS

Perintah console diketik di console MCS/SMCS, HMC, atau lewat `/` di SDSF. Sebagian besar perintah punya bentuk singkat, misalnya `D` untuk `DISPLAY` dan `F` untuk `MODIFY`.

## 48.1 Status sistem

| Perintah | Fungsi |
|---|---|
| `D IPLINFO` | Waktu IPL, LOADxx, IEASYSxx, dan volume SYSRES |
| `D SYMBOLS` | System symbol yang aktif |
| `D PARMLIB` | Konkatenasi PARMLIB aktif |
| `D M=CPU` / `D M=STOR` | Prosesor dan memori |
| `D A,L` | Address space aktif |
| `D A,jobname` | Detail satu address space |
| `D ASM` | Page dataset dan pemakaiannya |
| `D SMF` | Dataset atau logstream SMF |
| `D PROD,STATE` | Produk yang terdaftar dan statusnya |
| `D OPDATA` | Prefix perintah subsistem (Db2, MQ, dll.) |

## 48.2 Pengendalian kerja

| Perintah | Fungsi |
|---|---|
| `S procname` | Menjalankan started task |
| `P jobname` | Menghentikan started task dengan normal |

| Perintah | Fungsi |
|---|---|
| `F jobname,parameter` | Mengirim perintah ke address space |
| `C jobname` | Membatalkan job atau started task |
| `C jobname,DUMP` | Membatalkan sambil mengambil dump |
| `FORCE jobname,ARM` | Paksa berhenti jika C tidak berhasil (hanya sebagai langkah terakhir) |
| `R nn,teks` | Membalas WTOR nomor nn |
| `D R,L` | Daftar WTOR yang menunggu balasan |

## 48.3 Sumber daya dan contention

| Perintah | Fungsi |
|---|---|
| `D GRS,C` | Contention ENQ dan latch saat ini |
| `D GRS,RES=(SYSDSN,BANK.PROD.*)` | Siapa memegang dataset tertentu |
| `D GRS,ANALYZE,BLOCKER` | Penyebab utama antrean |
| `D U,,,8000,1` | Status satu device |
| `V 8000,OFFLINE` / `V 8000,ONLINE` | Ubah status device |

## 48.4 Console dan pesan

| Perintah | Fungsi |
|---|---|
| `D C` | Status console |
| `V CN(*),ACTIVATE` | Aktifkan console darurat di HMC |
| `K E,D` / `K C` | Hapus pesan dari layar console |
| `SET MPF=xx` | Aktifkan daftar penyaringan pesan |
| `D MPF` | Status MPF |

## 48.5 Perbandingan dengan perintah IBM i

| Kebutuhan | IBM i | z/OS |
|---|---|---|
| Lihat pekerjaan aktif | `WRKACTJOB` | `D A,L` atau SDSF `DA` |

| Kebutuhan | IBM i | z/OS |
|---|---|---|
| Lihat log sistem | `DSPLOG`, `WRKMSG QSYSOPR` | SDSF `LOG` / `OPERLOG` |
| Balas pesan menunggu | `WRKMSG`, opsi reply | `R nn,teks` |
| Hentikan job | `ENDJOB` | `C jobname` |
| Lihat status disk | `WRKDSKSTS` | `D SMS,SG(...)`, RMF DASD |
| Lihat output spool | `WRKSPLF` | SDSF `O`, `H`, `ST` |
| Mulai subsystem | `STRSBS` | `S procname` |
| Lihat lock/record lock | `WRKOBJLCK` | `D GRS,C` |

## 48.6 Analisis: Klasifikasi Risiko Perintah Console

Tidak semua perintah console sama berbahayanya. `D A,L` hanya menampilkan informasi, sedangkan `FORCE` dapat mematikan address space secara paksa dan meninggalkan sumber daya dalam keadaan tidak konsisten. Mengelompokkan perintah menurut risikonya memudahkan pembagian hak akses dan penulisan runbook.

| Kelas risiko | Contoh perintah | Siapa yang boleh | Kontrol |
|---|---|---|---|
| Baca saja | `D`, `$D`, `DISPLAY` subsistem | Operator, app support, sysprog | Bebas, tetap tercatat di SYSLOG |
| Operasional | `S`, `P`, `$S`, `$P`, `$H`, `$A`, `R` | Operator, sysprog | Sesuai runbook |
| Mengubah konfigurasi | `SET`, `SETPROG`, `SETXCF`, `V ... ONLINE/OFFLINE` | Sysprog | Tiket perubahan; catat ke PARMLIB |
| Destruktif | `C`, `FORCE`, `$C`, `$P JES2`, `Z EOD` | Operator senior, sysprog | Konfirmasi dua orang untuk sistem produksi |

Kelas-kelas ini ditegakkan dengan profil RACF di class `OPERCMDS`, misalnya:

```
RDEFINE OPERCMDS MVS.FORCE.**   UACC(NONE)
RDEFINE OPERCMDS MVS.SETPROG.** UACC(NONE)
RDEFINE OPERCMDS MVS.VARY.**    UACC(NONE)
RDEFINE OPERCMDS MVS.CANCEL.**  UACC(NONE)
```

```
PERMIT  MVS.FORCE.**   CLASS(OPERCMDS) ID(SYSPROG)
ACCESS(UPDATE)
PERMIT  MVS.CANCEL.**  CLASS(OPERCMDS) ID(OPERSR)
ACCESS(UPDATE)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Nama profil yang tepat untuk setiap perintah ada di dokumentasi *z/OS MVS Planning: Operations*. Aktifkan audit untuk kelas destruktif dan kelas perubahan konfigurasi, lalu tinjau pemakaiannya setiap bulan bersama review SDSF (Bab 24.1).

**Aturan emas `FORCE`.** Gunakan `FORCE` hanya setelah `CANCEL` gagal dan dampaknya dipahami. Untuk address space yang dikelola ARM, gunakan `FORCE ... ,ARM` sesuai prosedur. Catat alasannya di log insiden, karena address space yang di-FORCE sering meninggalkan ENQ, common storage, atau koneksi subsistem yang harus dibersihkan kemudian.

# 🚨 Bagian XIII — Troubleshooting dan Katalog Runbook

## Bab 49. Katalog Abend Sistem

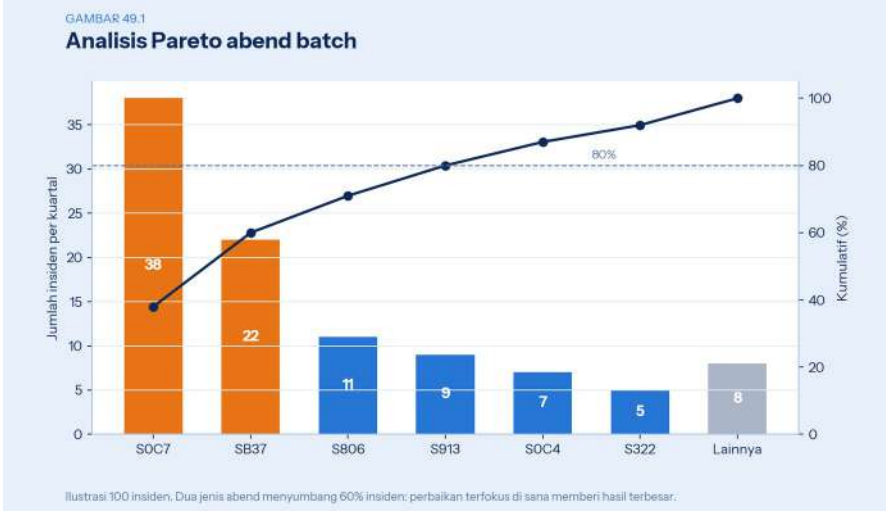
Abend adalah penghentian abnormal program. Kode Sxxx berasal dari sistem, sedangkan reason code yang menyertainya sering lebih penting daripada kodenya sendiri.

| Abend | Makna | Penyebab umum di core banking |
|---|---|---|
| `S0C1` | Operation exception | Program memanggil modul yang salah atau alamat rusak |
| `S0C4` | Protection exception | Subscript di luar batas array, pointer tidak valid |
| `S0C7` | Data exception | Field packed decimal berisi spasi atau karakter bukan angka |
| `S0CB` | Pembagian dengan nol | Rumus bunga dengan pembagi nol |
| `S013` | Masalah OPEN dataset | DCB tidak cocok, member PDS tidak ada |
| `S122` / `S222` | Dibatalkan operator | Pembatalan manual, dengan (122) atau tanpa dump (222) |
| `S322` | Batas waktu CPU terlampaui | Loop atau parameter `TIME` terlalu kecil |
| `S722` | Batas baris output terlampaui | Loop yang menulis laporan |
| `S806` | Modul tidak ditemukan | STEPLIB salah atau modul belum dipromosikan |
| `S80A` / `S878` | Virtual storage tidak cukup | `REGION` terlalu kecil atau kebocoran memori |
| `S913` | Ditolak RACF | User batch tidak punya akses ke dataset |

| Abend | Makna | Penyebab umum di core banking |
|---|---|---|
| SB37, SD37, SE37 | Ruang dataset habis | Lihat Bab 16 |

## 49.1 Analisis: Pareto Abend untuk Memilih Perbaikan

Tim yang sibuk memadamkan abend setiap malam jarang punya waktu untuk mencegahnya. Analisis Pareto membantu memilih perbaikan yang paling berdampak dengan waktu yang terbatas: kumpulkan insiden satu kuartal, kelompokkan menurut jenis abend, lalu urutkan dari yang paling sering.




*Gambar 49.1 — Analisis Pareto abend batch*

Dalam ilustrasi ini, `S0C7` dan SB37 menyumbang 60 dari 100 insiden. Menghilangkan sebagian besar keduanya akan mengurangi beban jaga lebih banyak daripada memperbaiki semua jenis abend lainnya sekaligus.

| Abend | Akar masalah yang biasa | Perbaikan pencegahan | Bab |
|---|---|---|---|
| `S0C7` | Data kotor dari file interface | Validasi header, trailer, dan field numerik di step awal | 77, 147 |
| SB37 | SPACE ditebak, | Hitung SPACE; data | 18.3 |

| Abend | Akar masalah yang biasa | Perbaikan pencegahan | Bab |
|---|---|---|---|
| | volume akhir bulan lebih besar | class extended format | |
| S806 | Deploy tidak lengkap | Verifikasi library dan PROC otomatis setelah deploy | 19.6 |
| S913 | Akses belum diberikan sebelum rilis | Uji akses di UAT dengan user produksi yang setara | 28.2 |
| S0C4 | Subscript di luar batas, pointer rusak | Opsi SSRANGE di uji, review kode | 74 |
| S322 | Loop atau TIME terlalu kecil | TIME realistis per step, alert CPU | 90 |

Ukur keberhasilan dari tren, bukan dari satu kuartal. Jika jumlah S0C7 turun dari 38 ke 10 setelah validasi interface diterapkan, hasilnya layak dilaporkan ke manajemen sebagai pengurangan risiko operasional yang nyata. Ulangi analisis setiap kuartal, karena setelah dua penyebab teratas teratasi, urutan Pareto akan berubah.

## Bab 50. Katalog Pesan Operasional

| Message ID | Arti | Tindakan / runbook |
|---|---|---|
| IEF450I | Job berakhir dengan abend | Baca SYSOUT step yang gagal |
| ICH408I | Pelanggaran akses RACF | Verifikasi apakah akses memang sah, lalu PERMIT atau eskalasi ke security |
| IEC161I | Masalah OPEN/CLOSE VSAM | Periksa return code, jalankan VERIFY |
| $HASP050 | Kekurangan sumber daya JES2 | RB-002 |
| IRA200E / IRA201E | Kekurangan auxiliary storage (page dataset) | RB-007 |
| IEA404A / IEA405E | Kekurangan buffer WTO | Cari address space yang membanjiri pesan |

| Message ID | Arti | Tindakan / runbook |
|---|---|---|
| IOS071I | Start pending atau missing interrupt pada device | Periksa jalur I/O dan storage |
| IXC402D | Sistem lain dianggap tidak merespons | RB-009 |
| DSNT375I / DSNT376I | Deadlock atau timeout Db2 | RB-004 |
| IEA995I | Symptom dump dari abend | Catat PSW, modul, dan offset |

Seperti saran di edisi IBM i, bangun kamus pesan internal: setiap message ID yang pernah muncul di produksi dicatat bersama arti, dampak, dan tindakannya.

## 50.1 Analisis: Membangun Kamus Pesan dari OPERLOG

Satu sistem produksi menghasilkan ratusan ribu pesan per hari. Hampir semuanya rutin, tetapi di antaranya terselip beberapa pesan yang menandakan masalah serius. Kamus pesan, yaitu daftar pesan penting beserta arti dan tindakannya, membantu petugas jaga memisahkan keduanya dengan cepat.

Kamus yang baik dibangun dari data, bukan dari ingatan. Langkahnya dimulai dengan menghitung frekuensi setiap ID pesan dalam sebulan OPERLOG, lalu mengelompokkannya.

```
ALGORITMA BangunKamusPesan(operlog_30_hari)
  hitung frekuensi setiap ID pesan (tujuh atau delapan
karakter pertama)
  kelompokkan:
      RUTIN     : muncul setiap hari dalam jumlah stabil,
tidak pernah terkait insiden
      PENTING   : pernah muncul dalam laporan insiden atau
postmortem
      LANGKA    : muncul kurang dari lima kali; tinjau
satu per satu
      BARU      : tidak ada di kamus sebelumnya
  untuk setiap pesan PENTING dan LANGKA yang relevan:
      tulis arti singkat, dampak, tindakan pertama, dan
runbook terkait
  pesan RUTIN yang membanjiri: pertimbangkan penekanan
```

```
lewat MPFLSTxx
  jalankan ulang setiap bulan; pesan BARU masuk antrean
peninjauan
```

| Kolom kamus | Contoh isi |
|---|---|
| ID pesan | `IEF450I` |
| Arti singkat | Job berakhir abnormal dengan kode abend |
| Dampak | Tergantung job; cek apakah job ada di jalur kritis EOD |
| Tindakan pertama | Lihat kode abend, buka runbook sesuai kode |
| Runbook | RB-006, Bab 77 untuk `S0C7` |
| Pemilik | Tim batch |

Pesan yang terlalu sering muncul dan tidak pernah membutuhkan tindakan bisa disembunyikan dari console lewat `MPFLSTxx`. Pesan itu tetap tercatat di log, sehingga console lebih bersih tanpa kehilangan jejak. Sebaliknya, pesan yang pernah menjadi awal insiden besar sebaiknya disorot atau diteruskan otomatis ke alert.

## Bab 51. Dump dan IPCS

Dump adalah salinan memori saat masalah terjadi dan merupakan bukti utama saat membuka kasus ke IBM Support. Pastikan dataset dump sistem tersedia dan tidak penuh.

| Jenis dump | Kapan dibuat | Dianalisis dengan |
|---|---|---|
| SYSUDUMP / SYSABEND | Abend program batch, jika DD tersedia | Dibaca langsung atau IPCS |
| SYSMDUMP | Abend program, format mesin | IPCS |
| SVC dump | Diminta sistem atau operator (`DUMP COMM=...`) | IPCS |
| Stand-alone dump (SADMP) | Sistem hang total, di-IPL dari HMC | IPCS |
| Transaction dump CICS | Abend transaksi CICS | IPCS dengan verb CICS |

Perintah IPCS yang paling sering dipakai:

```
STATUS                          Ringkasan kondisi saat dump
VERBX LOGDATA                   Entri LOGREC di dalam dump
SUMMARY FORMAT                  Ringkasan per address space
ANALYZE RESOURCE                Contention sumber daya
IP SYSTRACE ALL                 Trace sistem
VERBX MTRACE                    Pesan terakhir di console
```

Untuk masalah yang sulit ditangkap, pasang SLIP trap atas arahan IBM Support, misalnya `SLIP SET,ID=X001,COMP=0C4,JOBNAME=ACRBUNGA,ACTION=SVCD,END`.

## 51.1 Analisis: Triase Dump yang Efisien

Dump bisa berukuran gigabyte, dan membacanya tanpa urutan yang jelas bisa menghabiskan berjam-jam. Triase yang baik bertujuan menjawab tiga pertanyaan secepat mungkin: apa yang gagal, di mana, dan apakah ini masalah yang sudah dikenal.


*Gambar 51.1 — Triase dump dalam 15 menit*

Urutan perintah IPCS yang umum dipakai untuk triase awal:

```
SETDEF DSNAME('SYS1.DUMP.D260915.T021533.SYSA.S00012')
NOCONFIRM
LIST TITLE                  Judul dump: siapa yang meminta
dan mengapa
STATUS FAILDATA             Kode abend, modul, offset, PSW
saat gagal
STATUS WORKSHEET            Ringkasan kondisi sistem
VERBEXIT LOGDATA            Catatan LOGREC di sekitar waktu
kejadian
SYSTRACE                    Jejak sistem sebelum kegagalan
SUMMARY FORMAT              Detail address space dan TCB
WHERE alamat                Modul pemilik suatu alamat
memori
```

Sebelum menganalisis, pastikan dump-nya utuh. Pesan `IEA611I` menyatakan apakah dump lengkap atau parsial. Dump parsial sering terjadi karena ruang dataset dump tidak cukup, sehingga bagian penting mungkin hilang. Jika ini sering terjadi, perbesar dataset dump atau aktifkan alokasi otomatis dengan `DUMPDS`.

| Keputusan setelah triase | Kapan |
|---|---|
| Perbaiki sendiri | Penyebab jelas di kode aplikasi atau konfigurasi internal |
| Cari di basis pengetahuan IBM | Modul milik IBM; gunakan kombinasi kode abend, modul, dan offset sebagai kata kunci |
| Buka case ke IBM | Modul IBM dan tidak ada APAR yang cocok; sertakan dump utuh dan kronologi |
| Buka tiket ke vendor | Modul milik produk pihak ketiga |

Kombinasi kode abend, nama modul, dan offset (misalnya “S0C4 di modul XYZ offset +1A2”) adalah kata kunci yang paling efektif untuk mencari masalah yang sudah dikenal. Banyak kasus selesai di sini tanpa perlu membaca dump lebih dalam, karena perbaikannya sudah tersedia sebagai PTF (Bab 176).

# Bab 52. Katalog Runbook

Setiap runbook ditulis dengan pola yang sama: gejala, diagnosis, tindakan, dan pencegahan. Uji setiap runbook di lingkungan non-produksi sebelum dipakai.

## RB-001: Storage group produksi melewati ambang kritis

- **Gejala:** peringatan pemakaian SG di atas 85%, job mulai SB37.
- **Diagnosis:** `D SMS,SG(SGPROD),LISTVOL`; cari dataset terbesar yang baru dibuat lewat DSLIST atau laporan DCOLLECT.
- **Tindakan:** jalankan migrasi HSM untuk data tidak aktif; hapus GDG usang sesuai retensi; tambah volume ke SG.
- **Pencegahan:** alert di 75%, laporan tren kapasitas bulanan.

## RB-002: Spool JES2 hampir penuh

- **Gejala:** `$HASP050`, job baru tidak bisa masuk.
- **Diagnosis:** `$D SPOOL`, lalu cari job dengan output raksasa di SDSF ST (urutkan kolom baris/record).
- **Tindakan:** batalkan job yang loop; purge output lama yang sudah diarsip; tambahkan volume spool dengan `$S SPOOL(vol)`.
- **Pencegahan:** arsip output otomatis dan kebijakan purge per class.

## RB-003: Job EOD menunggu dataset (ENQ contention)

- **Gejala:** job berstatus menunggu, pesan `IEF863I` atau `IEF099I` tentang dataset tidak tersedia.
- **Diagnosis:** `D GRS,RES=(SYSDSN,dataset)` untuk melihat pemegang ENQ.
- **Tindakan:** jika pemegang adalah region CICS, tutup file lewat `CEMT S FILE(...) CLO`; jika job lain, koordinasikan dengan pemiliknya.
- **Pencegahan:** langkah penutupan file CICS eksplisit di awal fase EOD.

### RB-004: Deadlock atau timeout Db2 saat EOD

- **Gejala:** `DSNT375I` atau `DSNT376I`, SQLCODE -911 atau -913.
- **Diagnosis:** identifikasi dua pihak dari pesan; periksa apakah transaksi online masih berjalan.
- **Tindakan:** restart job dari step aman; jika berulang, pisahkan jadwal atau tambah commit frequency.
- **Pencegahan:** commit berkala di program batch, urutan akses tabel yang konsisten.

### RB-005: EOD gagal karena S0C7

- **Gejala:** step akrual abend `S0C7`.
- **Diagnosis:** dari dump atau listing compile, cari offset dan field; cek record input di posisi tersebut.
- **Tindakan:** koreksi record data melalui prosedur yang disetujui; restart dari step akrual.
- **Pencegahan:** validasi input di step awal, bukan di tengah perhitungan.

### RB-006: Replikasi ke situs DR tertinggal

- **Gejala:** alert GDPS, RPO Global Mirror melebar.
- **Diagnosis:** periksa bandwidth link replikasi dan beban tulis saat EOD.
- **Tindakan:** eskalasi ke tim storage dan jaringan; pastikan tidak ada suspend pada pasangan volume.
- **Pencegahan:** kapasitas link dirancang untuk puncak tulis EOD akhir bulan.

### RB-007: Kekurangan auxiliary storage

- **Gejala:** `IRA200E` atau `IRA201E`.
- **Diagnosis:** `D ASM`; cari address space dengan pemakaian memori tumbuh tak wajar.

- **Tindakan:** tambahkan page dataset dengan `PAGEADD`; batalkan address space yang bocor jika aman.
- **Pencegahan:** page dataset di bawah 30% pemakaian pada kondisi normal.

### RB-008: Backup malam gagal

- **Gejala:** job backup RC di atas 4 atau abend.
- **Diagnosis:** baca pesan ADR di SYSPRINT; periksa ketersediaan tape atau target FlashCopy.
- **Tindakan:** jalankan ulang untuk dataset yang gagal sebelum EOD berikutnya; laporkan jika backup tidak lengkap.
- **Pencegahan:** verifikasi otomatis jumlah dataset yang dibackup dibandingkan hari sebelumnya.

### RB-009: Satu sistem sysplex tidak merespons

- **Gejala:** `IXC402D` atau SFM mulai mengisolasi sistem.
- **Diagnosis:** periksa sistem bermasalah di HMC (Operating System Messages, aktivitas LPAR).
- **Tindakan:** biarkan SFM mengisolasi sesuai policy; jika manual, lakukan system reset lewat HMC lalu jawab WTOR sesuai prosedur; pastikan DVIPA dan beban CICS pindah.
- **Pencegahan:** SFM policy teruji dan ARM untuk restart subsistem.

### RB-010: Banyak user teller terkunci bersamaan

- **Gejala:** keluhan massal dari cabang, `ICH408I` atau pesan revoke.
- **Diagnosis:** `SEARCH CLASS(USER) REVOKE`; cari pola: satu cabang, satu aplikasi, atau serangan tebak kata sandi.
- **Tindakan:** jika bukan serangan, `ALTUSER ... RESUME` terkontrol; jika serangan, eskalasi ke tim keamanan dulu.
- **Pencegahan:** pemantauan SMF tipe 80 secara real-time ke SIEM.

### RB-011: IPL tidak selesai

- **Gejala:** sistem berhenti di NIP atau menunggu prompt.
- **Diagnosis:** baca *Operating System Messages* di HMC; perhatikan wait state code (misalnya `wait 064`, `wait 0A2`).
- **Tindakan:** IPL ulang dengan LOADxx atau SYSRES terakhir yang terbukti baik; bandingkan perubahan PARMLIB.
- **Pencegahan:** selalu simpan konfigurasi "last known good" dan uji perubahan di LPAR sandbox.

### RB-012: Kanal digital gagal karena sertifikat kedaluwarsa

- **Gejala:** handshake TLS gagal serentak, pesan AT-TLS di syslog.
- **Diagnosis:** `RACDCERT ID(xxx) LIST` untuk melihat masa berlaku sertifikat di keyring.
- **Tindakan:** pasang sertifikat baru, `SETROPTS RACLIST(DIGTCERT) REFRESH`, lalu refresh policy AT-TLS.
- **Pencegahan:** inventaris sertifikat dengan alert 60 hari sebelum kedaluwarsa.

## 52.1 Analisis: Templat Runbook yang Konsisten

Katalog runbook yang ditulis banyak orang dalam waktu bertahun-tahun cenderung punya format yang berbeda-beda. Satu runbook dimulai dengan latar belakang panjang, yang lain langsung ke perintah, dan yang lain lagi tidak menyebutkan cara memastikan masalah sudah selesai. Templat yang konsisten membuat petugas jaga tahu di mana mencari setiap informasi, di runbook mana pun.

```
# RB-0xx — <judul singkat: gejala, bukan penyebab>
Terakhir diuji: <tanggal> oleh <nama>        Pemilik:
<tim>

## Pemicu
Pesan, alert, atau gejala yang membawa pembaca ke runbook
ini.

## Dampak dan urgensi
Layanan apa yang terdampak; berapa lama boleh ditunda
```

```
sebelum eskalasi.

## Prasyarat
Hak akses, alat, dan informasi yang dibutuhkan sebelum
mulai.

## Langkah
1. Perintah persis yang dijalankan, dengan hasil yang
diharapkan.
2. Titik keputusan: bila hasil X, lanjut ke langkah 3;
bila Y, ke langkah 6.

## Verifikasi
Bukti bahwa masalah sudah selesai.

## Kembali
Cara membatalkan langkah bila keadaan memburuk.

## Eskalasi
Kapan dan kepada siapa, dengan informasi apa.
```

| Unsur templat | Mengapa penting |
|---|---|
| Judul berdasarkan gejala | Petugas jaga mencari berdasarkan apa yang ia lihat, bukan penyebab yang belum diketahui |
| Tanggal uji dan pemilik di bagian atas | Pembaca tahu seberapa bisa mempercayai isinya (Bab 95.1) |
| Hasil yang diharapkan di setiap langkah | Pembaca tahu apakah langkahnya berhasil sebelum lanjut |
| Titik keputusan eksplisit | Tidak ada tebakan di tengah jalan |
| Bagian kembali | Setiap tindakan punya jalan mundur |

Konversi runbook lama ke templat baru tidak perlu dilakukan sekaligus. Mulailah dari runbook yang paling sering dipakai (Bab 88.1), dan konversi setiap runbook lain saat runbook itu ditinjau atau dipakai berikutnya. Dalam setahun, sebagian besar katalog yang benar-benar dipakai akan sudah seragam.

# 🚀 Bagian XIV — Integrasi dan Modernisasi

## Bab 53. Strategi Modernisasi

Modernisasi mainframe yang berhasil hampir selalu bertahap: buka akses lewat API, perbaiki cara kerja pengembangan, baru ubah aplikasi bila ada alasan bisnis yang jelas. Migrasi total keluar dari mainframe adalah proyek bertahun-tahun dengan risiko tinggi.

| Pendekatan | Isi | Risiko | Kapan cocok |
|---|---|---|---|
| Expose (API) | Bungkus transaksi CICS/Db2 sebagai REST API | Rendah | Kanal digital perlu akses cepat ke core |
| Modernisasi proses (DevOps) | Git, build otomatis, pengujian otomatis | Rendah–menengah | Tim ingin rilis lebih cepat dan aman |
| Refactor | Pecah atau tulis ulang modul tertentu (COBOL modern atau Java di z/OS) | Menengah | Modul sulit dirawat atau sering berubah |
| Replatform sebagian | Pindahkan beban non-kritikal ke Linux on Z atau cloud | Menengah | Reporting, analitik, aplikasi pendukung |
| Replace | Ganti core banking | Tinggi | Keputusan strategis tingkat direksi |

### 53.1 Analisis: Memprioritaskan Portofolio Modernisasi

Modernisasi yang dicoba sekaligus pada semua aplikasi hampir selalu gagal: anggaran habis sebelum hasil terlihat, dan risiko menumpuk. Pendekatan yang lebih sehat memetakan setiap komponen berdasarkan dua sumbu, yaitu nilai bisnis perubahan dan risiko teknisnya, lalu memilih urutan kerja dari peta itu.


*Gambar 53.1 — Matriks prioritas portofolio modernisasi*

| Kuadran | Strategi | Contoh |
|---|---|---|
| Nilai tinggi, risiko rendah | Kerjakan lebih dulu; hasil cepat membangun kepercayaan | API saldo dan mutasi lewat z/OS Connect |
| Nilai tinggi, risiko tinggi | Rencanakan bertahap dengan pola *strangler* | Modul GL, pembayaran real-time |
| Nilai rendah, risiko rendah | Tunda atau kerjakan sambil lalu | Utilitas internal |
| Nilai rendah, risiko tinggi | Pertahankan dan isolasi; dokumentasikan | Program Assembler lama yang stabil |

Penilaian dilakukan bersama. Nilai bisnis dinilai oleh pemilik produk: pendapatan, pengalaman nasabah, dan kebutuhan regulasi. Risiko teknis dinilai oleh tim teknis: ukuran kode, keterkaitan dengan komponen lain, ketersediaan dokumentasi, dan ketersediaan orang yang memahami kodenya.

Pola *strangler* untuk kuadran nilai tinggi dan risiko tinggi bekerja dengan membungkus fungsi lama di balik API, lalu memindahkan logika sedikit demi sedikit ke implementasi baru di balik API yang sama. Pada setiap tahap, sistem lama tetap berjalan sebagai jalan kembali. Modernisasi

berhasil bila setiap tahap menghasilkan nilai sendiri, bukan hanya di akhir proyek.

Peta ini dinilai ulang setiap tahun. Setelah API saldo tersedia, misalnya, komponen lain yang bergantung padanya menjadi lebih mudah diubah, sehingga posisinya bergeser ke kuadran yang lebih menarik.

# Bab 54. API dan Konektivitas

z/OS Connect memetakan transaksi CICS, IMS, program batch, dan Db2 menjadi REST API berbasis OpenAPI. Kanal digital tidak perlu memahami COMMAREA atau copybook COBOL.

| Teknologi | Kegunaan |
|---|---|
| z/OS Connect | REST API untuk CICS, IMS, Db2, MQ; mendukung OpenAPI 3 |
| CICS web services / JSON | Layanan langsung dari region CICS |
| Db2 REST services / JDBC | Akses data langsung, sebagian besar dieksekusi di zIIP |
| IBM MQ | Integrasi asinkron dan andal dengan sistem lain |
| Change Data Capture | Replikasi perubahan data ke data lake atau Kafka |

Prinsip rancangan API di atas core banking sama dengan edisi IBM i: idempotensi untuk transaksi keuangan, batas laju per klien, dan audit setiap panggilan. Letakkan API gateway di depan z/OS Connect, jangan ekspos mainframe langsung ke internet.

## 54.1 Analisis: Merancang API di Atas CICS

Membuka transaksi CICS sebagai REST API itu mudah secara teknis. Merancangnya agar cepat, aman diulang, dan tidak membebani MSU membutuhkan keputusan yang lebih hati-hati.

**Granularitas.** Satu layar mobile banking yang membutuhkan enam panggilan API masing-masing 80 ms menghabiskan sekitar 480 ms, belum

termasuk jaringan seluler. Satu API agregat yang mengambil data yang sama dalam satu transaksi CICS bisa selesai dalam 120 ms. API yang terlalu halus (*chatty*) memperlambat nasabah dan menambah beban CPU karena setiap panggilan membawa overhead TLS, parsing JSON, dan pemanggilan transaksi.

**Rantai timeout.** Setiap lapisan harus menyerah sedikit lebih cepat daripada lapisan di atasnya. Jika tidak, lapisan atas sudah menyerah sementara lapisan bawah masih bekerja, dan hasilnya adalah transaksi yang berhasil di core tetapi dilaporkan gagal ke nasabah.

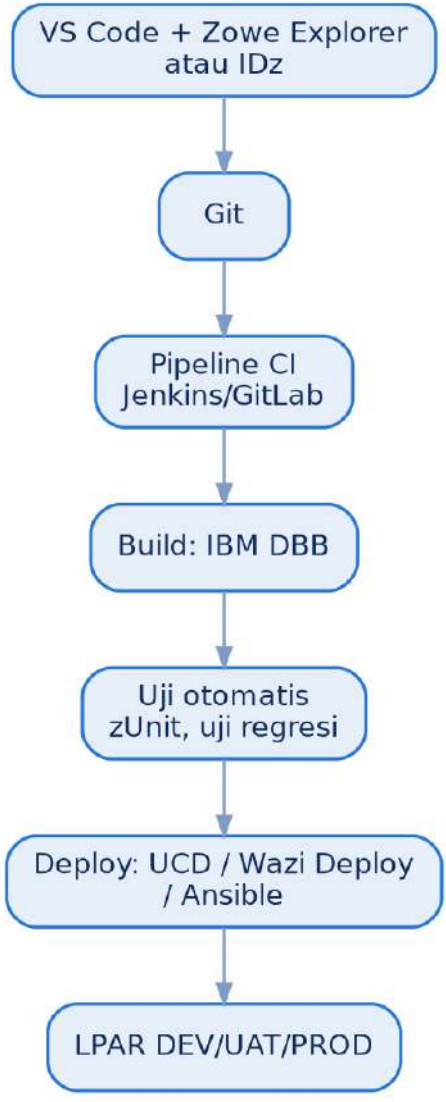
$$T_{\text{klien}} > T_{\text{gateway}} > T_{\text{z/OS Connect}} > T_{\text{CICS}} + T_{\text{margin}}$$

| Keputusan desain | Rekomendasi | Alasan |
|---|---|---|
| Idempotensi | Header kunci idempotensi wajib untuk transaksi keuangan | Retry aman tanpa transaksi ganda (Bab 102) |
| Versi | Versi di path, misalnya `/v1/`, dengan masa transisi | Klien lama tidak rusak saat API berubah |
| Pemetaan error | Abend dan kegagalan sistem menjadi 5xx; penolakan bisnis menjadi 4xx dengan kode jelas | Klien bisa membedakan “coba lagi” dan “jangan coba lagi” |
| Batas laju | Per klien di gateway | Satu mitra tidak bisa menghabiskan kapasitas dan MSU |
| Data sensitif | Hanya field yang dibutuhkan; masking nomor identitas | Mengurangi dampak kebocoran |
| Observabilitas | ID korelasi dari gateway sampai SMF 110 | Satu transaksi dapat dilacak ujung ke ujung (Bab 35.2) |

API yang dirancang baik juga melindungi mainframe. Batas laju di gateway mencegah lonjakan dari satu mitra menaikkan 4HRA, dan API agregat mengurangi jumlah transaksi CICS untuk hasil bisnis yang sama.

# Bab 55. DevOps untuk z/OS

DevOps mainframe modern memakai Git sebagai sumber kebenaran untuk COBOL, JCL, copybook, dan PARMLIB. Build dan deploy dijalankan pipeline, bukan lewat salin manual antar-library.



| Alat | Fungsi |
|---|---|
| Zowe (CLI, Explorer, API ML) | Akses z/OS dari alat modern dan skrip |
| z/OSMF | REST API dan antarmuka web untuk administrasi, workflow, software management |
| IBM Dependency Based Build (DBB) | Build COBOL/PL/I berbasis dependensi dari Git |
| Ansible for IBM Z (`ibm.ibm_zos_core`) | Otomasi operasi: submit job, kelola dataset, perintah console |
| Wazi as a Service / Wazi Sandbox | Lingkungan z/OS untuk pengembangan dan pengujian |
| watsonx Code Assistant for Z | Bantuan AI untuk memahami dan mentransformasi COBOL |

Contoh playbook Ansible sederhana untuk memeriksa spool dan storage group:

```
- name: Cek kesehatan harian z/OS
  hosts: zos_prod
  gather_facts: false
  tasks:
    - name: Pemakaian spool JES2
      ibm.ibm_zos_core.zos_operator:
        cmd: "$D SPOOL"
      register: spool

    - name: Status storage group produksi
      ibm.ibm_zos_core.zos_operator:
        cmd: "D SMS,SG(SGPROD),LISTVOL"
      register: sg

    - name: Tampilkan hasil
      ansible.builtin.debug:
        msg: "{{ spool.content + sg.content }}"
```

## 55.1 Analisis: Mengukur Kinerja DevOps di Mainframe

Perubahan cara kerja perlu diukur agar kemajuannya terlihat dan tidak berhenti di tataran slogan. Empat metrik yang banyak dipakai di industri, dikenal sebagai metrik DORA, dapat diterapkan juga di mainframe.

| Metrik | Definisi | Cara mengukur di mainframe |
|---|---|---|
| Frekuensi deploy | Seberapa sering perubahan masuk produksi | Jumlah paket deploy per bulan dari pipeline |
| Lead time perubahan | Waktu dari commit sampai berjalan di produksi | Selisih waktu commit Git dan deploy produksi |
| Tingkat kegagalan perubahan | Porsi deploy yang memicu insiden atau rollback | Tiket insiden yang terkait dengan deploy |
| Waktu pemulihan | Waktu dari insiden akibat perubahan sampai pulih | Laporan insiden (Bab 57.1) |

$$\text{Tingkat kegagalan perubahan} = \frac{\text{deploy yang memicu insiden atau rollback}}{\text{total deploy}}$$

Tujuannya bukan meniru angka perusahaan teknologi yang deploy puluhan kali sehari. Core banking punya kebutuhan kontrol yang lebih ketat, dan itu wajar. Tujuannya adalah tren: lead time yang turun dari enam pekan menjadi dua pekan, dengan tingkat kegagalan yang tidak naik, adalah kemajuan besar bagi tim mainframe.

Metrik ini juga mengungkap hambatan. Jika lead time panjang tetapi pembangunan kodenya cepat, hambatannya biasanya ada di antrean persetujuan atau jadwal uji. Jika tingkat kegagalan tinggi, cakupan uji otomatis perlu ditambah sebelum frekuensi deploy dinaikkan. Kecepatan tanpa keandalan hanya memindahkan masalah ke petugas jaga.

## Bab 56. IBM Z dan Hybrid Cloud

Posisi realistis IBM Z dalam arsitektur hybrid adalah inti transaksi dan data sensitif, dengan layanan di sekitarnya berjalan di cloud atau Kubernetes. Integrasi dilakukan lewat API, MQ, dan replikasi data, bukan dengan memindahkan core.

- Jalankan Red Hat OpenShift di Linux on IBM Z atau LinuxONE untuk layanan mikro yang dekat dengan data.
- Gunakan HiperSockets atau SMC-D untuk komunikasi latensi rendah antara z/OS dan Linux dalam satu CPC.

- Manfaatkan inferensi AI on-chip (Telum dan Telum II) untuk skor fraud langsung di jalur transaksi.
- Salin data ke platform analitik lewat CDC agar beban query analitik tidak mengganggu OLTP.

## 56.1 Mengukur keberhasilan modernisasi

| Indikator | Contoh target |
|---|---|
| Lead time perubahan | Dari minggu menjadi hari |
| Frekuensi rilis | Dari bulanan menjadi mingguan |
| Kegagalan perubahan | Menurun setelah pengujian otomatis |
| Jumlah API core yang dipakai kanal digital | Meningkat per kuartal |
| MSU per transaksi | Turun atau stabil saat volume naik |

## 56.2 Analisis: Latensi dan Gravitasi Data dalam Hybrid Cloud

Arsitektur hybrid cloud yang menarik di atas kertas bisa lambat dalam praktik, karena setiap panggilan dari cloud ke mainframe melintasi jaringan antar-pusat data. Faktor yang paling menentukan bukan kecepatan mainframe atau cloud, melainkan jumlah perjalanan pulang-pergi.

$$T_{\text{jaringan}} = N_{\text{panggilan berurutan}} \times RTT_{\text{WAN}}$$

Contoh: aplikasi di cloud publik dengan RTT 15 ms ke pusat data bank. Jika satu layar membutuhkan lima panggilan API berurutan ke mainframe, 75 ms hilang hanya untuk perjalanan jaringan, sebelum satu baris kode pun dijalankan. Jika panggilan itu disatukan menjadi satu API agregat (Bab 54.1), biaya jaringannya turun menjadi 15 ms.

Konsep *gravitasi data* menjelaskan mengapa logika yang banyak menyentuh data inti cenderung paling efisien bila dijalankan dekat dengan data itu. Memindahkan logika ke cloud sementara datanya tetap di mainframe sering menghasilkan aplikasi yang lebih lambat dan lebih mahal.

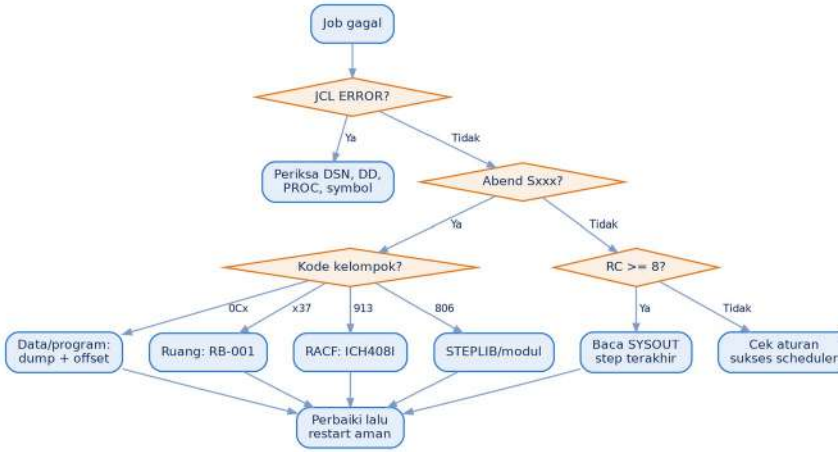
| Jenis beban | Lokasi yang cocok | Alasan |
|---|---|---|
| Logika transaksi yang banyak membaca dan menulis data inti | Dekat data: CICS atau Linux on Z | Menghindari banyak perjalanan jaringan |
| Antarmuka nasabah, konten, kampanye | Cloud | Elastis, cepat berubah, sedikit menyentuh data inti |
| Analitik atas salinan data | Cloud atau platform analitik | Data dapat direplikasi lewat CDC |
| Integrasi dengan mitra | Gateway di tepi jaringan | Titik kontrol keamanan dan batas laju |

Pertimbangkan juga biaya dan keamanan. Transfer data keluar dari cloud publik biasanya dikenai biaya, dan setiap jalur antar-lokasi harus dilindungi dengan TLS dua arah dan koneksi privat. Keputusan arsitektur hybrid sebaiknya diuji dengan prototipe yang mengukur latensi nyata, bukan hanya dengan diagram.

# 🧮 Bagian XV — Algoritma Operasional

Bagian ini merumuskan keputusan operasional yang paling sering diambil menjadi algoritma: diagram alur untuk dibaca cepat saat insiden, dan pseudocode untuk diotomasi dengan REXX, Ansible, atau produk otomasi. Setiap algoritma berakhir pada satu dari tiga hasil: selesai, eskalasi, atau kembali ke langkah aman.

## Bab 57. Algoritma Diagnosis Job Batch Gagal




Klasifikasi pertama selalu memisahkan kegagalan sebelum eksekusi (JCL ERROR) dari kegagalan saat eksekusi. Sebagian besar insiden malam jatuh ke empat kelompok abend di atas.

```
ALGORITMA DiagnosisJobGagal(job)
  log  <- ambil_joblog(job)
  jika log berisi "JCL ERROR" maka
      kembalikan PERIKSA_JCL(pesan IEFxxx pertama)
  step <- step_terakhir_yang_gagal(log)
  jika step.abend dimulai "S0C" maka
      simpan dump; cari offset di listing compile
      kembalikan KOREKSI_DATA_ATAU_PROGRAM
  jika step.abend dalam {SB37, SD37, SE37} maka jalankan
RB-001
  jika step.abend = S913 maka kembalikan ESKALASI_SECURITY
```

```
jika step.abend = S806 maka kembalikan PERIKSA_STEPLIB
jika step.rc >= 8 maka kembalikan BACA_SYSOUT(step)
kembalikan PERIKSA_ATURAN_SCHEDULER
```

## 57.1 Analisis: Menguraikan Waktu Pemulihan

MTTR, rata-rata waktu pemulihan, bukan satu angka yang bisa diperbaiki sekaligus. MTTR adalah jumlah dari beberapa tahap, dan setiap tahap punya cara perbaikan sendiri.

$$MTTR = T_{\text{deteksi}} + T_{\text{diagnosis}} + T_{\text{perbaikan}} + T_{\text{verifikasi}}$$

| Tahap | Contoh durasi insiden job gagal | Cara memendekkan |
|---|---|---|
| Deteksi | 25 menit sampai operator menyadari job merah | Alert otomatis dari scheduler, bukan pengamatan layar |
| Diagnosis | 40 menit membaca log dan mencari penyebab | Algoritma diagnosis (Bab 57), kamus pesan, runbook |
| Perbaikan | 15 menit | Langkah tertulis, perintah yang bisa disalin |
| Verifikasi | 20 menit | Total kontrol otomatis dan pembanding (Bab 61) |
| Total | 100 menit | |

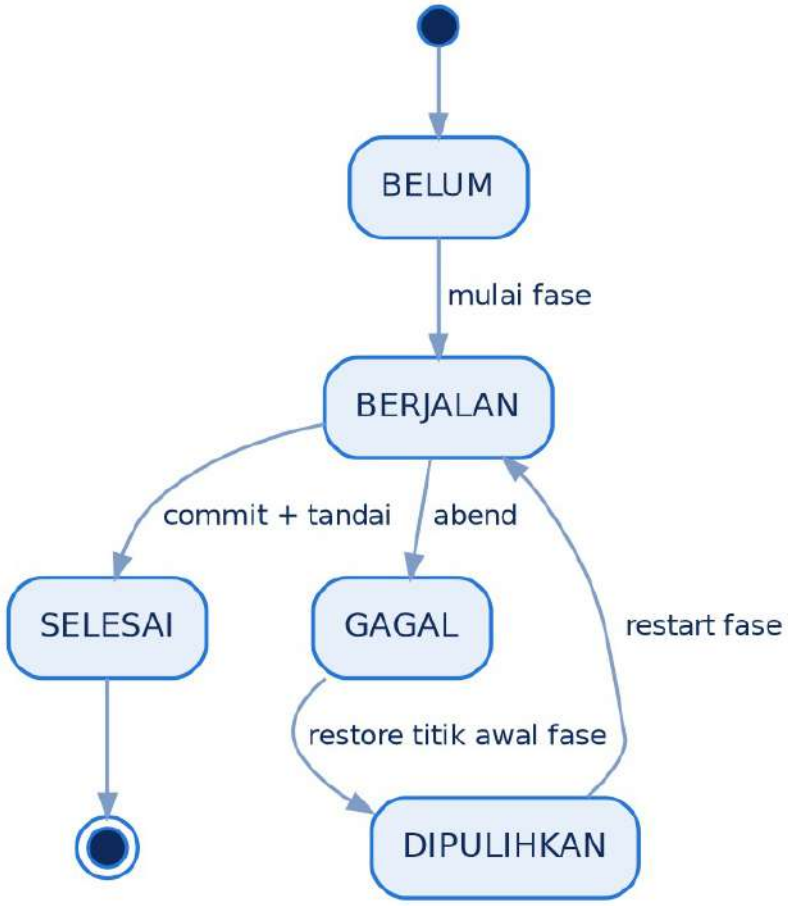
Contoh ini menunjukkan pola yang sering terjadi: perbaikannya sendiri cepat, tetapi deteksi dan diagnosis memakan dua pertiga waktu. Mengotomasi deteksi saja bisa memangkas 20 menit, dan runbook yang baik bisa memotong diagnosis menjadi separuhnya. Totalnya turun dari 100 menit menjadi sekitar 60 menit tanpa mengubah sistem sama sekali.

Kaitkan MTTR dengan ketersediaan di Bab 158. Menurunkan MTTR dari 100 menjadi 60 menit untuk jenis insiden yang terjadi dua kali sebulan menghemat sekitar 16 jam gangguan per tahun. Angka seperti ini membuat investasi dalam otomasi dan dokumentasi mudah dijelaskan kepada manajemen.

Catat keempat komponen waktu di setiap laporan pasca-insiden (Bab 175). Setelah beberapa bulan, pola yang muncul menunjukkan tahap mana yang paling layak diperbaiki lebih dulu.

## Bab 58. Algoritma Restart Fase EOD yang Aman

Restart hanya aman jika setiap fase idempoten atau dapat dikembalikan ke titik awal. Tabel kontrol fase adalah satu-satunya sumber kebenaran tentang fase mana yang sudah selesai.




```
ALGORITMA RestartEOD(tanggal_bisnis)
  untuk setiap fase f dalam urutan_EOD:
```

```
      status <- baca_tabel_kontrol(tanggal_bisnis, f)
      jika status = SELESAI maka lanjut          -- jangan
jalankan dua kali
      jika status = BERJALAN atau GAGAL maka
          jika f.idempoten = TIDAK maka
              restore(f.titik_awal)              --
FlashCopy / image copy
          tandai(f, BERJALAN)
      jalankan(f)
      jika gagal(f) maka tandai(f, GAGAL); berhenti;
eskalasi
      verifikasi_total_kontrol(f)                -- lihat
Bab 61
      tandai(f, SELESAI)
```

Kesalahan paling mahal di EOD adalah menjalankan akrual dua kali. Algoritma ini mencegahnya dengan memeriksa status sebelum eksekusi, bukan sesudahnya.

## 58.1 Analisis: Matriks Uji Restart EOD

Fase EOD yang dirancang aman diulang harus dibuktikan aman. Bukti terbaik adalah uji kegagalan yang disengaja di UAT: fase dihentikan paksa di titik-titik berbeda, lalu dijalankan ulang, dan hasilnya dibandingkan dengan run yang tidak pernah gagal.

Setiap fase diuji di minimal tiga titik kegagalan:

| Titik kegagalan | Cara menyuntikkan | Hasil yang diharapkan setelah restart |
|---|---|---|
| Sebelum fase mulai | Hentikan job sebelum klaim tabel kontrol | Fase berjalan normal sekali |
| Di tengah fase | Batalkan job setelah sebagian commit | Tidak ada record diproses dua kali; total sama dengan run bersih |
| Setelah selesai, sebelum ditandai SELESAI | Batalkan tepat sebelum update status | Fase tidak dijalankan ulang, atau dijalankan ulang tanpa efek ganda |

Jumlah skenario uji minimum:

$$N_{\text{uji}} = N_{\text{fase}} \times N_{\text{titik}} + N_{\text{uji ganda}}$$

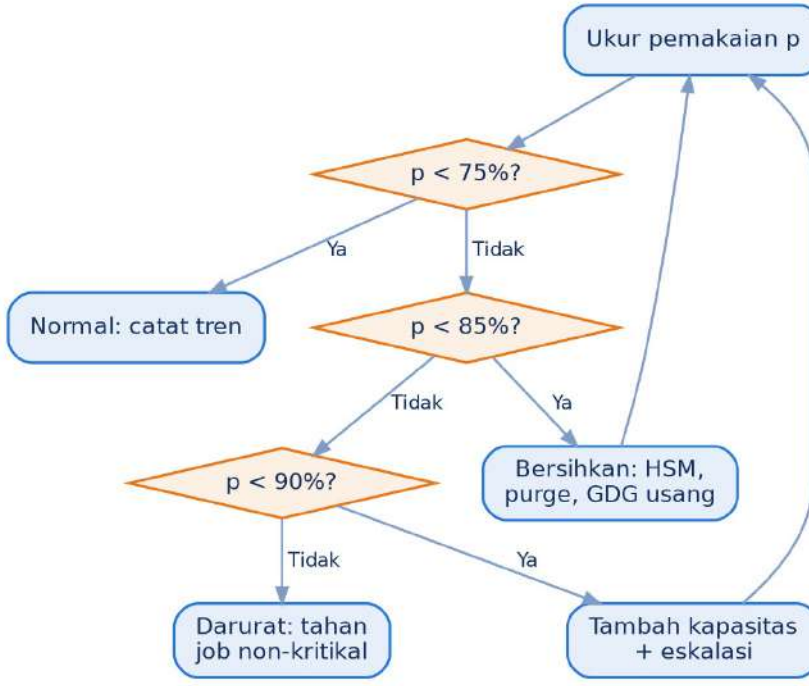
Untuk EOD dengan 8 fase dan 3 titik kegagalan, ditambah 2 uji “dijalankan dua kali tanpa gagal”, dibutuhkan 26 skenario. Angka ini terlihat banyak, tetapi setiap skenario dapat diotomasi dan dijalankan ulang setiap kali program EOD berubah.

```
ALGORITMA UjiRestartFase(fase f, titik t)
  pulihkan data UAT ke snapshot awal yang sama
  jalankan EOD sampai fase f, suntikkan kegagalan di titik
t
  jalankan prosedur restart sesuai runbook (bukan cara
pintas)
  selesaikan EOD
  bandingkan dengan run acuan tanpa kegagalan:
      jumlah record per tabel, total nominal per akun GL,
saldo sampel rekening
  jika ada perbedaan maka catat CACAT pada fase f titik t
```

Uji ini juga menguji runbook-nya. Jika operator UAT kesulitan mengikuti langkah restart, langkah itu akan lebih sulit lagi pukul dua pagi di produksi.

## Bab 59. Algoritma Respons Kapasitas

Algoritma yang sama berlaku untuk storage group, spool JES2, page dataset, zFS, dan kedalaman antrean MQ. Yang berbeda hanya ambang dan tindakannya.




Selain nilai sesaat, ukur laju pertumbuhan. Waktu tersisa sebelum penuh dihitung dengan rumus berikut, dan menjadi dasar keputusan kapan harus menambah kapasitas.

$$T_{\text{penuh}} = \frac{K_{\text{total}} - K_{\text{terpakai}}}{\Delta K / \Delta t}$$

Jika T penuh lebih pendek dari sisa durasi batch EOD malam ini, lompat langsung ke tindakan darurat, apa pun persentasenya.

## 59.1 Analisis: Proyeksi Kapasitas Storage dengan Regresi

Algoritma kapasitas di bab ini memakai rata-rata pertumbuhan sederhana. Untuk keputusan pengadaan yang bernilai besar, proyeksi yang lebih kuat bisa dibuat dengan regresi linear atas data pemakaian harian beberapa bulan terakhir.

$$U(t) = a + b\,t \qquad b = \frac{\sum (t_i - \bar{t})(U_i - \bar{U})}{\sum (t_i - \bar{t})^2} \qquad T_{\text{ambang}} = \frac{U_{\text{ambang}} - U_{\text{sekarang}}}{b}$$

U(t) adalah pemakaian pada hari t, dan b laju pertumbuhan harian hasil regresi. Contoh: storage group produksi saat ini 72%, laju pertumbuhan hasil regresi 0,12% per hari, dan ambang 85%. Waktu tersisa sekitar (85 – 72) / 0,12 ≈ 108 hari.

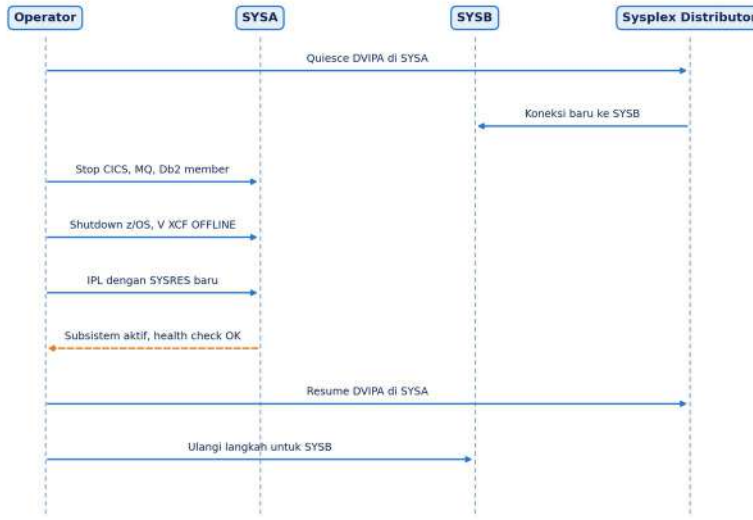
Angka 108 hari baru bermakna bila dibandingkan dengan waktu yang dibutuhkan untuk menambah kapasitas. Jika pengadaan disk membutuhkan 90 hari termasuk persetujuan anggaran, instalasi, dan konfigurasi, keputusan harus diambil sekarang. Menunggu sampai alert 80% berbunyi berarti terlambat.

| Faktor | Pengaruh pada proyeksi |
|---|---|
| Siklus akhir bulan dan akhir tahun | Tambahkan puncak musiman di atas garis tren |
| Proyek migrasi atau produk baru | Tambahkan lonjakan yang direncanakan secara terpisah |
| Pembersihan data atau arsip | Kurangi bila memang terjadwal dan pasti |
| Snapshot dan salinan di array | Hitung kapasitas array terpisah dari storage group (Bab 7.1) |

Sajikan proyeksi bersama rentang ketidakpastiannya, misalnya “antara 95 dan 125 hari”. Pengambil keputusan lebih mudah bertindak bila tahu seberapa yakin angka itu, dan tim teknis terlindungi dari kesan menjanjikan angka pasti.

## Bab 60. Algoritma Rolling IPL dalam Sysplex

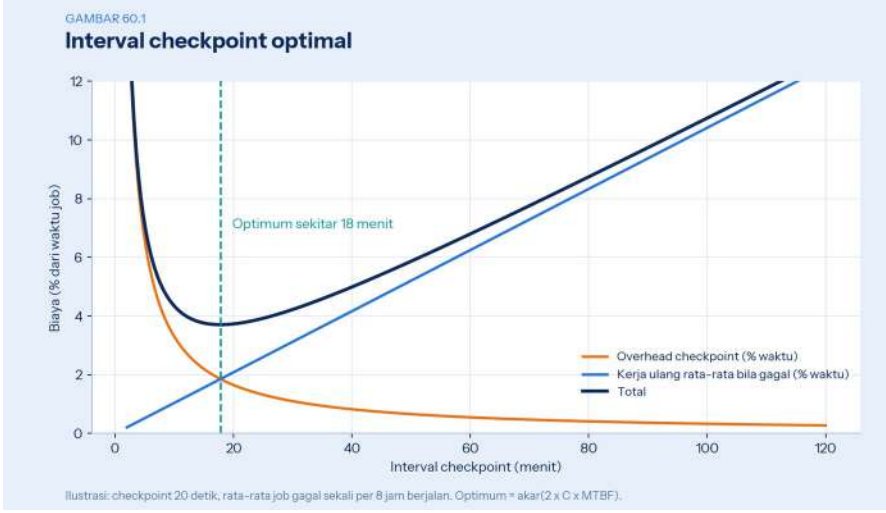
Rolling IPL memperbarui satu sistem pada satu waktu sementara sistem lain terus melayani transaksi. Kuncinya adalah memindahkan beban sebelum mematikan apa pun.




```
ALGORITMA RollingIPL(sysplex, sysres_baru)
  prasyarat: uji di sandbox lulus; backout plan tertulis
  untuk setiap sistem s dalam sysplex (satu per satu):
      pastikan sistem lain sehat dan kapasitas cukup
menampung beban s
      quiesce DVIPA dan routing CICSPlex SM ke s
      tunggu sampai transaksi aktif di s = 0
      shutdown_berurutan(s)                          -- Bab
12.4
      IPL(s, sysres_baru, LOADPARM)
      jalankan health check dan transaksi uji
      jika gagal maka IPL(s, sysres_lama); hentikan
rollout
      resume DVIPA dan routing ke s
      pantau minimal satu jam sebelum sistem berikutnya
```

## 60.1 Analisis: Interval Checkpoint yang Optimal

Checkpoint yang lebih sering memperkecil kerja yang harus diulang bila program gagal, tetapi setiap checkpoint juga memakan waktu. Keduanya bisa diseimbangkan dengan rumus klasik yang dikenal sebagai pendekatan Young.




*Gambar 60.1 — Interval checkpoint optimal*

$$T_{\text{optimal}} \approx \sqrt{2 \times C \times M}$$

C adalah waktu yang dibutuhkan satu checkpoint, dan M rata-rata waktu berjalan sampai program gagal. Dalam ilustrasi, checkpoint memakan 20 detik dan program rata-rata gagal sekali setiap 8 jam berjalan. Interval optimalnya sekitar 18 menit. Pada interval itu, total biaya, yaitu overhead checkpoint ditambah rata-rata kerja yang diulang, berada di titik terendah.

| Interval | Akibat |
|---|---|
| Jauh lebih pendek dari optimal | Waktu terbuang untuk checkpoint yang jarang dibutuhkan |
| Sekitar optimal | Keseimbangan terbaik |
| Jauh lebih panjang dari optimal | Kegagalan berarti mengulang kerja berjam-jam, dan window EOD bisa terlewat |

Rumus ini memberi titik awal, bukan angka pasti. Dua pertimbangan praktis sering menggeser keputusan:

- **Batas window.** Jika kerja ulang setelah kegagalan tidak boleh melebihi slack jalur kritis (Bab 22.1), interval checkpoint harus cukup pendek untuk menjamin itu, walaupun sedikit di bawah optimal.

- **Kaitannya dengan commit.** Di program Db2, checkpoint biasanya dilakukan bersama commit. Interval checkpoint dan frekuensi commit (Bab 70.3) harus diputuskan bersama, karena commit terlalu jarang juga berarti lock ditahan terlalu lama.

Nilai M diambil dari riwayat kegagalan job yang sebenarnya, bukan dari tebakan. Job yang hampir tidak pernah gagal bisa memakai interval yang lebih panjang. Job yang membaca file interface dari mitra dan sering gagal karena data kotor sebaiknya memakai interval yang lebih pendek, atau lebih baik lagi, validasi input di awal (Bab 77.2).

## Bab 61. Algoritma Verifikasi Pasca-EOD

EOD dinyatakan selesai hanya jika angka-angkanya cocok, bukan hanya karena semua job RC 0. Verifikasi memakai total kontrol yang dihitung independen dari program yang diverifikasi.

Persamaan dasar keseimbangan buku besar:

$$\sum_{i=1}^{n} D_i - \sum_{j=1}^{m} K_j = 0$$

Persamaan kesinambungan saldo per rekening dari hari ke hari:

$$S_t = S_{t-1} + \sum \text{mutasi kredit}_t - \sum \text{mutasi debit}_t + \text{bunga}_t - \text{biaya}_t$$

```
ALGORITMA VerifikasiEOD(tanggal)
  cek1 <- total_debit_GL(tanggal) =
total_kredit_GL(tanggal)
  cek2 <- jumlah_transaksi_diposting =
jumlah_transaksi_diterima - ditolak
  cek3 <- untuk sampel rekening: S_t = S_(t-1) + mutasi +
bunga - biaya
  cek4 <- total_akrual(tanggal) dalam toleransi terhadap
proyeksi
  cek5 <- tanggal_sistem = tanggal + 1 hari kerja
  jika semua cek benar maka buka_online(); catat bukti
verifikasi
  selain itu: tahan pembukaan, eskalasi ke App Support dan
Akuntansi
```

## 61.1 Analisis: Total Kontrol dan Hash Total

Verifikasi hasil batch bergantung pada angka pembanding yang bisa dihitung di kedua sisi: sisi pengirim dan sisi penerima, atau sebelum dan sesudah proses. Tiga jenis total kontrol saling melengkapi.

| Total | Cara menghitung | Mendeteksi |
|---|---|---|
| Jumlah record | Hitung record | Record hilang atau tergandakan |
| Total nominal | Jumlahkan field nominal | Nilai berubah atau hilang |
| Hash total | Jumlahkan field yang biasanya tidak dijumlahkan, misalnya nomor rekening | Record tertukar ke rekening lain meskipun jumlah dan nominal sama |

$$H = \left( \sum_{i=1}^{n} \text{nomor rekening}_i \right) \bmod 10^k$$

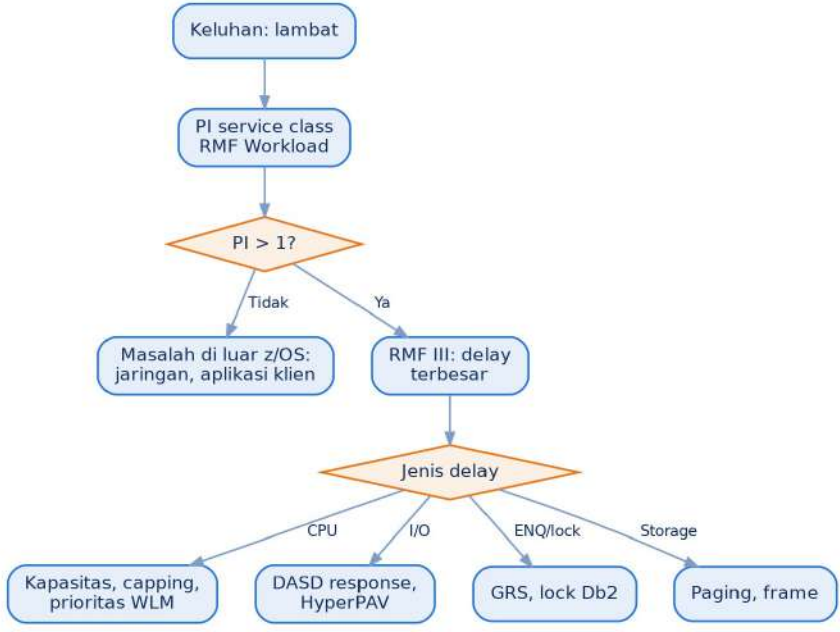
Hash total tidak punya arti bisnis. Nilainya dipakai semata sebagai sidik jari. Jika dua transaksi dengan nominal sama tertukar rekening tujuannya, jumlah record dan total nominal tetap cocok, tetapi hash total berubah. Modulo dengan pangkat sepuluh menjaga angkanya tetap dalam panjang field yang wajar.

**Contoh.** File transfer masuk berisi 120.000 record dengan total Rp48,2 miliar dan hash total 7.315.902.144. Setelah diposting, laporan posting mencatat 120.000 record dan total yang sama, tetapi hash total 7.315.902.136. Selisih kecil itu menunjukkan ada record yang diposting ke rekening berbeda dari file asli, meskipun nominalnya sama. Tanpa hash total, kesalahan ini baru ditemukan saat nasabah mengeluh.

Total kontrol bisa dihitung dengan DFSORT atau ICETOOL tanpa program khusus (Bab 20.1), lalu dibandingkan otomatis dengan trailer file (Bab 147). Simpan ketiga total untuk setiap file dan setiap fase EOD. Riwayat ini juga berguna sebagai bukti audit bahwa pemrosesan berjalan lengkap.

# Bab 62. Algoritma Analisis Kinerja Top-Down

Analisis kinerja yang efisien dimulai dari tujuan bisnis, lalu turun ke sumber daya. Memulai dari metrik CPU mentah sering membuat tim memperbaiki hal yang tidak berdampak.




Delay terbesar menunjukkan di mana waktu habis. Perbaiki satu penyebab, ukur ulang PI, lalu ulangi sampai tujuan tercapai.

## 62.1 Analisis: Contoh Lengkap Analisis Top-Down

Algoritma di bab ini lebih mudah dipahami dengan contoh utuh dari keluhan sampai perbaikan. Contoh berikut menggabungkan beberapa analisis dari bab-bab sebelumnya.

**Keluhan.** Pukul 10.30, beberapa cabang melaporkan transaksi teller lambat. Tidak ada alert yang berbunyi.

| Langkah | Data | Temuan |
|---|---|---|
| 1. PI per service class (Bab | `CICSTLR` PI 1,6 sejak pukul | Masalah terbatas pada |

| Langkah | Data | Temuan |
|---|---|---|
| 168) | 10.05; service class lain normal | transaksi teller |
| 2. Pecah per dimensi (Bab 119.1) | Semua wilayah terdampak, tidak hanya satu | Bukan masalah klasifikasi WLM satu wilayah |
| 3. Delay RMF Monitor III (Bab 41.2) | Delay device 35%, jauh di atas biasanya 8% | Transaksi menunggu I/O |
| 4. Waktu respons per volume (Bab 6.3) | Volume `PRD101` 1,1 ms, didominasi IOSQ | Antrean di host untuk satu volume |
| 5. Isi volume | File master CIF dan satu dataset baru yang dibuat pagi ini | Beban tambahan di volume yang sama |
| 6. Perubahan terakhir (Bab 191.1) | Job laporan baru dijadwalkan mulai pukul 10.00, membaca file CIF secara sequential | Pemicu masalah |

**Perbaikan jangka pendek.** Job laporan ditahan (`$H`), dan dalam beberapa menit PI `CICSTLR` kembali di bawah 1.

**Perbaikan jangka panjang.**

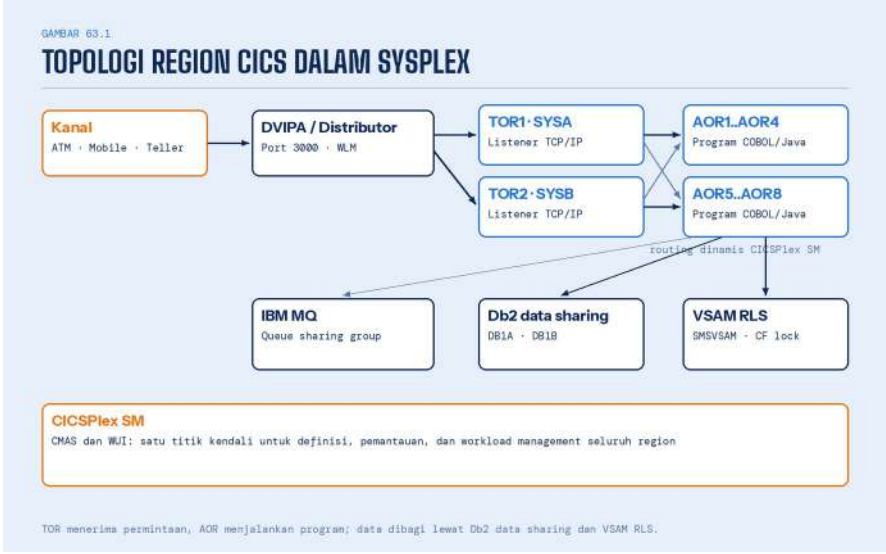
1. Job laporan dipindahkan ke luar jam sibuk, atau diarahkan membaca salinan data (Bab 56.2).
2. Alias HyperPAV untuk volume yang menampung file master ditinjau kembali.
3. Aturan baru: job baru yang membaca file master online wajib ditinjau tim kinerja sebelum dijadwalkan di jam kerja.
4. Alert baru: PI `CICSTLR` di atas 1 selama tiga interval berturut-turut (Bab 185.1).

Contoh ini memperlihatkan pola yang berulang di banyak insiden kinerja. Gejala terlihat di satu tempat (transaksi teller), penyebab langsung di tempat lain (antrean I/O satu volume), dan pemicunya di tempat ketiga (job laporan baru). Pendekatan top-down menghubungkan ketiganya dalam kurang dari setengah jam, tanpa menebak.

# 🧩 Bagian XVI — Pendalaman CICS

## Bab 63. Arsitektur Region dan Topologi

CICS yang sehat di lingkungan bank dibangun sebagai banyak region kecil yang saling mengisi, bukan satu region raksasa. Pemisahan peran membuat satu region bisa di-restart tanpa menghentikan layanan.




*Gambar 63.1 — Topologi region CICS dalam sysplex*

### 63.1 Komunikasi antar-region

| Mekanisme | Jangkauan | Kegunaan |
|---|---|---|
| MRO (Multi-Region Operation) | Region dalam satu z/OS atau sysplex lewat XCF | Function shipping, DPL, transaction routing |
| IPIC (IP Interconnectivity) | Antar-region lewat TCP/IP | Pengganti modern ISC/LU6.2, mendukung channel dan container |
| ISC over SNA | Antar-sistem lewat VTAM | Warisan; hindari untuk desain baru |
| CICSPlex SM dynamic routing | Seluruh region yang dikelola | Menyeimbangkan transaksi berdasarkan kesehatan dan |

| Mekanisme | Jangkauan | Kegunaan |
|---|---|---|
| | | WLM |

## 63.2 Pola penamaan yang disarankan

| Elemen | Pola | Contoh |
|---|---|---|
| APPLID | C + peran + lingkungan + nomor | `CTPRD01` (TOR produksi), `CAPRD05` (AOR) |
| Jobname | Sama dengan APPLID | `CAPRD05` |
| Group CSD | Aplikasi + lapisan | `BNKCIF`, `BNKDEP`, `BNKSYS` |
| Transaksi | 4 karakter, prefix per modul | `CI01` (CIF), `DP10` (deposito) |

## 63.3 Analisis: Memilih Topologi Region CICS

Topologi CICS menentukan seberapa besar dampak satu region yang bermasalah dan seberapa mudah kapasitas ditambah. Tidak ada satu topologi yang cocok untuk semua bank, tetapi pilihannya dapat dianalisis dengan beberapa kriteria.

| Topologi | Kelebihan | Kekurangan | Cocok untuk |
|---|---|---|---|
| Satu region serba bisa | Sederhana | Satu masalah menghentikan semua layanan; tidak bisa diskalakan | Lingkungan uji kecil |
| TOR dan beberapa AOR | Beban dibagi; AOR bisa di-restart bergantian | Konfigurasi routing lebih rumit | Sebagian besar core banking |
| AOR per kelompok aplikasi | Masalah satu aplikasi tidak menjalar ke aplikasi lain | Lebih banyak region untuk dikelola | Aplikasi dengan profil risiko berbeda |
| AOR tersebar di beberapa sistem sysplex | Bertahan saat satu sistem berhenti | Butuh data sharing atau RLS | Layanan dengan target ketersediaan tinggi |

Dengan workload routing dari CICSPlex SM, transaksi diarahkan ke AOR yang sehat dan tidak terlalu sibuk. Bila satu AOR berhenti, porsi kapasitas yang hilang kira-kira:

$$\text{Kapasitas hilang} \approx \frac{1}{N_{\text{AOR}}}$$

Dengan empat AOR, satu AOR yang berhenti mengurangi kapasitas sekitar 25%. AOR yang tersisa harus sanggup menampungnya, dengan logika yang sama seperti cadangan N+1 sysplex (Bab 11.1). Dengan dua AOR, satu AOR yang berhenti berarti separuh kapasitas hilang.

Pemisahan aplikasi ke AOR berbeda juga melindungi layanan penting dari aplikasi yang kurang stabil. Aplikasi baru yang belum matang sebaiknya tidak ditempatkan di AOR yang sama dengan transaksi teller dan transfer. Satu kebocoran storage di aplikasi baru dapat membuat seluruh AOR mengalami SOS, dan semua transaksi di dalamnya ikut terdampak.

Dokumentasikan alasan setiap keputusan topologi: aplikasi apa di region mana, dan mengapa. Topologi yang tumbuh tanpa catatan sulit dirapikan di kemudian hari.

## Bab 64. Definisi Resource dan Parameter Sistem

Setiap program, transaksi, file, dan koneksi di CICS harus didefinisikan sebagai resource. Definisi disimpan di CSD dan dimuat saat region start melalui daftar group (`GRPLIST`).

| Cara mendefinisikan | Kapan dipakai |
|---|---|
| `DFHCSDUP` (batch) | Promosi terkontrol dari pipeline; paling mudah diaudit |
| `CEDA` (online) | Pengembangan dan perbaikan darurat |
| CICSPlex SM BAS | Definisi terpusat untuk banyak region |
| CICS bundle | Aplikasi modern: definisi dikemas bersama kode |

Contoh `DFHCSDUP` untuk menambah program dan transaksi:

```
//CSDUPD   EXEC PGM=DFHCSDUP
//STEPLIB  DD DISP=SHR,DSN=CICSTS.SDFHLOAD
//DFHCSD   DD DISP=SHR,DSN=CICS.PROD.DFHCSD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE PROGRAM(DPINQ01) GROUP(BNKDEP) LANGUAGE(COBOL)
         CONCURRENCY(THREADSAFE) DATALOCATION(ANY)
  DEFINE TRANSACTION(DP10) GROUP(BNKDEP) PROGRAM(DPINQ01)
         TWASIZE(0) TASKDATALOC(ANY)
  ADD GROUP(BNKDEP) LIST(BNKLIST1)
/*
```

## 64.1 Parameter SIT yang paling berpengaruh

| Parameter | Fungsi | Catatan |
|---|---|---|
| APPLID | Nama region di jaringan | Unik per sysplex |
| GRPLIST | Daftar group CSD yang dimuat | Pisahkan group sistem dan aplikasi |
| MXT | Batas task bersamaan | Terlalu rendah memicu antrean; terlalu tinggi memicu SOS |
| EDSALIM | Batas DSA di atas 16 MB | Sesuaikan dengan REGION job |
| SEC=YES | Aktifkan keamanan RACF | Wajib di produksi |
| XTRAN, XPCT, XFCT | Pemeriksaan keamanan per jenis resource | Aktifkan sesuai kebijakan |
| AUXTR, TRTABSZ | Auxiliary trace dan ukuran trace internal | Trace internal yang cukup besar memudahkan diagnosis |
| ICVR | Batas runaway task (ms) | Pemicu abend AICA |
| DB2CONN=YES | Sambungkan ke Db2 saat start | Periksa definisi DB2CONN |
| SIT= dan override | Memilih tabel SIT | Simpan override di dataset SYSIN, bukan di PARM |

## 64.2 Analisis: Memperlakukan CSD sebagai Konfigurasi

CSD menyimpan definisi resource CICS: program, transaksi, file, koneksi, dan banyak lagi. Karena itu CSD adalah konfigurasi, dan harus dikelola seperti konfigurasi lain: terkendali, dapat dibandingkan, dan dapat dikembalikan.

Masalah yang sering muncul adalah perbedaan definisi antara lingkungan. Transaksi yang berjalan baik di UAT gagal di produksi karena satu atribut berbeda, misalnya definisi file yang di UAT boleh diperbarui tetapi di produksi hanya boleh dibaca. Atau situs DR memakai CSD yang tertinggal beberapa rilis.

| Praktik | Cara | Manfaat |
|---|---|---|
| Kelompokkan per aplikasi | Satu group CSD per aplikasi, disusun dalam list | Perubahan terisolasi per aplikasi |
| Ekstrak berkala | Utilitas `DFHCSDUP` dengan perintah `LIST` atau `EXTRACT` | Definisi dalam bentuk teks yang bisa dibandingkan |
| Simpan di Git | Hasil ekstrak disimpan dan dibandingkan per rilis | Riwayat perubahan dan jalan kembali |
| Bandingkan antar lingkungan | Diff UAT, produksi, dan DR | Drift ditemukan sebelum menjadi insiden (Bab 15.1) |
| Promosi lewat pipeline | Definisi diterapkan dari Git, bukan diketik ulang | Konsistensi antar lingkungan |

```
ALGORITMA BandingkanCSD(lingkungan_list)
  untuk setiap lingkungan e: ekstrak semua group aplikasi
dengan DFHCSDUP
  normalisasi: urutkan atribut, hilangkan atribut yang
memang berbeda per lingkungan
  bandingkan UAT vs PROD dan PROD vs DR per group
  laporkan perbedaan yang tidak ada di daftar pengecualian
  perbedaan PROD vs DR diperlakukan sebagai temuan
prioritas tinggi
```

Beberapa atribut memang wajar berbeda per lingkungan, misalnya nama dataset UAT dan produksi. Daftar pengecualian yang eksplisit membuat laporan perbandingan tetap bersih, sehingga perbedaan yang tidak disengaja langsung terlihat.

Perubahan CSD langsung di produksi lewat transaksi CEDA seharusnya menjadi pengecualian darurat. Setelah darurat selesai, perubahan itu wajib dimasukkan kembali ke sumber di Git, agar rilis berikutnya tidak menimpanya tanpa sengaja.

# Bab 65. Storage, TCB, dan Kinerja

Kinerja CICS ditentukan tiga hal: berapa banyak task yang boleh berjalan, di TCB mana task berjalan, dan cukup tidaknya storage. Ketiganya saling memengaruhi.

## 65.1 Dynamic Storage Area

| DSA | Lokasi | Isi |
|---|---|---|
| CDSA / UDSA | Di bawah 16 MB | Program dan data 24-bit lama |
| ECDSA / EUDSA | Di atas 16 MB | Sebagian besar program COBOL dan data task |
| ERDSA | Di atas 16 MB, read-only | Program reentrant |
| GCDSA / GUDSA | Di atas 2 GB (64-bit) | Channel, container, dan fungsi modern |

SOS terjadi ketika DSA tidak bisa memenuhi permintaan storage. CICS lalu memperlambat penerimaan task baru, sehingga dari luar terlihat seperti “CICS lambat”.

## 65.2 Mode TCB

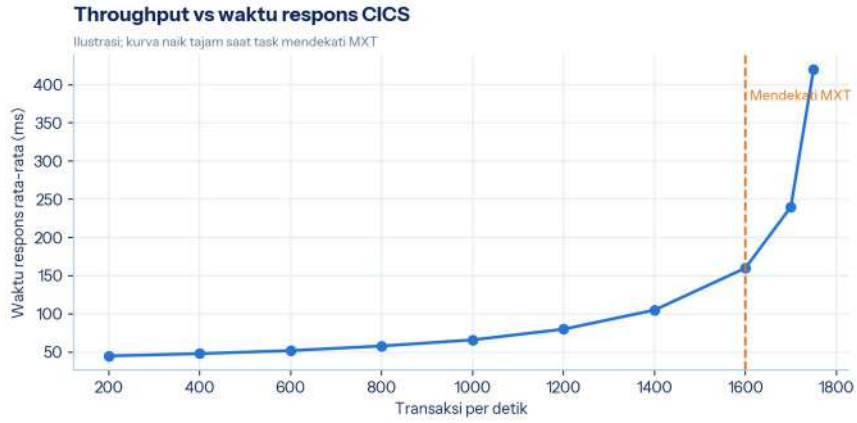
| TCB | Fungsi |
|---|---|
| QR (quasi-reentrant) | TCB utama; semua program non-threadsafe berjalan di sini secara bergiliran |
| L8 / L9 | Open TCB untuk program threadsafe dan panggilan Db2 |
| T8 | Layanan JVM dan web |
| SO / SL | Socket dan listener TCP/IP |

| TCB | Fungsi |
|---|---|
| RO / FO | Operasi file dan read-only |

Program non-threadsafe yang memanggil Db2 harus berpindah dari QR ke L8 dan kembali lagi di setiap panggilan. Menjadikan program threadsafe menghapus perpindahan itu dan mengurangi antrean di QR.

## 65.3 Throughput dan waktu respons

Waktu respons CICS relatif datar sampai jumlah task mendekati MXT, lalu melonjak tajam. Target kapasitas yang aman adalah menjaga puncak harian di bawah sekitar 80% titik lutut kurva ini.




*Ilustrasi · data rekaan*

```
F CAPRD05,CEMT I SYS                  Nilai MXT dan status
sistem
F CAPRD05,CEMT I DSAS                 Pemakaian dan status
SOS setiap DSA
F CAPRD05,CEMT I TASK                 Task aktif dan status
tunggunya
F CAPRD05,CEMT S SYS MAXTASKS(250)    Ubah MXT secara
dinamis
F CAPRD05,CEMT P STAT                 Tulis statistik
sekarang
```

## 65.4 Analisis: Menjadikan Program Threadsafe

Program yang tidak threadsafe berjalan di QR TCB, satu TCB utama yang hanya bisa memakai satu prosesor pada satu waktu. Ketika program itu memanggil Db2, CICS berpindah ke open TCB untuk panggilan Db2, lalu kembali ke QR. Setiap perpindahan TCB memakan CPU. Program yang threadsafe dapat tetap berjalan di open TCB, menghindari perpindahan bolak-balik dan melepaskan batas QR (Bab 32.3).

| Pengaturan program | Perilaku | Cocok untuk |
|---|---|---|
| `CONCURRENCY(QUASIRENT)` | Selalu kembali ke QR setelah panggilan Db2 | Program lama yang belum diperiksa |
| `CONCURRENCY(THREADSAFE)` | Tetap di open TCB setelah panggilan Db2 bila memungkinkan | Program yang sudah diperiksa aman |
| `CONCURRENCY(REQUIRED)` | Selalu berjalan di open TCB sejak awal | Program yang sepenuhnya threadsafe dan banyak memanggil Db2 |

**Daftar periksa sebelum menandai program threadsafe:**

1. Program tidak memperbarui storage bersama (misalnya CWA atau shared storage) tanpa serialisasi. Di QR, hanya satu task yang berjalan pada satu waktu, sehingga akses tanpa kunci kebetulan aman. Di open TCB, beberapa task bisa berjalan bersamaan.
2. Program tidak memanggil exit atau program lain yang belum threadsafe.
3. Perintah CICS yang dipakai program sudah threadsafe di rilis CICS yang berjalan.
4. Program diuji dengan beban bersamaan yang tinggi di lingkungan uji, bukan hanya uji fungsi satu pengguna.

Perubahan ke threadsafe paling menguntungkan untuk program yang banyak memanggil Db2. Ukur jumlah perpindahan TCB per transaksi dari statistik CICS sebelum dan sesudah perubahan, beserta CPU per transaksi. Penurunan perpindahan TCB yang besar biasanya diikuti penurunan CPU yang nyata.

Kesalahan threadsafe sulit ditemukan karena hanya muncul saat dua task kebetulan menyentuh storage yang sama pada waktu yang sama. Karena itu pemeriksaan kode (poin 1) tidak bisa digantikan uji saja. Uji beban hanya memperbesar peluang kesalahan tertangkap.

# Bab 66. VSAM RLS dan CICS

VSAM RLS memungkinkan banyak region CICS di banyak sistem memperbarui file VSAM yang sama, dengan penguncian per record di Coupling Facility. RLS menggantikan kebutuhan FOR tunggal yang menjadi titik kegagalan.

| Komponen RLS | Fungsi |
|---|---|
| SMSVSAM | Address space server RLS di setiap sistem |
| Structure `IGWLOCK00` | Lock structure RLS di CF |
| Cache structure | Buffer bersama untuk data VSAM |
| DFSMStvs | Opsi agar batch dapat memperbarui file RLS secara transaksional |

```
D SMS,SMSVSAM,ALL                  Status server RLS
D SMS,CFLS                         Statistik lock structure
D SMS,CFCACHE(*)                   Status cache structure
F CAPRD05,CEMT I FILE(ACCTMST) RLSACCESS   Mode akses file
```

Batch non-RLS tidak boleh membuka file yang sedang dipakai CICS dalam mode RLS. Tutup file di CICS dengan `CEMT S FILE(...) CLO`, atau quiesce dataset di semua region dengan `CEMT S DSNAME(...) QUIESCED` sesuai prosedur.

## 66.1 Analisis: Kapan RLS Diperlukan

VSAM RLS memungkinkan beberapa region CICS di beberapa sistem memperbarui file VSAM yang sama secara bersamaan, dengan integritas dijaga lewat lock di Coupling Facility. Kemampuan ini sangat kuat, tetapi membawa biaya dan konsekuensi operasional.

| Kebutuhan | Tanpa RLS | Dengan RLS |
|---|---|---|
| Beberapa AOR memperbarui | File dimiliki satu region; | Akses langsung dari setiap |

| Kebutuhan | Tanpa RLS | Dengan RLS |
|---|---|---|
| file yang sama | region lain mengakses lewat function shipping | AOR |
| AOR di beberapa sistem sysplex | Tidak mungkin tanpa satu pemilik file | Didukung penuh |
| Satu region berhenti | Region pemilik file adalah titik kegagalan tunggal | Region lain tetap bisa mengakses |
| Biaya per akses | Rendah bila lokal | Ada overhead lock dan cache CF |
| Batch memperbarui file | Batch berjalan saat CICS menutup file | Perlu pola khusus, lihat di bawah |

Konsekuensi terbesar RLS ada di batch. Program batch biasa yang memperbarui file tanpa mode RLS tidak dapat berjalan bersamaan dengan CICS yang membuka file itu dalam mode RLS. Pilihannya:

- **Menutup atau mengistirahatkan file di CICS** selama batch berjalan. Sederhana, tetapi layanan online yang membutuhkan file itu terhenti.
- **Mengubah batch agar memakai akses RLS**, dengan memperhatikan commit dan penguncian seperti transaksi online.
- **Memakai VSAM transaksional (DFSMStvs)**, sehingga batch dapat memperbarui file RLS dengan dukungan commit dan pemulihan.

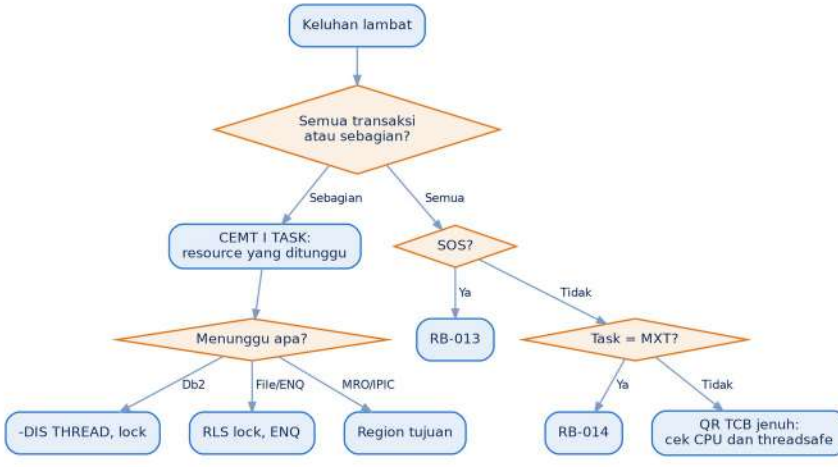
Keputusan memakai RLS sebaiknya diambil per file, berdasarkan kebutuhan ketersediaan dan pola akses. File master yang diperbarui terus-menerus oleh transaksi online dari banyak AOR adalah kandidat kuat. File referensi yang hampir tidak pernah berubah biasanya tidak memerlukannya.

Ukur overhead RLS di lingkungan uji dengan beban setara: waktu respons transaksi dan waktu layanan CF untuk structure lock dan cache RLS. Angka ini membantu memutuskan apakah RLS sepadan untuk setiap file.

# Bab 67. Diagnosis CICS

| Alat | Kegunaan |
|---|---|
| `CEMT I TASK` | Melihat task yang tertahan dan resource yang ditunggu |
| `CEDF` / `CEDX` | Menelusuri eksekusi transaksi langkah demi langkah (non-produksi) |
| `CETR` | Mengatur trace internal dan auxiliary |
| Transaction dump | Dibuat otomatis saat transaksi abend |
| System dump | Dianalisis di IPCS dengan `VERBX DFHPDnnn` sesuai level CICS |
| Statistik SMF 110 | Tren kinerja per transaksi |

## 67.1 Algoritma diagnosis “CICS lambat”




## 67.2 Runbook CICS

### RB-013: CICS Short on Storage

- **Gejala:** pesan seri `DFHSM013x`, transaksi lambat atau ditolak.
- **Diagnosis:** `CEMT I DSAS` untuk melihat DSA yang SOS; cari transaksi dengan pemakaian storage abnormal.

- **Tindakan:** batalkan task yang bocor jika teridentifikasi; naikkan `EDSALIM` jika REGION masih cukup; alihkan beban ke region lain lewat CICSPlex SM.
- **Pencegahan:** alert pemakaian DSA di atas 80%, uji beban setiap rilis besar.

### RB-014: Jumlah task mencapai MXT

- **Gejala:** transaksi mengantre, waktu respons melonjak tanpa SOS.
- **Diagnosis:** `CEMT I TASK`; biasanya satu resource (Db2, file, region lain) menahan banyak task.
- **Tindakan:** selesaikan penyebab tunggu lebih dulu; menaikkan MXT tanpa itu hanya memindahkan masalah ke SOS.
- **Pencegahan:** timeout (`DTIMOUT`) pada transaksi dan batas `MAXACTIVE` per `TRANCLASS`.

### RB-015: Transaksi abend AKCS massal

- **Gejala:** banyak transaksi timeout menunggu resource.
- **Diagnosis:** identifikasi resource bersama; periksa ENQ, record lock RLS, dan deadlock Db2.
- **Tindakan:** bebaskan pemegang resource; restart transaksi yang dibatalkan.
- **Pencegahan:** urutan akses resource yang konsisten di seluruh program.

### RB-016: Program baru tidak aktif setelah deploy

- **Gejala:** perilaku lama masih muncul setelah promosi load module.
- **Diagnosis:** `CEMT I PROG(xxx)` untuk melihat tanggal dan library asal program; periksa urutan `DFHRPL`.
- **Tindakan:** `CEMT S PROG(xxx) PHASEIN` di setiap region terkait.
- **Pencegahan:** langkah PHASEIN otomatis di pipeline deploy untuk semua AOR.

### RB-017: Koneksi CICS ke Db2 terputus

- **Gejala:** abend `AEY9`, pesan `DFHDB2xxx` di log.
- **Diagnosis:** `CEMT I DB2CONN`; periksa status subsistem Db2 dan pesan di SYSLOG.
- **Tindakan:** setelah Db2 aktif, `CEMT S DB2CONN CONNECTED`.
- **Pencegahan:** `STANDBYMODE(RECONNECT)` pada DB2CONN dan urutan start yang benar di otomasi.

## 67.3 Analisis: Membaca Statistik CICS Harian

CICS menghasilkan statistik yang sangat rinci, tetapi hanya beberapa angka yang perlu dilihat setiap hari untuk mengetahui apakah sebuah region sehat. Angka-angka ini sebaiknya dikumpulkan otomatis dan disajikan sebagai tren, bukan dibaca manual dari laporan panjang.

| Angka | Arti | Tanda masalah |
|---|---|---|
| Puncak task bersamaan | Task terbanyak pada satu waktu | Mendekati MXT |
| Berapa kali mencapai MXT | Transaksi harus antre karena batas task | Lebih dari nol secara rutin |
| Berapa kali short on storage | Region kehabisan storage di salah satu DSA | Lebih dari nol |
| Puncak pemakaian EDSA | Storage tertinggi yang dipakai | Tren naik dari hari ke hari tanpa kenaikan beban |
| Abend transaksi per jenis | Kegagalan aplikasi | Lonjakan pada satu kode transaksi |
| CPU per transaksi | Biaya rata-rata | Kenaikan setelah rilis aplikasi |

$$\text{Rasio ketinggian task} = \frac{\text{puncak task}}{\text{MXT}}$$

Rasio di atas sekitar 0,8 di jam sibuk menandakan region mendekati batasnya. Penyebabnya perlu dicari sebelum MXT benar-benar tercapai. Ingat bahwa puncak task yang naik sering bukan karena beban naik, tetapi karena waktu respons memburuk, misalnya karena Db2 atau file yang

lambat (Bab 32.3). Menaikkan MXT tanpa mengetahui penyebabnya hanya memindahkan antrean.

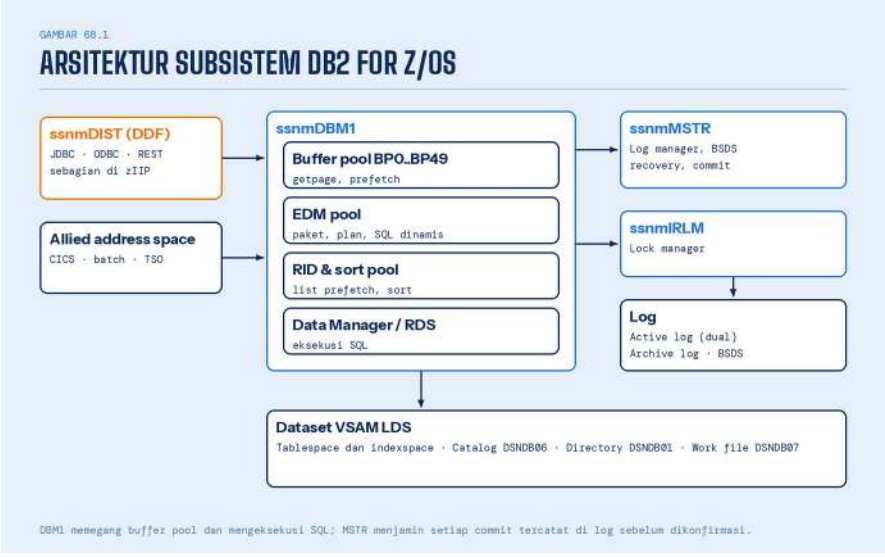
Tren pemakaian EDSA yang terus naik dari hari ke hari, sementara jumlah transaksi stabil, adalah pola kebocoran storage yang sama seperti di common area z/OS (Bab 10.4). Garis dasarnya naik, dan pada suatu titik region mengalami SOS. Restart region terjadwal bisa menunda masalah, tetapi yang harus dicari adalah program yang tidak melepaskan storage.

Satukan angka-angka ini dalam laporan harian per region, berdampingan dengan laporan kesehatan sysplex (Bab 181.1). Region yang sehat cukup ditandai OK. Region dengan angka di luar batas ditampilkan dengan penjelasan singkat dan pemiliknya.

# 🗄️ Bagian XVII — Pendalaman Db2 for z/OS

## Bab 68. Arsitektur Internal Db2

Subsistem Db2 terdiri dari beberapa address space yang masing-masing punya tugas jelas. Memahami pembagian ini mempercepat diagnosis: masalah log dicari di MSTR, masalah SQL dan memori di DBM1, masalah koneksi jarak jauh di DIST.




*Gambar 68.1 — Arsitektur subsistem Db2 for z/OS*

| Objek sistem | Fungsi | Risiko jika rusak |
|---|---|---|
| BSDS (dual) | Inventaris log aktif dan arsip, titik restart | Db2 tidak bisa start |
| Active log (dual) | Catatan setiap perubahan sebelum commit dikonfirmasi | Kehilangan kemampuan recovery |
| Archive log | Salinan log yang sudah penuh | Recovery ke titik lama gagal |
| Catalog `DSNDB06` | Metadata semua objek | Db2 tidak dapat dipakai |
| Directory `DSNDB01` | Informasi internal paket dan | Db2 tidak dapat dipakai |

| Objek sistem | Fungsi | Risiko jika rusak |
|---|---|---|
| | recovery | |
| Work file DSNDB07 | Sort dan tabel sementara | Query besar gagal |
| DSNZPARM | Parameter subsistem | Perilaku Db2 berubah |

## 68.1 Parameter DSNZPARM yang sering disentuh

| Parameter | Fungsi |
|---|---|
| CTHREAD | Batas thread lokal bersamaan |
| MAXDBAT | Batas thread DDF aktif |
| IRLMRWT | Batas waktu tunggu lock sebelum timeout |
| UTIMOUT | Pengali timeout untuk utilitas |
| NUMLKTS | Batas lock per tablespace sebelum eskalasi |
| NUMLKUS | Batas lock per user |
| CHKFREQ | Frekuensi checkpoint sistem |
| IDTHTOIN | Timeout thread DDF yang menganggur |

Perubahan DSNZPARM yang bisa diaktifkan tanpa restart dilakukan lewat `-SET SYSPARM LOAD(DSNZPRMx)`. Catat setiap perubahan karena dampaknya luas.

## 68.2 Analisis: Menentukan Batas Thread dan Koneksi

Db2 membatasi jumlah thread yang boleh aktif bersamaan lewat parameter subsistem. Batas ini melindungi Db2 dari kelebihan beban, tetapi batas yang terlalu kecil membuat pekerjaan sah harus antre. Parameter yang paling sering perlu ditinjau adalah batas thread lokal (CTHREAD), batas thread DDF yang aktif (MAXDBAT), dan batas koneksi DDF (CONDBAT).

Kebutuhan thread DDF dapat diperkirakan dengan hukum Little yang sudah dipakai berkali-kali di buku ini:

$$N_{\text{DBAT aktif}} \approx \lambda_{\text{permintaan DDF}} \times T_{\text{di Db2}}$$

Contoh: 800 permintaan DDF per detik di jam puncak, masing-masing menghabiskan rata-rata 50 ms di Db2, membutuhkan sekitar 40 DBAT aktif. Dengan cadangan untuk lonjakan dan permintaan yang lebih lambat, MAXDBAT di kisaran 100–150 masuk akal untuk beban ini. Bila Db2 melambat dan waktunya naik menjadi 200 ms, kebutuhan naik menjadi 160 DBAT, dan batas mulai tercapai.

| Gejala | Kemungkinan penyebab | Tindakan |
|---|---|---|
| Permintaan DDF antre menunggu DBAT | MAXDBAT tercapai | Cari penyebab waktu di Db2 naik; baru pertimbangkan menaikkan batas |
| Koneksi baru ditolak | CONDBAT tercapai | Periksa connection pool di sisi klien; koneksi mungkin tidak dilepas |
| Thread lokal menunggu | CTHREAD tercapai | Periksa job batch dan region CICS yang membuka thread berlebihan |
| Banyak thread diam lama | Klien menahan koneksi tanpa aktivitas | Mode thread inaktif DDF dan pengaturan pool klien |

Seperti MXT di CICS, menaikkan batas thread tanpa mengetahui penyebabnya hanya memindahkan antrean ke dalam Db2, dan bisa memperburuk kontensi lock dan memori. Batas thread adalah katup pengaman. Saat katup itu terbuka penuh, yang harus diselidiki adalah mengapa tekanan naik.

## Bab 69. Buffer Pool dan Memori

Buffer pool adalah faktor kinerja Db2 yang paling besar. Setiap halaman yang ditemukan di memori menghemat satu I/O ke disk.

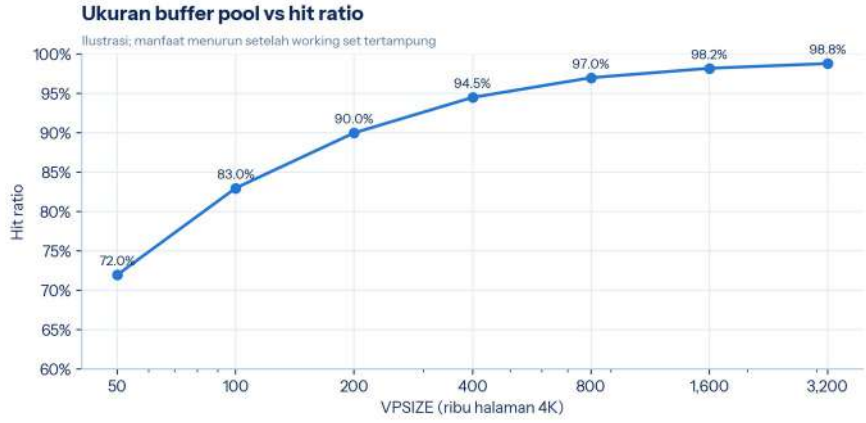
| Buffer pool | Isi yang disarankan |
|---|---|
| BP0 | Catalog dan directory saja |
| BP1 | Tablespace data transaksi utama |
| BP2 | Index transaksi utama |
| BP3 | Tabel referensi kecil yang sering dibaca |

| Buffer pool | Isi yang disarankan |
|---|---|
| | (pasang `PGSTEAL(NONE)` bila muat seluruhnya) |
| BP7 / BP8K7 | Work file sort |
| BP8K0, BP16K0, BP32K | Objek dengan halaman 8K, 16K, 32K |

Rasio hit sistem dihitung dari statistik buffer pool:

$$HR = \frac{\text{getpage} - (\text{sync read} + \text{halaman dibaca prefetch})}{\text{getpage}}$$

Menambah ukuran buffer pool memberi manfaat besar di awal, lalu menurun setelah working set aplikasi tertampung. Pertimbangkan juga `PGFIX(YES)` agar halaman buffer tidak perlu di-fix ulang setiap I/O.




*Ilustrasi · data rekaan*

```
-DIS BUFFERPOOL(BP1) DETAIL(INTERVAL)   Statistik sejak
tampilan terakhir
-ALTER BUFFERPOOL(BP1) VPSIZE(400000)   Ubah ukuran
(halaman)
-ALTER BUFFERPOOL(BP1) PGFIX(YES)       Berlaku setelah
pool dialokasi ulang
```

## 69.1 Analisis: Strategi Pembagian Buffer Pool

Satu buffer pool besar untuk semua objek terlihat sederhana, tetapi membuat objek dengan pola akses berbeda saling mengganggu. Scan besar dari laporan analitik bisa mengusir halaman index yang sering dipakai transaksi online. Pembagian buffer pool yang dirancang dengan baik memisahkan pola-pola itu.

| Kelompok objek | Pola akses | Alasan dipisah |
|---|---|---|
| Index transaksi online | Acak, sangat sering | Harus tetap di memori; tidak boleh terusir oleh scan |
| Data transaksi online | Acak | Ukuran disesuaikan dengan working set |
| Tabel referensi kecil | Sangat sering, jarang berubah | Bisa dimuat hampir seluruhnya |
| Tabel riwayat dan laporan | Sequential, volume besar | Scan tidak boleh mengganggu kelompok lain |
| Work file (sort, temporary) | Sementara, intensif | Pola yang sangat berbeda dari data permanen |
| Katalog Db2 | Sistem | Dipisah agar tidak dipengaruhi beban aplikasi |

$$\text{Sync I/O per detik} = \text{getpage per detik} \times (1 - HR)$$

Rumus ini menunjukkan mengapa hit ratio yang tampak tinggi tetap bisa berarti banyak I/O. Buffer pool dengan 200.000 getpage per detik dan hit ratio 98% masih melakukan sekitar 4.000 I/O sinkron per detik. Menaikkan hit ratio menjadi 99% memotongnya menjadi sekitar 2.000. Karena itu ukuran dan pembagian buffer pool dinilai dari I/O sinkron per detik, bukan dari persentase saja (Bab 69).

Dua pengaturan lain sering memberi manfaat besar untuk buffer pool yang sibuk. Pertama, `PGFIX(YES)` mengunci halaman buffer di memori nyata dan menghemat CPU untuk setiap I/O. Kedua, frame berukuran besar (`FRAMESIZE`) mengurangi overhead pengelolaan memori. Keduanya membutuhkan memori nyata yang cukup di LPAR, jadi perhitungannya harus disesuaikan dengan kapasitas memori sistem.

Tinjau pembagian buffer pool setiap kali aplikasi besar ditambahkan atau pola akses berubah, dan ukur dampak setiap perubahan dari statistik buffer pool sebelum dan sesudahnya.

# Bab 70. Logging, Locking, dan Commit

Setiap perubahan data ditulis ke log sebelum commit dikonfirmasi. Karena itu kecepatan tulis log menentukan batas atas throughput transaksi.

## 70.1 Penguncian

| Konsep | Arti | Praktik baik |
|---|---|---|
| `LOCKSIZE` | Granularitas lock: ROW, PAGE, TABLE, ANY | PAGE atau ANY untuk kebanyakan tabel; ROW hanya jika perlu |
| `LOCKMAX` | Batas lock sebelum eskalasi ke tablespace | Sesuaikan dengan pola batch |
| Lock escalation | Ribuan lock kecil diganti satu lock besar | Hindari di jam online |
| Timeout | Menunggu lock melebihi `IRLMRWT` (SQLCODE -911/-913) | Commit lebih sering |
| Deadlock | Dua pihak saling menunggu | Urutan akses tabel konsisten |
| Isolation | UR, CS, RS, RR | CS untuk OLTP; UR hanya untuk laporan yang toleran |

## 70.2 Algoritma commit dan restart batch

Program batch besar harus commit berkala dan mampu melanjutkan dari titik terakhir. Tanpa itu, satu abend di menit ke-90 berarti mengulang dari nol dan menahan lock selama 90 menit.

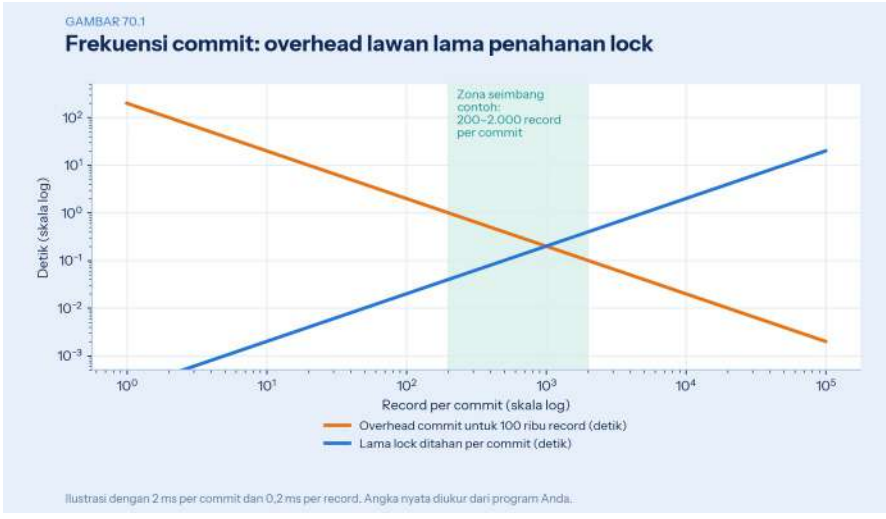
```
ALGORITMA BatchDenganCheckpoint(input, N_commit)
  posisi <- baca_tabel_checkpoint(job)          -- kunci
terakhir yang sudah commit
  buka kursor input WHERE kunci > posisi ORDER BY kunci
  hitung <- 0
  untuk setiap record r:
```

```
      proses(r)                                          --
UPDATE/INSERT
      hitung <- hitung + 1
      jika hitung mod N_commit = 0 maka
          update_tabel_checkpoint(job, r.kunci)  -- dalam
unit kerja yang sama
          COMMIT
  update_tabel_checkpoint(job, SELESAI)
  COMMIT
```

Nilai `N_commit` yang umum berkisar ratusan hingga beberapa ribu record, atau setiap beberapa detik. Kursor harus dideklarasikan `WITH HOLD` agar tetap terbuka setelah commit.

## 70.3 Analisis: Menentukan Frekuensi Commit Batch

Program batch yang memperbarui jutaan baris harus memilih seberapa sering melakukan commit. Pilihan ini adalah pertukaran antara dua biaya yang bergerak berlawanan arah.




*Gambar 70.1 — Frekuensi commit: overhead lawan lama penahanan lock*

$$T_{\text{overhead}} = \frac{N_{\text{record}}}{n} \times t_{\text{commit}} \qquad T_{\text{lock}} = n \times t_{\text{record}}$$

n adalah jumlah record per commit. Commit terlalu sering, misalnya setiap record, membuat overhead commit mendominasi: setiap commit

memaksa penulisan log ke disk. Commit terlalu jarang, misalnya sekali di akhir, membuat lock ditahan sangat lama. Transaksi online yang menyentuh data yang sama harus menunggu, lock bisa bereskalasi menjadi lock tabel, dan bila program gagal, rollback-nya membutuhkan waktu sangat lama.

| Masalah | Penyebab | Gejala |
|---|---|---|
| Batch lambat tanpa alasan jelas | Commit terlalu sering | Waktu tunggu log tinggi di statistik akuntansi |
| Timeout di transaksi online saat batch berjalan | Commit terlalu jarang | Lock wait dan `SQLCODE -911` atau `-913` di online |
| Lock escalation | Terlalu banyak lock dalam satu unit kerja | Pesan eskalasi lock di log Db2 |
| Rollback berjam-jam setelah batch gagal | Unit kerja sangat besar | Pemulihan lama setelah abend |

Nilai yang seimbang biasanya berada di kisaran ratusan sampai beberapa ribu record per commit, tergantung biaya per record dan seberapa sering data yang sama disentuh transaksi online. Buat frekuensi commit sebagai parameter program, bukan konstanta di kode, agar bisa disesuaikan tanpa kompilasi ulang: lebih jarang di window EOD yang sepi, lebih sering bila batch berjalan bersamaan dengan online.

Commit yang tepat juga harus disertai checkpoint yang tepat (Bab 60). Setiap commit sebaiknya mencatat posisi terakhir yang diproses, agar restart setelah kegagalan dapat melanjutkan dari commit terakhir tanpa memproses ulang (Bab 136.1).

## Bab 71. Data Sharing

Db2 data sharing membuat beberapa member Db2 di sistem berbeda membaca dan menulis database yang sama. Koordinasi dilakukan lewat tiga jenis structure di Coupling Facility.

| Structure | Fungsi |
|---|---|
| Lock structure | Lock global antar-member |
| SCA (Shared Communications Area) | Status bersama: objek bermasalah, BSDS, dll. |

| Structure | Fungsi |
|---|---|
| Group buffer pool (GBP) | Cache halaman yang diubah dan dibaca bersama |

```
-DIS GROUP                          Status seluruh member
-DIS GBPOOL(GBP1) GDETAIL          Statistik group buffer
pool
-DIS DB(*) SPACENAM(*) LPL          Halaman di logical
page list
-DIS DB(*) SPACENAM(*) GRECP        Objek butuh pemulihan
GBP
D XCF,STR,STRNAME=DSNDB0G_GBP1     Ukuran structure di CF
```

Jika satu member gagal, lock yang dipegangnya menjadi *retained lock* sampai member itu di-restart. Restart member secepatnya, bila perlu di sistem lain dengan `LIGHT(YES)`, agar data yang terkunci kembali tersedia.

## 71.1 Analisis: Saat Satu Anggota Data Sharing Gagal

Dalam Db2 data sharing, kegagalan satu anggota tidak menghentikan anggota lain. Namun ada satu efek yang sering mengejutkan: lock yang dipegang anggota yang gagal tidak langsung dilepas. Lock itu menjadi *retained lock*, dijaga di CF, dan melindungi data yang mungkin sedang diubah saat anggota gagal. Selama retained lock ada, anggota lain tidak dapat mengakses baris atau halaman yang dilindunginya.

$$T_{\text{dampak}} \approx T_{\text{deteksi}} + T_{\text{restart anggota}} + T_{\text{pemulihan unit kerja}}$$

Retained lock baru dilepas setelah anggota yang gagal di-restart dan menyelesaikan atau membatalkan unit kerja yang tertunda. Karena itu kecepatan restart anggota yang gagal menentukan berapa lama sebagian data tidak dapat diakses oleh anggota lain.

| Mekanisme | Fungsi |
|---|---|
| ARM | Restart otomatis anggota Db2 yang gagal, di sistem yang sama atau sistem lain |
| Restart ringan (`LIGHT(YES)`) | Anggota di-restart di sistem lain dengan jejak minimal, hanya untuk melepas retained lock, |

| Mekanisme | Fungsi |
|---|---|
| | lalu berhenti |
| Aplikasi yang menangani `-904` atau `-911` | Transaksi yang menyentuh data terkunci gagal dengan jelas dan dapat dicoba lagi |

Restart ringan berguna bila sistem tempat anggota berjalan juga gagal. Anggota di-restart di sistem lain yang masih hidup tanpa membebaninya dengan beban penuh, cukup untuk membersihkan unit kerja dan melepas retained lock.

Uji skenario ini di lingkungan data sharing uji: batalkan satu anggota dengan paksa di tengah beban, lalu ukur berapa lama sampai retained lock hilang dan transaksi di anggota lain kembali normal. Angka ini adalah bagian nyata dari RTO untuk kegagalan satu anggota, dan sering lebih panjang dari yang diperkirakan bila ARM belum dikonfigurasi dengan benar.

# Bab 72. Utilitas, Recovery, dan Access Path

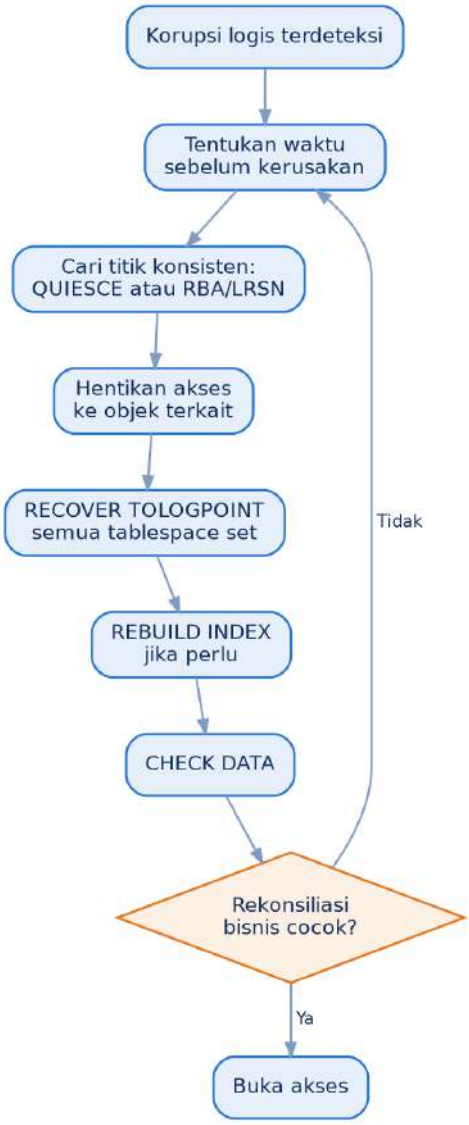
## 72.1 Utilitas rutin

```
//UTIL     EXEC
DSNUPROC,SYSTEM=DB1A,UID='COPY.TRX',UTPROC=''
//SYSIN    DD *
  TEMPLATE TCOPY DSN(DB1A.IC.&DB..&TS..D&DATE..T&TIME.)
           UNIT(SYSDA)
  COPY TABLESPACE BANKDB.TSTRX COPYDDN(TCOPY)
       FULL YES SHRLEVEL CHANGE
  RUNSTATS TABLESPACE BANKDB.TSTRX TABLE(ALL) INDEX(ALL)
/*
```

| Utilitas | Fungsi | Frekuensi umum |
|---|---|---|
| `COPY` | Image copy untuk recovery | Penuh mingguan, incremental harian |
| `REORG ... SHRLEVEL CHANGE` | Reorganisasi tanpa menghentikan akses | Berdasarkan statistik real-time |
| `RUNSTATS` | Statistik untuk optimizer | Setelah perubahan volume besar |

| Utilitas | Fungsi | Frekuensi umum |
|---|---|---|
| `QUIESCE` | Titik konsisten untuk sekelompok tablespace | Sebelum EOD |
| `RECOVER` | Memulihkan ke saat ini atau titik tertentu | Saat insiden |
| `MODIFY RECOVERY` | Menghapus riwayat image copy lama | Bulanan |
| `CHECK DATA` / `CHECK INDEX` | Memeriksa integritas | Setelah recovery atau load |

## 72.2 Algoritma recovery ke titik waktu




Recover ke titik waktu harus mencakup semua tablespace yang berelasi. Memulihkan satu tabel saja akan membuat data antar-tabel tidak konsisten.

## 72.3 Menjaga stabilitas access path

- Jalankan `EXPLAIN` untuk SQL kritikal dan simpan hasil `PLAN_TABLE` sebagai baseline.
- Gunakan `REBIND PACKAGE(...) APCOMPARE(WARN) APREUSE(WARN)` setelah upgrade agar access path lama dipertahankan bila memungkinkan.
- Aktifkan plan management (`PLANMGMT(EXTENDED)`) agar `REBIND ... SWITCH(PREVIOUS)` bisa mengembalikan access path lama dalam hitungan detik.

## 72.4 Runbook Db2

### RB-018: Active log hampir penuh

- **Gejala:** pesan `DSNJ110E`, lalu `DSNJ111E` bila berlanjut.
- **Diagnosis:** `-DIS LOG`; periksa apakah proses offload ke archive terhambat (tape, ruang disk).
- **Tindakan:** pulihkan jalur offload; `-ARCHIVE LOG` bila perlu; tambah dataset active log lewat `-SET LOG NEWLOG`.
- **Pencegahan:** active log cukup besar untuk puncak EOD dan archive ke disk.

### RB-019: Objek masuk LPL atau GRECP

- **Gejala:** SQLCODE -904 dengan reason resource unavailable; tampil di `-DIS ... LPL` atau `GRECP`.
- **Diagnosis:** cari penyebab awal di SYSLOG (kegagalan CF, I/O error, member abend).
- **Tindakan:** `-START DB(...) SPACENAM(...)` untuk memicu pemulihan otomatis dari log.
- **Pencegahan:** CF duplexing untuk GBP penting dan pemantauan kesehatan CF.

### RB-020: Kinerja query turun setelah REBIND

- **Gejala:** elapsed time transaksi atau batch naik drastis setelah maintenance.
- **Diagnosis:** bandingkan `PLAN_TABLE` sebelum dan sesudah.
- **Tindakan:** `REBIND PACKAGE(...) SWITCH(PREVIOUS)`; lalu analisis statistik dan index.
- **Pencegahan:** `APCOMPARE`/`APREUSE` dan plan management aktif.

### RB-021: Lock escalation saat batch di jam online

- **Gejala:** transaksi online timeout, pesan eskalasi lock di SYSLOG.
- **Diagnosis:** identifikasi job batch dan tablespace yang tereskalasi.
- **Tindakan:** hentikan atau tunda batch; restart dengan commit lebih sering.
- **Pencegahan:** algoritma checkpoint Bab 70.2 dan jadwal batch di luar jam puncak.

### RB-022: Group buffer pool kekurangan ruang

- **Gejala:** pesan kekurangan GBP (misalnya `DSNB319A`), write failure ke GBP.
- **Diagnosis:** `-DIS GBPOOL(...) GDETAIL` untuk rasio directory dan data entry.
- **Tindakan:** perbesar structure dengan `SETXCF START,ALTER` jika CFRM mengizinkan; percepat castout.
- **Pencegahan:** ukur GBP dari statistik puncak dan izinkan auto-alter di CFRM policy.

## 72.5 Analisis: Strategi Image Copy dan Waktu Pemulihan

Image copy menentukan seberapa cepat sebuah tablespace dapat dipulihkan. `RECOVER` memulihkan image copy terakhir, lalu menerapkan semua perubahan dari log sejak image copy itu dibuat. Semakin jarang image copy, semakin banyak log yang harus diterapkan.


*Gambar 72.1 — Frekuensi image copy dan waktu pemulihan*

$$T_{\text{RECOVER}} \approx T_{\text{restore}} + \bar{L} \times t_{\text{apply per jam log}}$$

L adalah rata-rata jumlah log yang perlu diterapkan, kira-kira separuh interval antar-image copy. Dalam ilustrasi, image copy mingguan membuat waktu pemulihan tablespace yang sibuk mendekati 2 jam, tepat di batas RTO contoh. Image copy dua pekanan melampauinya.

| Pilihan | Pengaruh | Pertimbangan |
|---|---|---|
| Full image copy lebih sering | Log yang diterapkan lebih sedikit | Waktu dan ruang untuk image copy |
| Incremental image copy | Hanya halaman yang berubah; lebih cepat dibuat | Pemulihan menggabungkan beberapa salinan |
| Image copy dengan FlashCopy | Salinan hampir seketika di level storage | Butuh dukungan storage dan kapasitas salinan |
| Frekuensi berbeda per objek | Objek kritikal lebih sering | Kebijakan per kelompok objek |

Objek yang sering berubah dan kritikal bagi layanan, seperti tabel saldo dan transaksi, adalah kandidat untuk image copy harian atau lebih sering, dan untuk FlashCopy agar tidak membebani window. Tabel referensi yang jarang berubah bisa cukup mingguan, karena log yang harus diterapkan untuknya sedikit.

Ukur waktu RECOVER nyata untuk beberapa tablespace perwakilan di lingkungan uji, setidaknya sekali setahun. Bandingkan hasilnya dengan RTO layanan terkait. Seperti uji DR, angka yang terukur lebih berharga daripada perkiraan, dan sering mengejutkan.

# 📜 Bagian XVIII — COBOL untuk System Engineer

System engineer tidak harus menulis COBOL, tetapi wajib bisa membacanya. Hampir setiap insiden batch EOD berakhir di satu baris program COBOL dan satu field data yang tidak sesuai harapan.

## Bab 73. Anatomi Program COBOL

Program COBOL terdiri dari empat division yang urutannya tetap. Mengetahui di division mana sesuatu didefinisikan mempercepat penelusuran dari dump ke kode.

| Division | Isi | Yang dicari system engineer |
|---|---|---|
| IDENTIFICATION | Nama program, penulis | `PROGRAM-ID` untuk dicocokkan dengan nama load module |
| ENVIRONMENT | Pemetaan file ke nama DD | `SELECT ... ASSIGN TO` → nama DD di JCL |
| DATA | File layout, working storage, linkage | Definisi field, panjang, tipe data |
| PROCEDURE | Logika program | Paragraf dan baris yang ditunjuk offset abend |

Contoh program ringkas yang menghitung akrual harian:

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. ACRBUNGA.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT TRX-IN  ASSIGN TO TRXIN
                  FILE STATUS IS WS-FS-IN.
           SELECT RPT-OUT ASSIGN TO RPTOUT.
       DATA DIVISION.
       FILE SECTION.
       FD  TRX-IN RECORDING MODE F.
       COPY ACRREC.
```

```
       FD  RPT-OUT RECORDING MODE F.
       01  RPT-LINE               PIC X(133).
       WORKING-STORAGE SECTION.
       01  WS-FS-IN               PIC XX.
       01  WS-EOF                 PIC X VALUE 'N'.
           88 EOF-TRX                   VALUE 'Y'.
       01  WS-TOTAL-BUNGA         PIC S9(13)V99 COMP-3
VALUE 0.
       01  WS-JML-REC             PIC S9(9) COMP VALUE 0.
       PROCEDURE DIVISION.
       0000-MAIN.
           OPEN INPUT TRX-IN OUTPUT RPT-OUT
           IF WS-FS-IN NOT = '00'
              DISPLAY 'OPEN TRXIN GAGAL FS=' WS-FS-IN
              MOVE 16 TO RETURN-CODE
              GOBACK
           END-IF
           PERFORM UNTIL EOF-TRX
              READ TRX-IN
                 AT END SET EOF-TRX TO TRUE
                 NOT AT END PERFORM 1000-PROSES
              END-READ
           END-PERFORM
           DISPLAY 'RECORD: ' WS-JML-REC ' TOTAL: ' WS-
TOTAL-BUNGA
           CLOSE TRX-IN RPT-OUT
           GOBACK.
       1000-PROSES.
           IF ACR-SALDO IS NOT NUMERIC
              DISPLAY 'DATA TIDAK VALID REK=' ACR-NO-REK
              MOVE 8 TO RETURN-CODE
           ELSE
              COMPUTE ACR-BUNGA-HARIAN ROUNDED =
                      ACR-SALDO * ACR-RATE / 36500
              ADD ACR-BUNGA-HARIAN TO WS-TOTAL-BUNGA
              ADD 1 TO WS-JML-REC
           END-IF.
```

Perhatikan tiga kebiasaan baik di contoh ini: `FILE STATUS` selalu diperiksa, field numerik divalidasi dengan `IS NUMERIC` sebelum dihitung, dan hasil akhir dicetak sebagai total kontrol.

## 73.1 Analisis: Memahami Program COBOL yang Tidak Anda Tulis

System engineer sering harus membaca program COBOL yang ditulis puluhan tahun lalu oleh orang yang sudah tidak ada, saat program itu gagal pukul dua pagi. Membaca dari baris pertama sampai terakhir hampir tidak pernah efisien. Pendekatan yang lebih cepat mengikuti data, bukan urutan kode.

```
ALGORITMA PahamiProgramCOBOL(program, masalah)
  1. ENVIRONMENT dan FILE SECTION: file apa yang dibaca
dan ditulis (SELECT, FD)
  2. Cocokkan nama file dengan DD di JCL: dataset nyata
mana yang dipakai
  3. Temukan field yang terkait masalah (misalnya field
nominal saat S0C7)
  4. Cari semua tempat field itu diubah: MOVE ke field,
COMPUTE, READ INTO, REDEFINES
  5. Ikuti PERFORM dari paragraf utama hanya sepanjang
jalur yang menyentuh field itu
  6. Catat asumsi program tentang data (panjang, tanda,
isi default)
  7. Bandingkan asumsi itu dengan data nyata di file saat
kegagalan
```

Pendekatan ini memotong program ribuan baris menjadi beberapa puluh baris yang relevan. Pertanyaan “apa yang dilakukan program ini?” diganti dengan pertanyaan yang lebih kecil dan bisa dijawab: “dari mana nilai field ini berasal?”

| Tanda program sulit dirawat | Risiko | Mitigasi |
|---|---|---|
| Banyak `GO TO` lintas paragraf | Alur sulit diikuti | Petakan alur dengan alat analisis sebelum mengubah |
| `REDEFINES` bertingkat pada record yang sama | Satu byte dibaca dengan beberapa tipe berbeda | Dokumentasikan layout setiap varian record |
| Copybook yang disalin dan diubah per program | Layout yang sama tetapi berbeda di tiap program | Satukan copybook; kompilasi ulang semua pemakai |
| Konstanta bisnis ditulis langsung di kode | Perubahan tarif atau batas butuh kompilasi ulang | Pindahkan ke tabel parameter |

Alat analisis aplikasi, seperti IBM Application Discovery and Delivery Intelligence, dapat memetakan program, copybook, file, dan tabel yang saling terkait untuk seluruh portofolio. Untuk investigasi insiden, pendekatan mengikuti data di atas biasanya sudah cukup. Untuk proyek modernisasi (Bab 53.1), peta lengkap dari alat seperti itu sangat membantu menilai risiko perubahan.

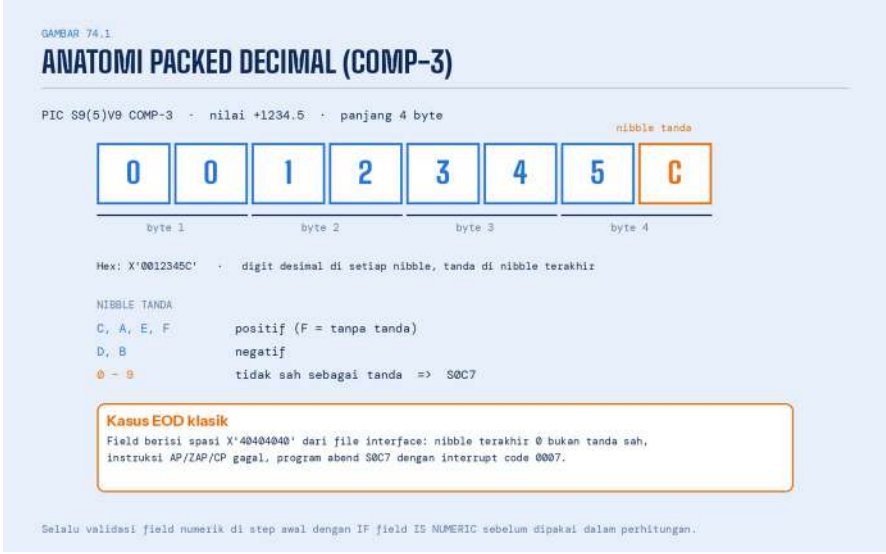
# Bab 74. Tipe Data dan Representasi di Memori

Sebagian besar abend data di mainframe berasal dari salah paham tentang bagaimana angka disimpan. COBOL menyimpan angka dalam beberapa format yang ukurannya berbeda untuk nilai yang sama.

| Klausa | Format | Contoh `PIC S9(7)` | Ukuran |
|---|---|---|---|
| `DISPLAY` (default) | Zoned decimal: satu digit per byte, tanda di zone byte terakhir | `F1F2F3F4F5F6C7` | 7 byte |
| `COMP-3` / `PACKED-DECIMAL` | Dua digit per byte, tanda di nibble terakhir | `1234567C` | 4 byte |
| `COMP` / `BINARY` | Biner berkomplemen dua | Fullword | 4 byte |
| `COMP-5` | Biner native, tanpa pemotongan desimal | Fullword | 4 byte |
| `COMP-1` / `COMP-2` | Floating point hexadesimal | — | 4 / 8 byte |

Panjang field packed decimal dihitung dari jumlah digit n:

$$\text{byte} = \left\lfloor \frac{n}{2} \right\rfloor + 1$$



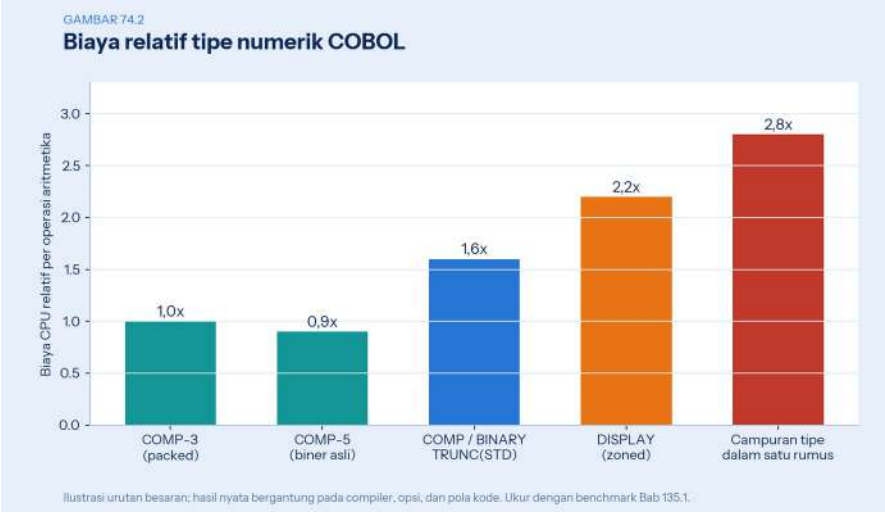

*Gambar 74.1 — Anatomi packed decimal (COMP-3)*

## 74.1 Jebakan umum

| Jebakan | Akibat | Pencegahan |
|---|---|---|
| `REDEFINES` dengan panjang tidak cocok | Data terbaca bergeser, S0C7 atau hasil salah | Periksa panjang setiap redefinisi |
| Copybook berubah tetapi program lama tidak dikompilasi ulang | Record terbaca bergeser | Kompilasi ulang semua pemakai copybook |
| `MOVE` alfanumerik ke numerik | Nilai tidak sah tersimpan tanpa error | Validasi dengan `IS NUMERIC` |
| Working storage tidak diinisialisasi | Nilai sampah X'00' atau X'40' | `VALUE` atau `INITIALIZE`; opsi `INITCHECK` |
| Subscript di luar batas `OCCURS` | S0C4 atau data lain tertimpa | Opsi `SSRANGE` di lingkungan uji |
| Pemotongan biner (`TRUNC`) | Nilai terpotong diam-diam | Samakan opsi `TRUNC` di seluruh aplikasi |

## 74.2 Analisis: Memilih Tipe Numerik COBOL

Pilihan tipe data numerik di COBOL memengaruhi dua hal: ketepatan hasil dan biaya CPU. Untuk uang, ketepatan tidak bisa ditawar. Untuk penghitung dan indeks, efisiensi CPU lebih penting. Memilih tipe yang tepat untuk setiap kegunaan menghemat CPU tanpa mengorbankan ketepatan.




*Gambar 74.2 — Biaya relatif tipe numerik COBOL*

| Kegunaan | Tipe yang disarankan | Alasan |
|---|---|---|
| Nominal uang, saldo, bunga | `COMP-3` (packed decimal) | Aritmetika desimal persis; didukung instruksi hardware |
| Penghitung loop, subscript, indeks | `COMP-5` atau binary dengan opsi yang sesuai | Aritmetika biner paling efisien |
| Field di record yang dipertukarkan dengan sistem lain | Sesuai spesifikasi interface, sering `DISPLAY` | Kompatibilitas lebih penting dari efisiensi |
| Perhitungan ilmiah | `COMP-1` / `COMP-2` (floating point) | Tidak pernah untuk uang: pembulatan biner tidak cocok untuk desimal |

Dua jebakan umum:

- **Binary dengan `TRUNC(STD)`.** Compiler harus memotong hasil sesuai jumlah digit di PIC, sehingga setiap operasi membutuhkan instruksi tambahan. `COMP-5` tidak dipotong sesuai PIC, sehingga lebih efisien untuk penghitung.
- **Campuran tipe dalam satu rumus.** Rumus yang mencampur `DISPLAY`, `COMP`, dan `COMP-3` memaksa konversi berulang. Samakan tipe field yang sering dihitung bersama.

Floating point untuk uang adalah kesalahan yang berbahaya. Nilai seperti 0,1 tidak dapat disimpan persis dalam biner. Setelah jutaan operasi, selisih kecil terakumulasi menjadi selisih rekonsiliasi yang sulit dijelaskan.

Perubahan tipe data pada field yang sudah ada di file atau tabel adalah perubahan layout, bukan sekadar optimasi. Terapkan optimasi tipe pada field kerja di WORKING-STORAGE lebih dulu. Field ini tidak memengaruhi format data yang disimpan atau dipertukarkan.

## Bab 75. Compile, Link, dan Deploy

Program COBOL diproses dua tahap: compiler (`IGYCRCTL`) menghasilkan object, lalu binder (`IEWL`) menghasilkan load module atau program object di PDSE.

```
//COMPLINK JOB (BANK,DEV),'ACRBUNGA',CLASS=A,MSGCLASS=X
//COBOL    EXEC PGM=IGYCRCTL,REGION=0M,
//
PARM='ARCH(14),OPT(2),LIST,MAP,XREF,RENT,SQL,TEST(NOSEP)'
//STEPLIB  DD DISP=SHR,DSN=IGY.V6R4M0.SIGYCOMP
//SYSLIB   DD DISP=SHR,DSN=BANK.DEV.COPYLIB
//DBRMLIB  DD DISP=SHR,DSN=BANK.DEV.DBRMLIB(ACRBUNGA)
//SYSIN    DD DISP=SHR,DSN=BANK.DEV.COBOL(ACRBUNGA)
//SYSLIN   DD DSN=&&OBJ,DISP=(NEW,PASS),SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD SPACE=(CYL,(5,5)),UNIT=SYSALLDA
//*  (SYSUT2 s.d. SYSUT15 dan SYSMDECK diperlukan
compiler)
//LKED     EXEC PGM=IEWL,COND=(4,LT),
//         PARM='LIST,MAP,RENT,AMODE=31,RMODE=ANY'
//SYSLIB   DD DISP=SHR,DSN=CEE.SCEELKED
//SYSLIN   DD DSN=&&OBJ,DISP=(OLD,DELETE)
```

```
//SYSLMOD  DD DISP=SHR,DSN=BANK.DEV.LOADLIB(ACRBUNGA)
//SYSPRINT DD SYSOUT=*
```

Nama dataset compiler bergantung pada versi yang terpasang. Program dengan SQL menghasilkan DBRM yang harus di-BIND ke Db2 sebelum dijalankan.

## 75.1 Opsi compiler yang penting

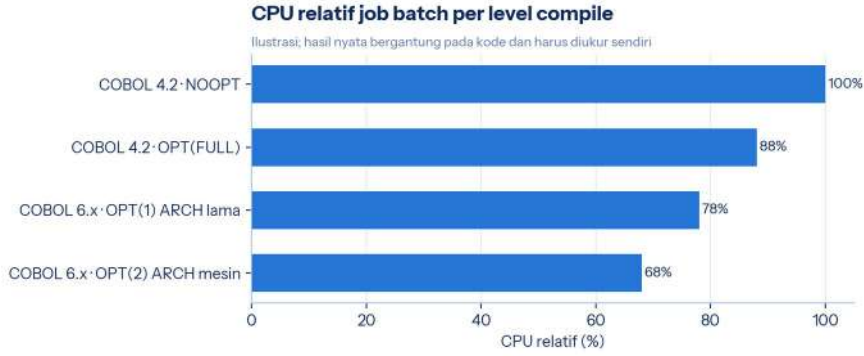
| Opsi | Fungsi | Rekomendasi |
|---|---|---|
| `ARCH(n)` | Target generasi prosesor | Setara mesin tertua yang menjalankan program, termasuk mesin DR |
| `OPT(0/1/2)` | Tingkat optimasi | `OPT(2)` produksi, `OPT(0)` saat debug |
| `LIST` atau `OFFSET` | Listing kode mesin atau tabel offset | Wajib diarsip untuk setiap versi produksi |
| `MAP`, `XREF` | Peta data dan referensi silang | Mempercepat membaca dump |
| `TEST(NOSEP)` | Informasi debug | Aktifkan agar CEEDUMP menampilkan nama variabel |
| `SSRANGE` | Cek batas subscript saat runtime | Lingkungan uji; pertimbangkan biayanya di produksi |
| `NUMCHECK` | Cek validitas data numerik | Menangkap data kotor lebih awal |
| `INITCHECK` | Peringatan data belum diinisialisasi | Tahap pengembangan |
| `SQL`, `CICS` | Coprocessor terintegrasi | Ganti precompiler terpisah |

Arsipkan listing compile untuk setiap load module produksi. Tanpa listing, offset di dump tidak bisa diterjemahkan menjadi baris kode.

## 75.2 Dampak rekompilasi pada CPU

Mengompilasi ulang program lama dengan compiler baru dan `ARCH` yang sesuai mesin sering memangkas CPU batch secara nyata. Ukur hasilnya

dari SMF tipe 30 sebelum dan sesudah, karena besarnya penghematan sangat bergantung pada pola kode.




*Ilustrasi · data rekaan*

## 75.3 Analisis: Opsi Compile yang Konsisten antara Uji dan Produksi

Program yang lolos uji tetapi berperilaku berbeda di produksi sering dikompilasi dengan opsi yang berbeda. Opsi compiler COBOL memengaruhi kinerja, pemeriksaan runtime, dan bahkan hasil aritmetika pada kasus tepi.

| Opsi | Pengaruh | Rekomendasi umum |
|---|---|---|
| `OPT` | Tingkat optimasi | Tingkat yang sama di uji akhir dan produksi |
| `ARCH` | Instruksi hardware yang boleh dipakai | Sesuai generasi CPC terlama tempat program mungkin berjalan, termasuk situs DR |
| `TRUNC` | Cara pemotongan field binary | Jangan diubah tanpa analisis; bisa mengubah hasil |
| `SSRANGE` | Pemeriksaan batas subscript saat runtime | Aktif di uji; di produksi sesuai kebijakan dan hasil ukur kinerja |
| `NUMCHECK` | Pemeriksaan data numerik tidak valid saat runtime | Membantu menangkap data kotor di uji (Bab 77) |
| `INITCHECK` | Peringatan compile untuk | Aktif di pipeline build |

| Opsi | Pengaruh | Rekomendasi umum |
|---|---|---|
| | data yang mungkin dipakai sebelum diisi | |

ARCH perlu perhatian khusus. Program yang dikompilasi untuk generasi hardware yang lebih baru mungkin memakai instruksi yang tidak ada di mesin lama. Jika situs DR memakai CPC generasi lebih lama daripada situs utama, program itu bisa gagal justru saat DR diaktifkan. Pilih ARCH berdasarkan mesin terlama di seluruh lingkungan produksi dan DR.

**Praktik yang disarankan:**

- Simpan opsi compile standar di satu tempat, misalnya prosedur compile bersama atau konfigurasi pipeline, bukan disalin di setiap JCL.
- Catat opsi yang dipakai untuk setiap load module yang di-deploy. Listing compile menyimpannya, dan listing itu juga dibutuhkan untuk analisis dump (Bab 79.3).
- Kompilasi ulang dengan opsi produksi sebelum uji akhir. Uji dengan build pengembangan tidak membuktikan perilaku build produksi.
- Tinjau opsi standar setiap kali compiler atau CPC di-upgrade, karena opsi baru sering tersedia dan nilai ARCH yang optimal ikut berubah.

# Bab 76. Language Environment

Language Environment (LE) adalah runtime bersama untuk COBOL, PL/I, C, dan sebagian Java. LE menangani storage, kondisi error, dan pembuatan dump saat program gagal.

| Opsi runtime LE | Fungsi | Rekomendasi produksi |
|---|---|---|
| `TRAP(ON)` | LE menangkap program check | Selalu `ON` |
| `TERMTHDACT(UADUMP)` | Tindakan saat terminasi abnormal: CEEDUMP plus system dump | `UADUMP` atau `DUMP` |

| Opsi runtime LE | Fungsi | Rekomendasi produksi |
|---|---|---|
| ABTERMENC(ABEND) | Akhiri dengan abend, bukan return code | ABEND agar scheduler melihat kegagalan |
| RPTOPTS(ON) | Laporkan opsi yang berlaku | Saat diagnosis |
| HEAP, STACK | Ukuran awal dan tambahan storage | Sesuaikan untuk program besar |
| ALL31(ON) | Semua storage di atas 16 MB | ON untuk program modern |

Opsi dapat diberikan per job lewat DD CEEOPTS:

```
//CEEOPTS  DD *
  TERMTHDACT(UADUMP),RPTOPTS(ON)
/*
```

## 76.1 Pesan LE untuk abend umum

| Pesan LE | Abend sistem | Arti |
|---|---|---|
| CEE3201S | S0C1 | Operation exception |
| CEE3204S | S0C4 | Protection exception |
| CEE3207S | S0C7 | Data exception |
| CEE3209S | S0C9 | Fixed-point divide |
| CEE3211S | S0CB | Decimal divide |

## 76.2 Membaca CEEDUMP

CEEDUMP adalah laporan teks yang dibuat LE, jauh lebih mudah dibaca daripada dump sistem.

1. **Traceback:** urutan program yang dipanggil, dengan nama program, offset, dan nomor statement bila TEST aktif.
2. **Condition information:** pesan CEE dan lokasi kegagalan.
3. **Local variables:** nilai variabel program saat gagal, termasuk isi hex field yang rusak.
4. **Storage:** isi area memori penting.

Mulailah dari baris traceback paling atas yang milik aplikasi, bukan modul LE atau sistem.

## 76.3 Analisis: Opsi Runtime LE yang Menentukan Kualitas Diagnosis

Ketika program COBOL gagal, informasi yang tersedia untuk diagnosis ditentukan oleh opsi runtime Language Environment yang berlaku saat itu. Opsi yang tepat bisa berarti perbedaan antara CEEDUMP lengkap yang langsung menunjuk baris kode, dan abend tanpa jejak yang harus direproduksi berjam-jam.

| Opsi | Fungsi | Pertimbangan |
|---|---|---|
| TERMTHDACT | Informasi yang dihasilkan saat program berakhir abnormal | TRACE untuk ringkas; DUMP atau UADUMP untuk CEEDUMP dan dump sistem |
| ABTERMENC | Program yang gagal berakhir dengan abend atau return code | ABEND agar kegagalan terlihat jelas oleh scheduler |
| RPTSTG | Laporan pemakaian storage saat program selesai | Aktifkan sementara untuk menyetel HEAP dan STACK |
| RPTOPTS | Laporan opsi runtime yang berlaku | Membuktikan opsi mana yang benar-benar aktif |
| HEAP, STACK | Ukuran awal dan penambahan storage | Disetel dari laporan RPTSTG, bukan ditebak |

Opsi runtime dapat berasal dari beberapa tempat: nilai default sistem, opsi di level region atau aplikasi, opsi yang dikompilasi ke dalam program, dan override lewat DD CEEOPTS di JCL batch. Urutan prioritasnya menentukan nilai yang benar-benar berlaku. Karena itu RPTOPTS(ON) sesekali berguna untuk membuktikan opsi mana yang aktif, terutama setelah perubahan default sistem.

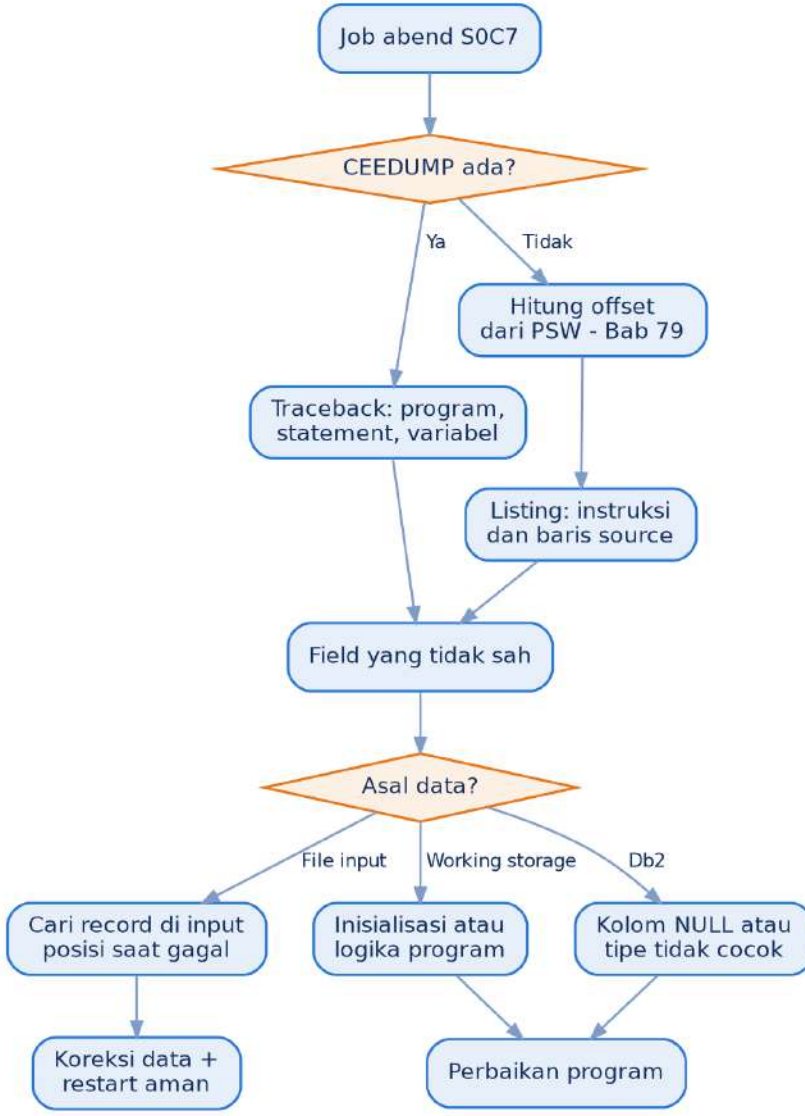
```
//CEEOPTS  DD *
RPTOPTS(ON),RPTSTG(ON),TERMTHDACT(UADUMP)
/*
```

Baris di atas ditambahkan sementara ke satu job batch untuk investigasi: menampilkan opsi yang aktif, laporan storage, dan dump yang lengkap bila job gagal. Setelah investigasi selesai, override dihapus agar job kembali ke opsi standar.

`ABTERMENC(ABEND)` layak menjadi standar untuk batch produksi. Program yang gagal dengan return code biasa bisa terlihat seperti “selesai dengan peringatan” oleh scheduler, sehingga rantai EOD berlanjut dengan data yang tidak lengkap. Abend yang jelas menghentikan rantai tepat di titik masalah.

## Bab 77. Algoritma Analisis S0C7

S0C7 adalah abend batch paling sering di core banking. Prosedur berikut hampir selalu menemukan akar masalahnya dalam 30 menit.




## 77.1 Pola cacat yang berulang

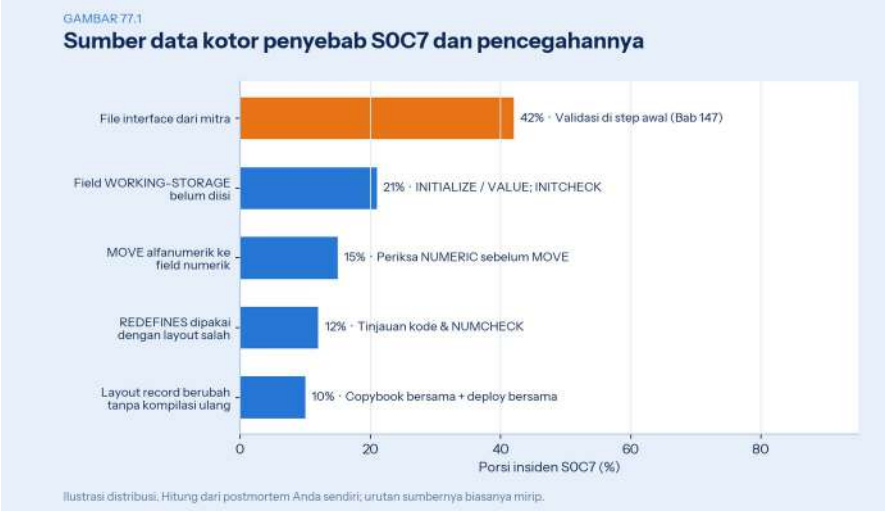
| Pola | Ciri di dump | Akar masalah tipikal |
|---|---|---|
| Spasi di field packed | Field berisi `40404040` | File interface dari sistem luar mengirim field kosong |

| Pola | Ciri di dump | Akar masalah tipikal |
|---|---|---|
| Nol biner | Field berisi `00000000` | Record pendek atau field tidak diisi |
| Pergeseran layout | Digit masuk akal tetapi di posisi salah | Copybook versi berbeda antara penulis dan pembaca file |
| Karakter ASCII | Nilai seperti `31323334` | File dikirim tanpa konversi EBCDIC |
| Indikator NULL diabaikan | Field host variable kosong | Kolom Db2 nullable tanpa indicator variable |

Setiap temuan harus menghasilkan dua tindakan: koreksi data untuk malam ini, dan validasi di hulu agar tidak terulang.

## 77.2 Analisis: Mencegah S0C7 di Sumbernya

Algoritma di bab ini membantu menemukan penyebab `S0C7` setelah terjadi. Namun analisis Pareto di Bab 49.1 menunjukkan `S0C7` sebagai salah satu abend yang paling sering. Upaya yang paling berharga adalah mencegahnya, dan pencegahan dimulai dari mengetahui dari mana data kotor datang.




*Gambar 77.1 — Sumber data kotor penyebab S0C7 dan pencegahannya*

File interface dari pihak luar biasanya menjadi sumber terbesar. Program COBOL membaca file itu dengan asumsi bahwa field nominal berisi packed decimal yang valid. Satu record dengan spasi atau karakter acak di field itu cukup untuk menghentikan seluruh job EOD.

| Sumber | Pencegahan | Di mana diterapkan |
|---|---|---|
| File interface | Validasi struktur dan field numerik sebelum diproses; karantina record rusak | Step awal setiap job yang membaca file luar (Bab 147.1) |
| Field belum diisi | `VALUE` atau `INITIALIZE` untuk semua field kerja; opsi `INITCHECK` saat compile | Standar koding dan pipeline build |
| MOVE alfanumerik ke numerik | Periksa `IF field IS NUMERIC` sebelum dipakai dalam aritmetika | Program yang menerima input bebas |
| REDEFINES dengan layout salah | Tinjauan kode; uji dengan semua varian record | Setiap perubahan layout |
| Layout record berubah | Copybook bersama; semua program pemakai dikompilasi dan di-deploy bersama | Proses rilis |

```
       IF WS-NOMINAL-IN IS NUMERIC
          MOVE WS-NOMINAL-IN TO WS-NOMINAL
       ELSE
          ADD 1 TO WS-JML-TOLAK
          PERFORM TULIS-KARANTINA
       END-IF
```

Pemeriksaan `IS NUMERIC` seperti di atas mengubah abend yang menghentikan job menjadi record yang dikarantina dan dilaporkan. Job tetap selesai, record bermasalah terdokumentasi, dan mitra dapat diberi tahu.

Opsi `NUMCHECK` di compiler COBOL modern dapat menangkap data numerik tidak valid saat runtime dengan pesan yang lebih jelas daripada `S0C7`. Opsi ini sangat berguna di lingkungan uji untuk menemukan program yang rentan sebelum sampai ke produksi (Bab 75.3).

# 🔬 Bagian XIX — HLASM dan Membaca Dump

Membaca dump adalah keterampilan yang memisahkan sysprog senior dari yang lain. Anda tidak perlu menulis program Assembler, tetapi perlu memahami register, PSW, dan beberapa instruksi dasar untuk menerjemahkan dump menjadi jawaban.

## Bab 78. Register dan PSW

Setiap prosesor IBM Z memiliki 16 general purpose register (GPR) 64-bit, bernomor 0 sampai 15. Beberapa register punya peran konvensional yang dipatuhi hampir semua program.

| Register | Peran konvensional | Kegunaan saat membaca dump |
|---|---|---|
| R0 | Parameter tambahan untuk layanan sistem | Sering berisi kode fungsi |
| R1 | Alamat daftar parameter | Menunjuk parameter yang diterima program |
| R12 | Base register (umum, bukan wajib) | Titik acuan program untuk alamat relatif |
| R13 | Alamat save area program saat ini | Titik awal menelusuri rantai pemanggilan |
| R14 | Alamat kembali ke pemanggil | Menunjukkan siapa yang memanggil |
| R15 | Alamat entry point saat dipanggil; return code saat kembali | Mengidentifikasi program yang dipanggil |

### 78.1 Isi PSW

PSW (*Program Status Word*) mencatat keadaan prosesor, termasuk alamat instruksi berikutnya. Saat abend, PSW adalah petunjuk pertama ke lokasi masalah.

| Bagian PSW | Arti |
|---|---|
| Storage key | Kunci proteksi memori yang berlaku |
| Problem/supervisor state | Program aplikasi atau kode sistem |
| Addressing mode | 24, 31, atau 64-bit |
| Condition code | Hasil perbandingan atau operasi terakhir |
| Instruction address | Alamat instruksi berikutnya yang akan dieksekusi |

Pada tampilan PSW 31-bit gaya lama seperti `078D1000 9A3F17B2`, bit tertinggi kata kedua menandai AMODE 31. Alamat sebenarnya adalah `1A3F17B2`.

## 78.2 Analisis: Membaca PSW Langkah demi Langkah

PSW (Program Status Word) mencatat keadaan prosesor saat sebuah instruksi dieksekusi, termasuk alamat instruksi berikutnya. Pada saat abend, PSW adalah petunjuk pertama ke lokasi kegagalan. Di z/Architecture, PSW berukuran 128 bit: bagian pertama berisi bit status, dan bagian kedua berisi alamat instruksi.

| Bagian | Isi yang penting untuk diagnosis |
|---|---|
| Bit mode pengalamatan | Program berjalan dalam mode 24, 31, atau 64 bit |
| Kunci storage dan status problem/supervisor | Program aplikasi biasa atau kode sistem |
| Condition code | Hasil perbandingan atau operasi terakhir |
| Alamat instruksi | Alamat instruksi berikutnya setelah yang gagal, untuk sebagian besar program check |

Langkah membaca PSW saat abend program check:

1. Ambil alamat instruksi dari PSW di laporan abend atau `STATUS FAILDATA`.
2. Kurangi dengan panjang instruksi (ILC × 2 byte) untuk mendapat alamat instruksi yang gagal, bila jenis interupsinya menunjuk instruksi sesudahnya.

3. Temukan modul yang memuat alamat itu, lalu hitung offset dari titik awal modul (Bab 79).
4. Periksa bit mode pengalamatan. Program 31-bit yang tiba-tiba berjalan dengan alamat di atas batas 2 GB, atau sebaliknya, menandakan kesalahan linkage.
5. Periksa status problem/supervisor. Kegagalan dalam mode supervisor biasanya berada di kode sistem atau exit, bukan di program aplikasi.

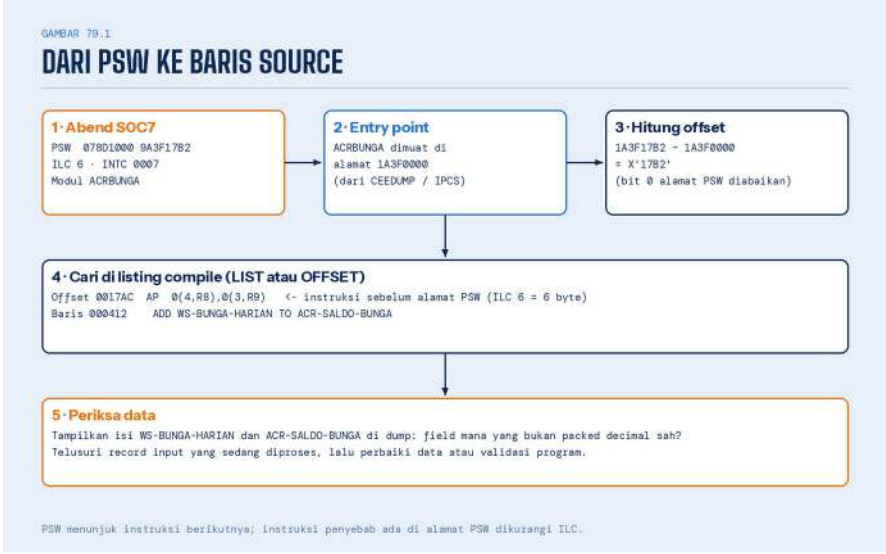
Langkah kedua sering terlewat. Untuk sebagian besar program check, PSW sudah menunjuk ke instruksi setelah yang gagal. Menganalisis instruksi di alamat PSW apa adanya berarti menganalisis instruksi yang salah. Panjang instruksi yang gagal tercatat dalam ILC, sehingga koreksinya mudah dan pasti.

Latih langkah-langkah ini dengan dump yang sengaja dibuat di lingkungan uji, misalnya program kecil yang memicu `S0C7` dengan data rusak. Sekali proses ini dilatih dengan tenang, membaca PSW dari dump produksi pukul dua pagi tidak lagi terasa menakutkan.

## Bab 79. Dari Abend ke Baris Source

PSW menunjuk instruksi *setelah* instruksi yang gagal untuk kebanyakan program interrupt. ILC (*Instruction Length Code*) memberi tahu panjang instruksi yang gagal, sehingga alamatnya bisa dihitung mundur.

Pengecualian penting: untuk interrupt yang di-*nullify*, seperti segment atau page translation (S0C4 reason 10 atau 11), PSW menunjuk instruksi yang gagal itu sendiri. Perintah IPCS `STATUS FAILDATA` sudah memperhitungkan hal ini, jadi utamakan hasilnya dibanding hitungan manual.




*Gambar 79.1 — Dari PSW ke baris source*

$$\text{offset} = (\text{alamat PSW} - \text{ILC}) - \text{entry point modul}$$

## 79.1 Kode interrupt program

| Kode | Abend | Nama | Penyebab umum |
|---|---|---|---|
| 0001 | S0C1 | Operation | Melompat ke alamat yang bukan kode |
| 0004 | S0C4 | Protection | Menulis ke memori milik key lain |
| 0005 | S0C5 | Addressing | Alamat di luar memori yang ada |
| 0006 | S0C6 | Specification | Operand tidak sejajar batas yang benar |
| 0007 | S0C7 | Data | Packed decimal tidak sah |
| 0009 | S0C9 | Fixed-point divide | Pembagian integer dengan nol |
| 000B | S0CB | Decimal divide | Pembagian desimal dengan nol |
| 0010, 0011 | S0C4 | Segment/page | Alamat tidak |

| Kode | Abend | Nama | Penyebab umum |
|---|---|---|---|
| | | translation | dipetakan; muncul sebagai S0C4 reason 10 atau 11 |

## 79.2 Instruksi yang sering muncul di dump

| Instruksi | Panjang | Fungsi |
|---|---|---|
| L / LG | 4 / 6 byte | Muat 32/64-bit dari memori ke register |
| ST / STG | 4 / 6 byte | Simpan register ke memori |
| LA | 4 byte | Muat alamat |
| MVC | 6 byte | Salin sampai 256 byte |
| CLC | 6 byte | Bandingkan dua area memori |
| PACK / UNPK | 6 byte | Konversi zoned ke packed dan sebaliknya |
| AP, SP, MP, DP, ZAP, CP | 6 byte | Aritmetika dan perbandingan packed decimal: sumber S0C7 |
| BASR / BRASL | 2 / 6 byte | Panggil subrutin |
| BR | 2 byte | Kembali (biasanya BR R14) |

Panjang instruksi dapat dibaca dari dua bit pertama opcode: `00` berarti 2 byte, `01` atau `10` berarti 4 byte, `11` berarti 6 byte.

## 79.3 Analisis: Listing Compile sebagai Aset Diagnosis

Menerjemahkan offset abend ke baris source membutuhkan listing compile dari versi program yang persis sama dengan yang sedang berjalan. Listing dari versi yang berbeda, walau hanya satu baris kode berubah, bisa menunjuk ke baris yang salah dan menyesatkan diagnosis.

| Kebutuhan | Praktik |
|---|---|
| Listing tersedia untuk setiap versi yang di-deploy | Pipeline build menyimpan listing sebagai artefak, bersama load module |

| Kebutuhan | Praktik |
|---|---|
| Listing dapat dicocokkan dengan load module | Simpan identitas build: waktu compile, versi commit Git, dan nama load module |
| Listing memuat peta offset | Opsi listing yang menghasilkan pemetaan offset ke statement aktif di build produksi |
| Retensi | Listing disimpan selama load module mungkin masih berjalan, termasuk versi lama untuk rollback |

```
ALGORITMA OffsetKeBarisSource(dump)
  modul, offset <- dari CEEDUMP atau STATUS FAILDATA
  waktu_link <- stempel waktu load module di dump atau
dari daftar library
  listing <- artefak build dengan nama modul dan waktu
compile yang cocok
  jika tidak ada listing yang cocok maka berhenti:
kompilasi ulang dari commit yang sama
      dengan opsi yang sama, lalu cocokkan ukuran modul
  cari offset di peta listing; ambil nomor statement dan
baris source
  periksa isi field di sekitar baris itu dari dump (Bab
77)
```

Langkah “cocokkan waktu compile” sering dilewatkan, padahal paling penting. Tim yang mencari listing di library source hampir selalu menemukan versi terbaru, bukan versi yang sedang berjalan di produksi. Bila program sudah diubah tetapi belum di-deploy, listing terbaru tidak cocok dengan modul di produksi.

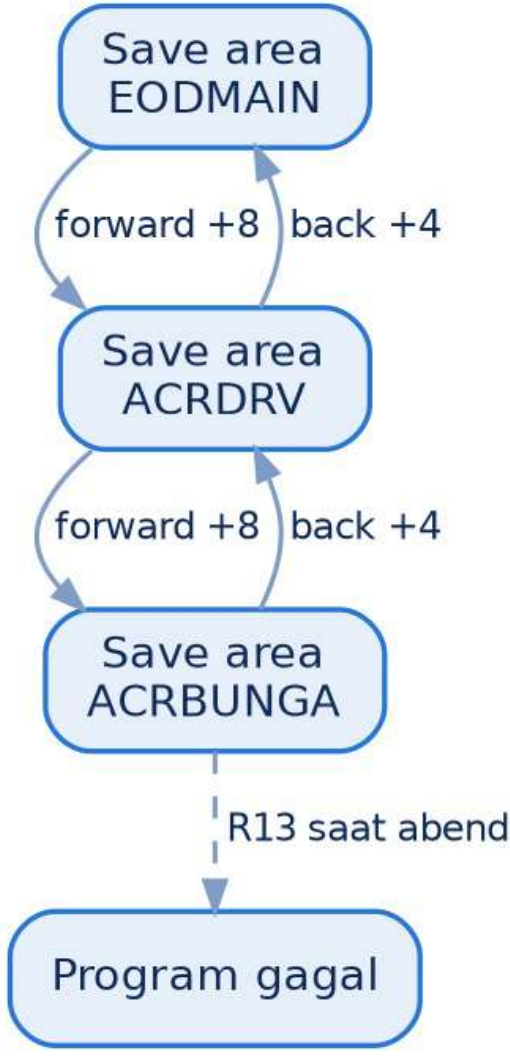
Pipeline modern (Bab 87.1) memudahkan ini: setiap build otomatis menyimpan load module, listing, dan identitas commit bersamaan. Pada saat insiden, mencari listing yang tepat menjadi pencarian sederhana berdasarkan nama modul dan waktu build, bukan penggalian di arsip.

## Bab 80. Konvensi Linkage dan Rantai Save Area

Program yang dipanggil menyimpan register pemanggilnya di save area, lalu menautkan save area-nya sendiri ke pemanggil. Rantai ini memungkinkan dump ditelusuri mundur dari program yang gagal sampai ke program utama.

```
ACRUTIL  CSECT
ACRUTIL  AMODE 31
ACRUTIL  RMODE ANY
         YREGS ,                   Definisi R0-R15
         STM   R14,R12,12(R13)     Simpan register
pemanggil
         LR    R12,R15             Base register = entry
point
         USING ACRUTIL,R12
         LA    R11,SAVEAREA
         ST    R13,4(,R11)         Back chain ke save area
pemanggil
         ST    R11,8(,R13)         Forward chain dari
pemanggil
         LR    R13,R11             Save area milik program
ini
*        ... logika program ...
         L     R13,4(,R13)         Kembali ke save area
pemanggil
         LM    R14,R12,12(R13)     Pulihkan register
pemanggil
         SR    R15,R15             Return code 0
         BR    R14                 Kembali
SAVEAREA DS    18F                 Save area standar 72
byte
         END   ACRUTIL
```

Contoh ini tidak reentrant dan hanya untuk ilustrasi. Program produksi yang reentrant mengambil save area lewat `STORAGE OBTAIN` atau makro sejenis.




Di dump, mulai dari R13 saat abend, ikuti pointer back chain di offset +4 untuk menemukan urutan pemanggilan dan register setiap program.

## 80.1 Analisis: Membaca Rantai Save Area yang Rusak

Rantai save area mencatat siapa memanggil siapa. Setiap program yang dipanggil menyimpan register pemanggilnya di save area, lalu menautkan save area itu ke save area pemanggil dan ke save area berikutnya. Dengan mengikuti rantai ini ke belakang, jalur pemanggilan sampai ke titik kegagalan dapat direkonstruksi.

| Format | Ukuran | Pengenalan |
|---|---|---|
| Format standar 31-bit | 72 byte | Tautan mundur dan maju di awal save area |
| Format 64-bit (F4SA) | 144 byte | Penanda format di awal; register 64 bit |

Register 13 menunjuk save area program yang sedang berjalan. Dari sana, tautan mundur menunjuk save area pemanggil, dan seterusnya sampai ke awal rantai.

**Tanda rantai rusak:**

| Tanda | Kemungkinan penyebab |
|---|---|
| Tautan menunjuk alamat di luar storage yang dipakai program | Save area ditimpa oleh penulisan melewati batas array |
| Tautan mundur dan maju tidak saling cocok | Program tidak mengikuti konvensi linkage |
| Rantai berputar kembali ke save area yang sama | Tautan ditimpa dengan nilai yang salah |
| Format berganti di tengah rantai tanpa penanda yang benar | Campuran program 31 dan 64 bit dengan linkage yang tidak tepat |

Rantai yang rusak adalah petunjuk tersendiri. Jika rantai rusak tepat di save area program tertentu, program itu atau program yang dipanggilnya kemungkinan besar menulis melewati batas storage-nya. Penyebab yang sering ditemukan adalah subscript di luar batas pada array di WORKING-STORAGE yang letaknya berdekatan dengan save area. Mengaktifkan `SSRANGE` di lingkungan uji dapat mengonfirmasi dugaan ini (Bab 75.3).

IPCS dapat menampilkan rantai save area secara otomatis dari dump, sehingga analisis manual jarang diperlukan. Namun memahami

strukturnya membantu mengenali saat tampilan otomatis terlihat janggal, dan saat itulah biasanya ditemukan akar masalahnya.

# Bab 81. Studi Kasus Analisis Dump

Skenario: job `EODGL030` abend `S0C4` reason `11` pukul 01.47 di tengah posting General Ledger. Dump SVC tersedia di dataset dump sistem.

## 81.1 Langkah analisis di IPCS

```
STATUS FAILDATA              PSW, register, modul, dan
offset kegagalan
WHERE 1A40C21E               Modul mana yang memuat alamat
ini
LIST 1A40C200 LENGTH(64)     Isi memori di sekitar
instruksi
SUMMARY FORMAT JOBNAME(EODGL030)   TCB, RB, dan save area
VERBX LEDATA 'CEEDUMP'       Laporan LE di dalam dump
sistem
```

## 81.2 Temuan dan kesimpulan

1. `STATUS FAILDATA` menunjukkan modul `GLPOST`, offset `X'0C218'`, instruksi `L R5,0(,R7)`.
2. R7 berisi `00000000`: program memuat data dari alamat nol, sehingga terjadi page translation exception.
3. Listing menunjukkan R7 seharusnya berisi alamat tabel kurs yang dimuat di awal job.
4. SYSOUT memperlihatkan pemuatan tabel kurs gagal karena dataset kosong, tetapi program tidak memeriksa return code.
5. Akar masalah: job pengisi tabel kurs dari treasury terlambat, dan dependensi di scheduler tidak didefinisikan.

Perbaikan malam itu adalah menjalankan job kurs lalu me-restart `EODGL030`. Perbaikan permanen adalah dependensi scheduler dan validasi return code di program.

## 81.3 Analisis: Pola Tanda Tangan Dump yang Sering Muncul

Setelah membaca cukup banyak dump, pola-pola tertentu mulai terlihat berulang. Mengenali pola ini mempercepat diagnosis, karena setiap pola langsung menunjuk ke tempat yang perlu diperiksa lebih dulu.

| Tanda tangan | Pola di dump | Penyebab yang paling mungkin | Periksa lebih dulu |
|---|---|---|---|
| `S0C7` di instruksi aritmetika desimal | Field operand berisi spasi atau byte non-desimal | Data kotor dari input atau field belum diisi | Isi field di dump dan asal datanya (Bab 77.2) |
| `S0C4` dengan alamat di luar storage program | Register basis berisi nilai tak masuk akal | Pointer atau subscript rusak | Array di dekat area yang tertimpa; `SSRANGE` di uji |
| `S0C1` dengan PSW menunjuk area data | Program melompat ke alamat yang bukan kode | Save area atau alamat kembali tertimpa | Rantai save area (Bab 80.1) |
| `S0C4` di awal program yang baru di-deploy | Modul dipanggil dengan mode pengalamatan atau parameter yang salah | Ketidakcocokan linkage antar-program | Atribut link-edit dan cara pemanggilan |
| `S878` atau `S80A` | Permintaan memori tidak terpenuhi | Region terlalu kecil atau kebocoran memori | Laporan storage LE, pemakaian memori job (Bab 10.4) |
| Abend di modul sistem dengan kode reason tertentu | Bukan di kode aplikasi | Masalah produk yang mungkin sudah dikenal | Cari kombinasi kode, modul, dan offset di basis pengetahuan IBM (Bab 51.1) |

Pola-pola ini adalah titik awal, bukan kesimpulan. Setiap dugaan tetap harus dibuktikan dengan isi dump: nilai register, isi field, dan instruksi di alamat kegagalan. Namun titik awal yang tepat bisa memangkas waktu analisis dari berjam-jam menjadi beberapa menit.

Simpan setiap dump yang sudah dianalisis bersama kesimpulannya di basis pengetahuan internal, lengkap dengan pola yang dikenali. Setelah beberapa tahun, koleksi ini menjadi materi pelatihan yang sangat berharga

bagi anggota tim baru. Materi seperti ini juga jarang tersedia di dokumentasi mana pun, karena berasal dari lingkungan Anda sendiri.

# 🤖 Bagian XX — REXX Lanjutan dan Otomasi

Bab 26 memperkenalkan REXX. Bagian ini membangun kemampuan itu sampai tingkat yang cukup untuk otomasi produksi: membaca dan menulis dataset, menangkap output perintah, memanggil utilitas, berinteraksi dengan console, dan menangani error dengan disiplin.

## Bab 82. Parsing dan Struktur Data

PARSE adalah fitur REXX yang paling kuat untuk system engineer, karena hampir semua otomasi berarti membaca pesan teks dan mengambil angka dari dalamnya.

```
/* Mengambil nilai dari pesan $HASP893 */
baris = '$HASP893 VOLUME(SPOOL1) STATUS=ACTIVE,PERCENT=37'
parse var baris . 'VOLUME(' vol ')' . 'PERCENT=' pct .
say vol pct                        /* SPOOL1 37 */

/* Memecah nama dataset menjadi qualifier */
dsn = 'BANK.PROD.CIF.MASTER'
parse var dsn hlq '.' env '.' rest
say hlq env rest                   /* BANK PROD CIF.MASTER
*/

/* Template posisi kolom untuk laporan lebar tetap */
parse var rec 1 norek 17 . 30 saldo 45 .
```

| Fungsi | Kegunaan |
|---|---|
| `WORD`, `WORDS`, `SUBWORD` | Mengambil kata dari baris pesan |
| `POS`, `LASTPOS` | Mencari teks di dalam string |
| `SUBSTR`, `LEFT`, `RIGHT`, `STRIP` | Memotong dan merapikan |
| `TRANSLATE`, `SPACE` | Mengubah karakter, merapikan spasi |
| `C2X`, `X2C`, `X2D`, `D2X` | Konversi hex, berguna saat membaca dump atau field packed |
| `DATATYPE(x,'N')` | Memeriksa apakah nilai numerik |

| Fungsi | Kegunaan |
|---|---|
| DATE, TIME | Stempel waktu untuk log |

Stem (`nama.`) adalah struktur data utama REXX. Konvensi yang dipakai semua perintah sistem adalah `nama.0` berisi jumlah elemen.

## 82.1 Analisis: Kinerja Skrip REXX pada Data Besar

REXX sangat nyaman untuk skrip kecil. Skrip yang sama bisa menjadi sangat lambat dan boros memori ketika dipakai untuk memproses file jutaan record atau output perintah yang panjang. Beberapa pola sederhana membuat perbedaan besar.

| Pola lambat | Pola cepat | Alasan |
|---|---|---|
| Membaca seluruh file ke stem sekaligus (`EXECIO * DISKR`) | Membaca per blok, misalnya 1.000 record sekali baca | Memori tetap kecil, berapa pun ukuran file |
| Menyambung string di dalam loop (`hasil = hasil baris`) | Menulis ke stem, lalu menulis semuanya sekali | Penyambungan berulang menyalin string berkali-kali |
| `POS` dan `SUBSTR` berulang untuk memecah baris | Satu `PARSE` dengan template | `PARSE` memecah baris dalam satu langkah |
| Memanggil perintah TSO untuk setiap record | Mengelompokkan kerja, satu perintah untuk banyak item | Setiap perintah eksternal punya overhead besar |
| Mencari di daftar dengan loop | Stem sebagai tabel kunci (`ada.kunci = 1`) | Pencarian langsung, tidak perlu loop |

```
/* Membaca file besar per blok 1.000 record */
"EXECIO 0 DISKR IN (OPEN"
DO FOREVER
  "EXECIO 1000 DISKR IN (STEM rec."
  bacaRC = rc
  DO i = 1 TO rec.0
    PARSE VAR rec.i rekening 17 . 30 nominal 38 .
    /* olah satu record */
  END
  IF bacaRC = 2 THEN LEAVE   /* akhir file */
END
"EXECIO 0 DISKR IN (FINIS"
```

Pola stem sebagai tabel kunci sangat berguna untuk rekonsiliasi kecil. Muat daftar referensi dari satu file ke `ada.referensi = 1`, lalu baca file kedua dan periksa `IF ada.ref = 1`. Pencarian ini tidak bergantung pada jumlah item, sehingga tetap cepat untuk ratusan ribu referensi.

Untuk volume yang sangat besar, misalnya puluhan juta record, pertimbangkan DFSORT atau ICETOOL (Bab 20.1), atau program yang dikompilasi. REXX paling tepat untuk logika pengendali dan pengolahan data berukuran sedang, sedangkan pekerjaan berat dengan volume besar lebih efisien dikerjakan alat yang dirancang untuk itu.

# Bab 83. Membaca Dataset dan Menangkap Output

## 83.1 EXECIO untuk dataset

```
"ALLOC F(INDD) DA('BANK.OPS.AMBANG') SHR REUSE"
"EXECIO * DISKR INDD (STEM ln. FINIS"
"FREE F(INDD)"
do i = 1 to ln.0
  parse var ln.i kunci nilai .
  ambang.kunci = nilai            /* contoh: ambang.SPOOL
= 75 */
end

out.1 = 'HEALTH CHECK' date('S') time()
out.0 = 1
"ALLOC F(OUTDD) DA('BANK.OPS.LAPORAN') MOD REUSE"
"EXECIO" out.0 "DISKW OUTDD (STEM out. FINIS"
"FREE F(OUTDD)"
```

Menyimpan ambang di dataset parameter, bukan di dalam kode, membuat operator bisa mengubah ambang tanpa menyentuh skrip.

## 83.2 OUTTRAP untuk output perintah TSO

```
call outtrap 'lc.'
"LISTCAT LEVEL(BANK.PROD) NAME"
call outtrap 'OFF'
jml = 0
do i = 1 to lc.0
  if word(lc.i, 1) = 'NONVSAM' | word(lc.i, 1) = 'CLUSTER'
```

```
then jml = jml + 1
end
say 'Jumlah dataset BANK.PROD:' jml
```

## 83.3 Memanggil utilitas batch dari REXX

```
"ALLOC F(SYSIN) NEW REUSE UNIT(SYSALLDA) SPACE(1 1) TRACKS
RECFM(F B) LRECL(80)"
"ALLOC F(SYSPRINT) NEW REUSE UNIT(SYSALLDA) SPACE(5 5)
TRACKS RECFM(V B A) LRECL(137)"
cmd.1 = " LISTCAT ENTRIES('BANK.PROD.CIF.MASTER') ALL"
cmd.0 = 1
"EXECIO 1 DISKW SYSIN (STEM cmd. FINIS"
address LINKMVS "IDCAMS"
say 'IDCAMS RC =' rc
"EXECIO * DISKR SYSPRINT (STEM pr. FINIS"
"FREE F(SYSIN SYSPRINT)"
```

## 83.4 Analisis: Memilih Cara Menangkap Output

Skrip otomasi sering perlu membaca output perintah: daftar dataset, status job, atau hasil perintah console. Ada beberapa cara menangkap output, dan memilih yang salah membuat skrip rapuh atau tidak berfungsi.

| Sumber output | Cara yang tepat | Catatan |
|---|---|---|
| Perintah TSO (misalnya `LISTCAT`, `LISTDS`) | `OUTTRAP` | Hanya menangkap output perintah TSO, bukan console |
| Perintah console (`D A,L`, `D GRS,C`) | Antarmuka console seperti SDSF REXX (`ISFSLASH`) atau console extended MCS | `OUTTRAP` tidak menangkap respons console |
| Status job dan output spool | SDSF REXX | Akses terstruktur ke panel seperti `ST` dan `DA` |
| Data dari sistem lain atau pipeline | Zowe CLI dengan output JSON (Bab 87.1) | Diolah di luar mainframe |

Kesalahan paling umum adalah mencoba menangkap respons perintah console dengan `OUTTRAP`. Perintah dikirim, tetapi responsnya muncul di console, bukan di variabel skrip. Skrip lalu mengira perintah tidak menghasilkan apa-apa.

**Kerapuhan parsing.** Output yang dibaca sebagai teks bergantung pada format tampilan. Satu kolom yang bergeser setelah upgrade bisa membuat skrip membaca angka yang salah tanpa error. Beberapa cara mengurangi risiko ini:

- Utamakan antarmuka terstruktur, seperti variabel kolom di SDSF REXX, daripada memotong teks berdasarkan posisi.
- Cocokkan baris berdasarkan ID pesan atau kata kunci, bukan nomor baris.
- Validasi hasil parsing, misalnya angka memang numerik dan dalam rentang wajar, sebelum dipakai untuk keputusan.
- Uji ulang skrip setelah setiap upgrade z/OS atau produk yang formatnya dibaca skrip.

Skrip yang membaca output untuk membuat keputusan otomatis (Bab 85) harus paling ketat dalam hal ini. Parsing yang salah dalam laporan hanya menghasilkan laporan yang salah. Parsing yang salah dalam otomasi bisa menghasilkan tindakan yang salah.

## Bab 84. Penanganan Error yang Disiplin

Skrip otomasi yang gagal diam-diam lebih berbahaya daripada tidak ada skrip. Setiap skrip produksi wajib menangkap error dan mengembalikan return code yang bermakna ke scheduler.

```
signal on novalue name TANGKAP     /* variabel belum diisi
*/
signal on syntax  name TANGKAP     /* kesalahan sintaks */
call on error     name GAGALCMD    /* perintah host RC <>
0 */
...
exit maxrc

TANGKAP:
  say 'ERROR' condition('C') 'di baris' sigl ':'
sourceline(sigl)
  exit 16

GAGALCMD:
  say 'Perintah gagal RC='rc 'di baris' sigl
```

```
  maxrc = max(maxrc, 8)
  return
```

| Return code skrip | Arti bagi scheduler |
|---|---|
| 0 | Semua pemeriksaan normal |
| 4 | Ada peringatan; lanjutkan, kirim notifikasi |
| 8 | Kondisi kritis; tahan job berikutnya, eskalasi |
| 16 | Skrip sendiri gagal; hasil tidak dapat dipercaya |

## 84.1 Analisis: Pola Penanganan Error REXX yang Bisa Dipakai Ulang

Skrip REXX untuk operasi sering ditulis cepat untuk satu kebutuhan, lalu dipakai bertahun-tahun oleh orang lain. Skrip tanpa penanganan error yang disiplin bisa gagal diam-diam: perintah gagal, variabel kosong, dan skrip melanjutkan seolah semuanya baik. Pola standar yang dipakai di semua skrip mencegah hal ini.

```
/* REXX - kerangka standar skrip operasi */
SIGNAL ON SYNTAX  NAME Gagal
SIGNAL ON NOVALUE NAME Gagal
SIGNAL ON FAILURE NAME Gagal
NUMERIC DIGITS 15
namaSkrip = 'SPOOLCHK'
rcAkhir = 0

/* ---- langkah kerja: setiap perintah diperiksa RC-nya
---- */
"ALLOC FI(IN) DA('HLQ.OPS.PARM') SHR REUSE"
IF rc <> 0 THEN CALL Berhenti 12, 'Gagal alokasi
parameter, RC='rc
/* ... logika utama ... */

CALL Selesai rcAkhir

Berhenti:
  PARSE ARG kode, pesan
  SAY namaSkrip 'ERROR' pesan
  CALL Selesai kode
```

```
Gagal:
  SAY namaSkrip 'KESALAHAN' condition('C') 'di baris' sigl
  SAY '  ' sourceline(sigl)
  CALL Selesai 16

Selesai:
  PARSE ARG kodeKeluar
  "FREE FI(IN)"
  SAY namaSkrip 'SELESAI RC='kodeKeluar
  EXIT kodeKeluar
```

| Unsur | Fungsi |
|---|---|
| `SIGNAL ON NOVALUE` | Menangkap variabel yang dipakai sebelum diisi, penyebab bug paling umum di REXX |
| `SIGNAL ON SYNTAX` | Menangkap kesalahan sintaks dan menampilkan baris penyebabnya |
| Pemeriksaan `rc` setiap perintah | Kegagalan perintah eksternal tidak diabaikan |
| Satu titik keluar | Sumber daya selalu dibebaskan, dan return code selalu dilaporkan |
| Konvensi return code | 0 sukses, 4 peringatan, 8 gagal sebagian, 12 atau lebih gagal |

Konvensi return code yang konsisten membuat skrip REXX dapat dipakai dari scheduler seperti program batch lainnya. Scheduler dapat menghentikan rantai bila return code melewati ambang, dan tidak ada kegagalan yang tersembunyi di dalam output yang tidak dibaca siapa pun.

## Bab 85. Otomasi Console

Ada tiga jalur utama bagi REXX untuk memberi perintah console dan membaca hasilnya.

| Jalur | Berjalan di | Kelebihan | Catatan |
|---|---|---|---|
| SDSF REXX (`ISFEXEC`, `ISFSLASH`) | TSO atau batch | Mudah, bisa membaca panel SDSF | Hak dikendalikan class RACF `SDSF` dan `OPERCMDS` |
| System REXX (AXR) | Address space AXR | Dapat dipicu dari console: `F AXR,nama` | Exec disimpan di `SYS1.SAXREXEC` atau konkatenasi |

| Jalur | Berjalan di | Kelebihan | Catatan |
|---|---|---|---|
| | | | AXRxx |
| TSO `CONSOLE` | TSO | Sesi console extended | Butuh otorisasi `CONSOLE` di TSOAUTH |

SDSF REXX juga dapat bertindak atas baris panel, misalnya membatalkan job atau mengubah class output lewat `ISFACT`. Batasi hak ini untuk user otomasi saja.

## 85.1 Analisis: Merancang Otomasi Berbasis Pesan

Otomasi console bekerja dengan mendengarkan pesan sistem, lalu bertindak saat pesan tertentu muncul. Pola ini menghemat banyak kerja manual, tetapi juga bisa bertindak di saat yang salah bila aturannya terlalu longgar.


*Gambar 85.1 — Otomasi console berbasis pesan*

Setiap aturan otomasi sebaiknya menjawab lima pertanyaan secara tertulis:

| Pertanyaan | Contoh untuk WTOR mount tape rutin |
|---|---|
| Pesan apa yang memicu, dan bagaimana pencocokannya? | ID pesan dan nama job tertentu, bukan hanya ID pesan |
| Tindakan apa yang dilakukan? | Menjawab dengan nilai tetap yang sudah disetujui |
| Kapan aturan tidak boleh bertindak? | Di luar jam batch, atau bila job tidak ada di daftar |
| Berapa kali boleh bertindak dalam satu periode? | Maksimal tiga kali per jam; lebih dari itu, alert ke manusia |
| Bagaimana tindakan dicatat? | Waktu, pesan pemicu, tindakan, dan hasilnya |

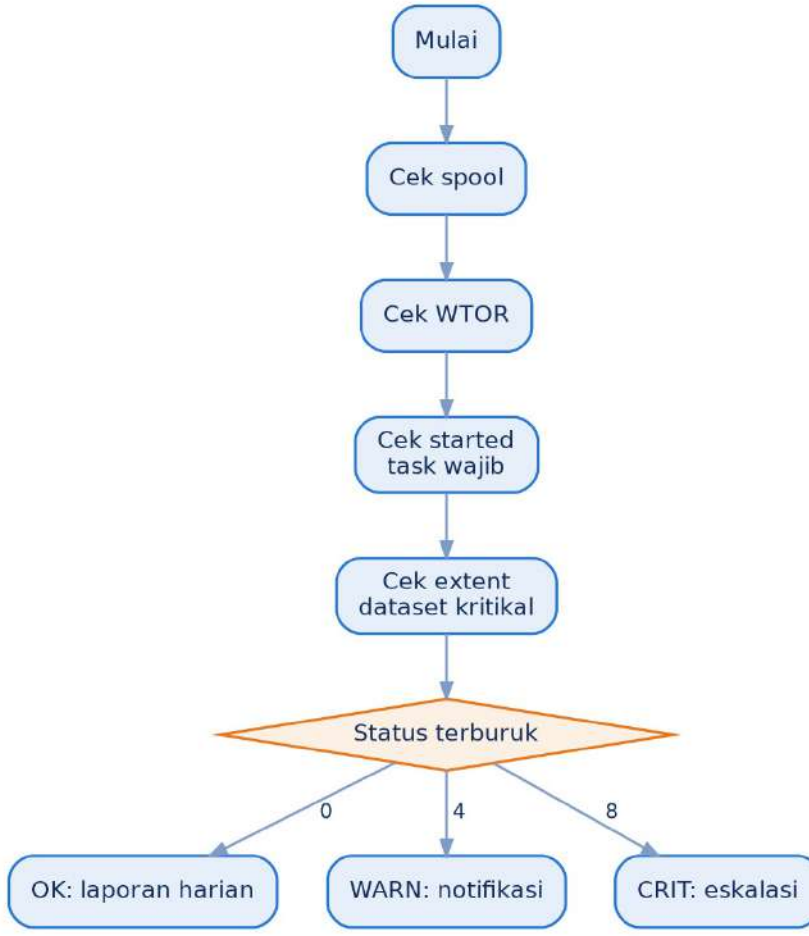
Pencocokan yang terlalu longgar adalah kesalahan paling berbahaya. Aturan yang menjawab semua WTOR dengan ID tertentu tanpa memeriksa nama job bisa menjawab WTOR dari job lain yang sebenarnya membutuhkan keputusan manusia.

Pesan yang tidak perlu dilihat operator sama sekali bisa disembunyikan dari console lewat `MPFLSTxx`, tetapi tetap tercatat di log. Pesan yang membutuhkan tindakan sebaiknya disorot, dan bila memungkinkan diteruskan otomatis sebagai alert dengan konteksnya.

Pantau otomasi itu sendiri. Laporan mingguan berisi jumlah tindakan per aturan, jumlah kejadian yang ditolak karena batas laju, dan jumlah alert ke manusia menunjukkan apakah aturan bekerja sesuai harapan. Aturan yang tiba-tiba bertindak jauh lebih sering dari biasanya adalah tanda ada masalah baru di sistem, bukan tanda otomasi bekerja dengan baik.

## Bab 86. Skrip Health Check Harian Lengkap

Skrip ini adalah jawaban proyek latihan di Lampiran C. Skrip memeriksa spool, WTOR yang menunggu, started task wajib, dan extent dataset kritikal, lalu menghasilkan satu status keseluruhan.




```
/* REXX - HLTHCHK: health check harian z/OS
*/
/* RC 0 = OK, 4 = WARN, 8 = CRIT, 16 = skrip gagal
*/
signal on syntax name TANGKAP
maxrc = 0
wajib = 'DB1AMSTR DB1ADBM1 CAPRD05 CTPRD01 TCPIP CSQ1MSTR'
dsnkrit = "'BANK.PROD.CIF.EXTRACT' 'BANK.PROD.GL.EXTRACT'"
rc = isfcalls('ON')

/* 1. Spool JES2 */
Address SDSF "ISFSLASH '$D SPOOL' (WAIT)"
maxpct = 0
```

```
do i = 1 to isfulog.0
  if pos('PERCENT=', isfulog.i) > 0 then do
    parse var isfulog.i . 'PERCENT=' pct .
    pct = strip(pct, 'T', ',')
    if datatype(pct, 'W') then maxpct = max(maxpct, pct)
  end
end
call LAPOR 'SPOOL', maxpct'%', NILAI(maxpct, 60, 75)

/* 2. WTOR yang menunggu balasan */
Address SDSF "ISFEXEC SR"
call LAPOR 'WTOR', isfrows 'pesan', NILAI(isfrows, 3, 10)

/* 3. Started task wajib aktif */
isfprefix = '*'; isfowner = '*'
Address SDSF "ISFEXEC DA"
aktif = ''
do i = 1 to jname.0
  aktif = aktif jname.i
end
do i = 1 to words(wajib)
  nm = word(wajib, i)
  if wordpos(nm, aktif) = 0 then call LAPOR 'STC', nm
'TIDAK AKTIF', 8
end

/* 4. Extent dataset kritikal */
do i = 1 to words(dsnkrit)
  d = word(dsnkrit, i)
  if listdsi(d) = 0 then call LAPOR 'EXTENT', d
sysextents, NILAI(sysextents, 10, 13)
  else call LAPOR 'EXTENT', d 'LISTDSI RC' sysreason, 4
end

rc = isfcalls('OFF')
say 'STATUS AKHIR:' word('OK WARN CRIT', min(3, maxrc % 4
+ 1))
exit maxrc

NILAI: procedure
  parse arg v, warn, crit
  if v >= crit then return 8
  if v >= warn then return 4
  return 0
```

```
LAPOR:
  parse arg area, info, lvl
  say left(area, 8) left(word('OK WARN CRIT', lvl % 4 +
1), 5) info
  maxrc = max(maxrc, lvl)
  return

TANGKAP:
  say 'HLTHCHK GAGAL di baris' sigl
  exit 16
```

Jalankan setiap pagi lewat JCL `IKJEFT01` (Bab 26.3) dan arahkan SYSTSPRT ke sistem output management. Untuk mengirim hasilnya lewat email, arahkan output ke external writer CSSMTP sesuai konfigurasi SMTP di sistem Anda.

## 86.1 Analisis: Health Check yang Hemat dan Mudah Dibaca

Skrip health check harian cenderung tumbuh. Setiap insiden menambah satu pemeriksaan baru, dan setelah beberapa tahun skrip berjalan puluhan menit dan menghasilkan laporan puluhan halaman yang tidak dibaca siapa pun. Health check yang efektif dirancang untuk tetap hemat dan tetap dibaca.

| Prinsip | Penerapan |
|---|---|
| Laporkan penyimpangan, bukan semua angka | Baris OK diringkas; hanya WARN dan CRIT yang ditampilkan rinci (Bab 181.1) |
| Bandingkan dengan kemarin | Tampilkan perubahan: “spool naik dari 42% ke 61%”, bukan hanya “61%” |
| Ambang dari data | Ambang disetel dari riwayat, bukan ditebak (Bab 105.2) |
| Setiap pemeriksaan punya pemilik | Pemeriksaan yang tidak ada pemiliknya dihapus atau diberi pemilik |
| Biaya pemeriksaan terukur | Waktu dan CPU setiap pemeriksaan dicatat; yang mahal dijalankan lebih jarang |

$$\text{Nilai pemeriksaan} \approx \frac{\text{insiden yang pernah dicegah atau dideteksi lebih awal}}{\text{biaya menjalankan dan membaca}}$$

Rumus ini tidak untuk dihitung persis, tetapi untuk dipakai saat meninjau daftar pemeriksaan setahun sekali. Pemeriksaan yang tidak pernah menemukan apa pun selama bertahun-tahun, dan memeriksa sesuatu yang sudah dipantau di tempat lain, adalah kandidat untuk dihapus. Pemeriksaan yang berkali-kali menangkap masalah lebih awal layak dipertahankan, bahkan dijalankan lebih sering.

Bedakan pemeriksaan yang cukup sekali sehari dari yang perlu terus-menerus. Status replikasi, kedalaman antrean kritikal, dan jalur sinyal sysplex terlalu penting untuk hanya diperiksa pagi hari. Kondisi seperti itu lebih tepat dipantau terus oleh sistem monitoring dengan alert (Bab 183). Health check pagi hari paling tepat untuk hal-hal yang berubah lambat: tren kapasitas, sertifikat, konfigurasi, dan hasil EOD.

# Bab 87. Zowe CLI untuk Otomasi dari Luar Mainframe

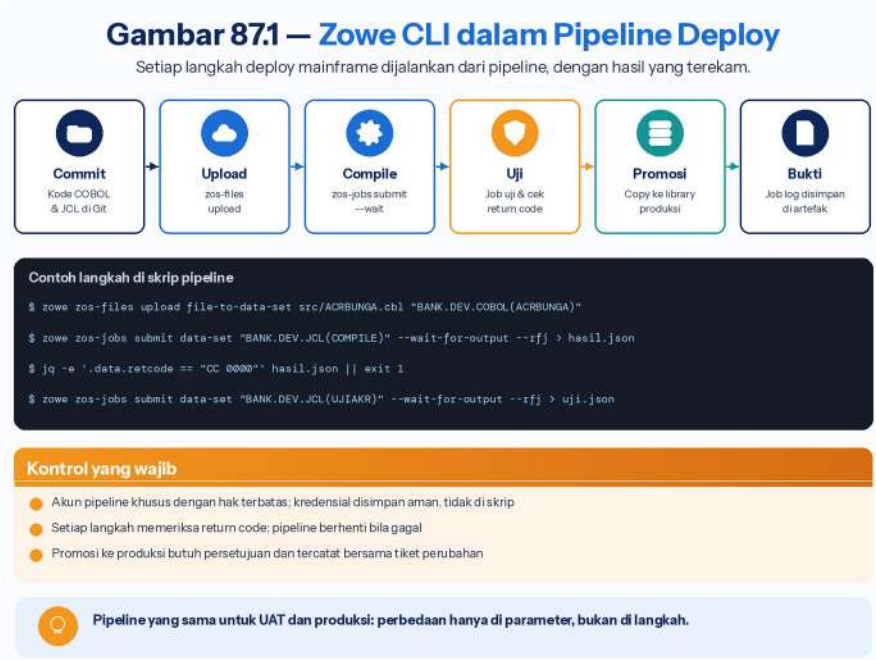
Zowe CLI memungkinkan skrip di laptop, server CI, atau alat monitoring menjalankan tugas z/OS tanpa emulator 3270.

```
zowe zos-console issue command "D A,L"
zowe zos-jobs submit data-set "BANK.OPS.JCL(HLTHCHK)" --
wait-for-output --view-all-spool-content
zowe zos-files list data-set "BANK.PROD.*"
zowe zos-files download data-set "BANK.OPS.LAPORAN" -f
laporan.txt
```

Gabungkan Zowe CLI dengan pipeline CI (Bab 55) agar health check, deploy, dan verifikasi pasca-deploy berjalan dari satu tempat dengan jejak audit yang lengkap.

## 87.1 Analisis: Zowe CLI dalam Pipeline Deploy

Zowe CLI memungkinkan pipeline CI/CD yang sama untuk sistem terbuka juga menangani mainframe: mengunggah source, menjalankan compile dan uji, memeriksa hasilnya, dan mempromosikan hasil build. Nilainya bukan hanya otomasi, tetapi juga jejak yang lengkap. Setiap langkah dijalankan dengan cara yang sama dan hasilnya tersimpan.




*Gambar 87.1 — Zowe CLI dalam pipeline deploy*

Langkah paling penting dalam skrip pipeline adalah pemeriksaan hasil. Menjalankan job lalu menganggapnya berhasil karena perintah submit tidak error adalah kesalahan umum. Job bisa berjalan tetapi berakhir dengan return code 8. Contoh di Gambar 87.1 menyimpan hasil job dalam format JSON, lalu menghentikan pipeline bila return code bukan nol.

| Keputusan | Rekomendasi | Alasan |
|---|---|---|
| Identitas pipeline | User teknis khusus, dengan akses hanya ke library pengembangan dan staging | Pipeline yang disusupi tidak dapat menyentuh produksi langsung |
| Kredensial | Disimpan di penyimpanan rahasia pipeline, bukan di skrip; token bila tersedia | Kata sandi di skrip mudah bocor |
| Promosi ke produksi | Langkah terpisah dengan persetujuan dan tiket perubahan | Kontrol perubahan tetap berlaku |
| Bukti | Job log, listing, dan hasil uji disimpan sebagai artefak build | Audit dan diagnosis (Bab 79.3) |

| Keputusan | Rekomendasi | Alasan |
|---|---|---|
| Lingkungan | Pipeline sama untuk UAT dan produksi; perbedaan di parameter | Perilaku yang teruji di UAT sama dengan di produksi |

Mulailah dengan satu aplikasi yang sering berubah dan memiliki uji otomatis yang cukup. Ukur lead time dan tingkat kegagalan perubahannya sebelum dan sesudah pipeline diterapkan (Bab 55.1). Hasil yang terukur dari satu aplikasi lebih meyakinkan daripada rencana besar untuk seluruh portofolio.

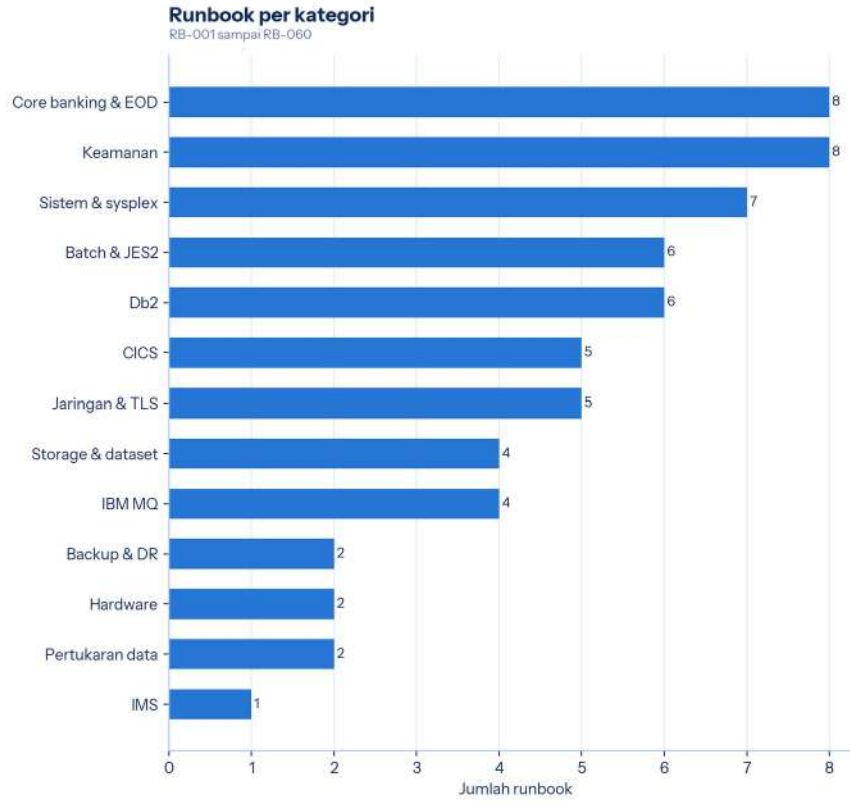
Pipeline tidak menghilangkan kebutuhan pemahaman mainframe. Justru sebaliknya: orang yang menulis pipeline harus memahami JCL, library, dan return code dengan baik, karena pipeline hanya mengotomasi proses yang sudah benar.

# 📕 Bagian XXI — Katalog Runbook Lanjutan

Katalog ini melanjutkan RB-001 sampai RB-022 menjadi 60 runbook (RB-057 sampai RB-060 tentang MQ dan IMS ada di Bab 167), termasuk RB-041 sampai RB-048 tentang core banking dan EOD di Bab 107, RB-049 sampai RB-052 tentang insiden keamanan di Bab 114, RB-053 dan RB-054 tentang pertukaran data di Bab 147, serta RB-055 dan RB-056 tentang TLS di Bab 151. Setiap runbook memakai pola yang sama: gejala, diagnosis, tindakan, pencegahan.

## Bab 88. Indeks Runbook

Sistem dan sysplex, batch, serta Db2 mendominasi katalog. Pola ini sejalan dengan sumber insiden yang paling sering di lingkungan core banking.




*Dihitung dari katalog buku ini*

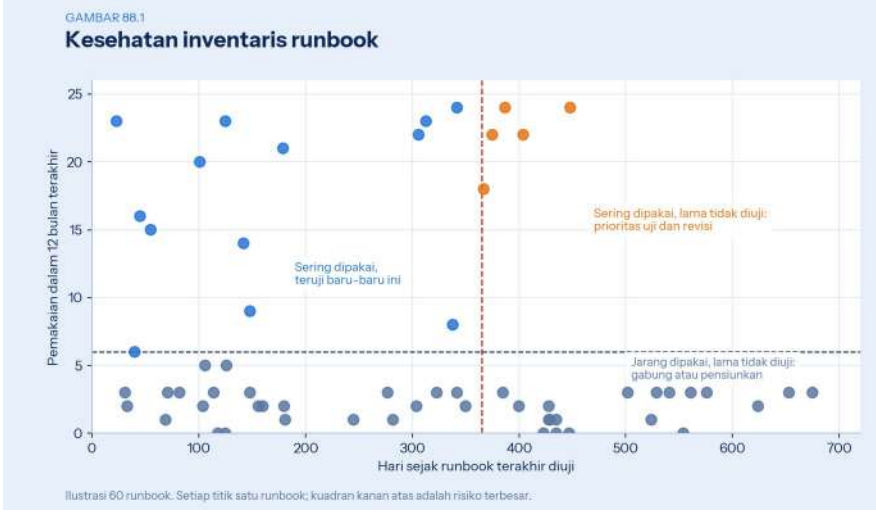
| Kategori | Runbook |
|---|---|
| Storage & dataset | RB-001, RB-023, RB-024, RB-025 |
| Batch & JES2 | RB-002, RB-003, RB-005, RB-026, RB-027, RB-028 |
| Backup & DR | RB-006, RB-008 |
| Sistem & sysplex | RB-007, RB-009, RB-011, RB-029, RB-030, RB-031, RB-032 |
| Keamanan | RB-010, RB-033, RB-034, RB-035, RB-049 s.d. RB-052 (Bab 114) |
| Jaringan | RB-012, RB-036, RB-037, RB-055, RB-056 (Bab 151) |
| CICS | RB-013 s.d. RB-017 |

| Kategori | Runbook |
|---|---|
| Db2 | RB-004, RB-018 s.d. RB-022 |
| IBM MQ | RB-038, RB-057 s.d. RB-059 (Bab 167) |
| Hardware | RB-039, RB-040 |
| Core banking & EOD | RB-041 s.d. RB-048 (Bab 107) |
| Pertukaran data | RB-053, RB-054 (Bab 147) |
| IMS | RB-060 (Bab 167) |

## 88.1 Analisis: Kesehatan Inventaris Runbook

Katalog runbook yang terus bertambah belum tentu semakin berguna. Runbook yang tidak pernah diuji bisa berisi langkah yang sudah tidak berlaku, dan runbook yang tidak pernah dipakai mungkin menutupi masalah yang sudah tidak ada. Dua ukuran sederhana memperlihatkan kesehatan seluruh katalog.

$$\text{Cakupan runbook} = \frac{\text{insiden yang punya runbook relevan}}{\text{total insiden}} \qquad \text{Kesegaran} = \text{hari sejak terakhir diuji}$$




*Gambar 88.1 — Kesehatan inventaris runbook*

Dalam ilustrasi, setiap titik adalah satu runbook. Kuadran kanan atas, yaitu runbook yang sering dipakai tetapi sudah lebih dari setahun tidak diuji,

adalah risiko terbesar. Runbook ini dipercaya petugas jaga, tetapi tidak ada yang tahu apakah langkahnya masih benar setelah perubahan sistem selama setahun terakhir.

| Kuadran | Tindakan |
|---|---|
| Sering dipakai, lama tidak diuji | Uji dan revisi segera, dimulai dari yang paling sering dipakai |
| Sering dipakai, teruji baru-baru ini | Pertahankan jadwal uji |
| Jarang dipakai, teruji | Pertahankan; mungkin untuk kejadian jarang tetapi berdampak besar |
| Jarang dipakai, lama tidak diuji | Tinjau: gabung dengan runbook lain, perbarui, atau pensiunkan |

Cakupan runbook dihitung dari laporan insiden. Insiden tanpa runbook yang relevan adalah kandidat runbook baru, terutama bila jenis insidennya berulang. Sebaliknya, insiden yang terjadi meskipun ada runbook, tetapi runbook-nya tidak membantu, menunjukkan runbook yang perlu diperbaiki.

Catat pemakaian dan tanggal uji terakhir di tabel indeks runbook di bab ini. Dengan dua kolom tambahan itu, grafik seperti Gambar 88.1 bisa dibuat otomatis setiap kuartal dan menjadi agenda tinjauan runbook (Bab 95).

# Bab 89. Runbook Storage dan Dataset

## RB-023: User catalog rusak

- **Gejala:** banyak job gagal menemukan dataset dengan HLQ yang sama; pesan `IDC3009I` dengan return code katalog.
- **Diagnosis:** `F CATALOG,OPEN` untuk memastikan katalog terbuka; jalankan `EXAMINE` dan `DIAGNOSE ICFCATALOG` untuk menentukan kerusakan.
- **Tindakan:** hentikan job yang memakai katalog itu; `F CATALOG,CLOSE(ucat)`; pulihkan dari backup `EXPORT` terakhir,

lalu lakukan forward recovery dari data SMF dengan alat recovery katalog; verifikasi ulang dengan `DIAGNOSE`.

- **Pencegahan:** backup katalog harian, katalog terpisah per aplikasi, dan uji pemulihan katalog setahun sekali.

### RB-024: Recall DFSMShsm macet

- **Gejala:** job menunggu lama saat membuka dataset termigrasi.
- **Diagnosis:** `HSEND QUERY WAITING` dan `HSEND QUERY ACTIVE`; periksa mount tape yang menunggu dengan `D R,L`.
- **Tindakan:** layani mount atau alihkan ke drive lain; batalkan permintaan bermasalah dengan `HSEND CANCEL REQUEST(n)`; recall ulang.
- **Pencegahan:** management class yang tidak memigrasi dataset EOD aktif ke ML2.

### RB-025: Filesystem zFS penuh

- **Gejala:** aplikasi Java atau Liberty gagal menulis log, error `ENOSPC`.
- **Diagnosis:** `D OMVS,F` dan `zfsadm aggrinfo` untuk agregat yang penuh; `du` untuk mencari direktori terbesar.
- **Tindakan:** bersihkan log lama; perbesar dengan `zfsadm grow`; pastikan agregat punya secondary allocation.
- **Pencegahan:** `aggrgrow` aktif, rotasi log, dan alert pemakaian zFS di atas 80%.

## 89.1 Analisis: Mengotomasi Langkah Awal Runbook Storage

Runbook storage termasuk yang paling sering dipakai, karena masalah ruang terjadi hampir setiap pekan di lingkungan yang sibuk. Banyak langkah awalnya selalu sama: memeriksa storage group, mencari volume dengan ruang bebas, dan melihat dataset terbesar. Langkah seperti ini layak diotomasi, sehingga petugas jaga langsung mulai dari analisis, bukan dari pengumpulan data.

| Langkah runbook | Otomasi yang cocok | Keputusan tetap pada |
|---|---|---|

| | | manusia? |
|---|---|---|
| Tampilkan pemakaian storage group terdampak | Ya, dilampirkan pada alert | Tidak |
| Tampilkan 10 dataset terbesar dan paling cepat tumbuh | Ya, dilampirkan pada alert | Tidak |
| Proyeksi kapan penuh dari laju 24 jam terakhir | Ya (Bab 59.1) | Tidak |
| Tambah volume cadangan ke storage group | Bisa, dengan persetujuan (tingkat 3, Bab 26.4) | Ya |
| Hapus atau migrasikan dataset | Tidak | Ya, selalu |

```
ALERT: Storage group SGPROD 91% (naik dari 84% dalam 3
jam)
  Perkiraan penuh: sekitar 3 jam 50 menit pada laju saat
ini
  Dataset tumbuh tercepat 3 jam terakhir:
    BANK.PROD.DEP.EOD.ACCRUAL.G0412V00   +38.000 trk
    BANK.PROD.TRX.SORTWK.T0215           +21.000 trk
  Volume cadangan tersedia: PRD240, PRD241 (persetujuan
dibutuhkan)
  Runbook: RB-001
```

Alert seperti ini memberi petugas jaga semua yang dibutuhkan untuk memutuskan dalam hitungan menit. Tanpa lampiran ini, sepuluh menit pertama habis untuk menjalankan perintah satu per satu, sementara storage group terus terisi.

Otomasi pengumpulan data hampir tanpa risiko, karena hanya membaca. Tindakan yang mengubah sistem, seperti menambah volume, tetap melewati persetujuan. Tindakan yang bisa menghapus data, seperti menghapus atau memigrasikan dataset, tetap sepenuhnya di tangan manusia.

# Bab 90. Runbook Batch dan JES2

## RB-026: Job tidak diambil initiator

- **Gejala:** job berstatus menunggu eksekusi padahal sistem tidak sibuk.
- **Diagnosis:** `$D J` untuk status hold; `$D I` apakah ada initiator untuk class tersebut; `$D JOBCLASS(x)` untuk mode JES atau WLM; jika memakai scheduling environment, `D WLM,SCHENV=nama`.
- **Tindakan:** `$A` job yang ditahan; tambah class ke initiator; aktifkan resource scheduling environment dengan `F WLM,RESOURCE=nama,ON` bila memang seharusnya tersedia.
- **Pencegahan:** dokumentasikan pemetaan class ke initiator dan pantau job yang menunggu lebih dari 15 menit.

## RB-027: Job batch looping dan memakan CPU

- **Gejala:** CPU LPAR naik, satu job mendominasi di panel `DA`, tanpa kemajuan output.
- **Diagnosis:** amati EXCP yang tidak bertambah sementara CPU terus naik; cek jumlah baris SYSOUT.
- **Tindakan:** `C jobname,DUMP` untuk membatalkan sambil menyimpan bukti; analisis dump untuk loop.
- **Pencegahan:** parameter `TIME` realistis per step dan alert CPU per job di otomasi.

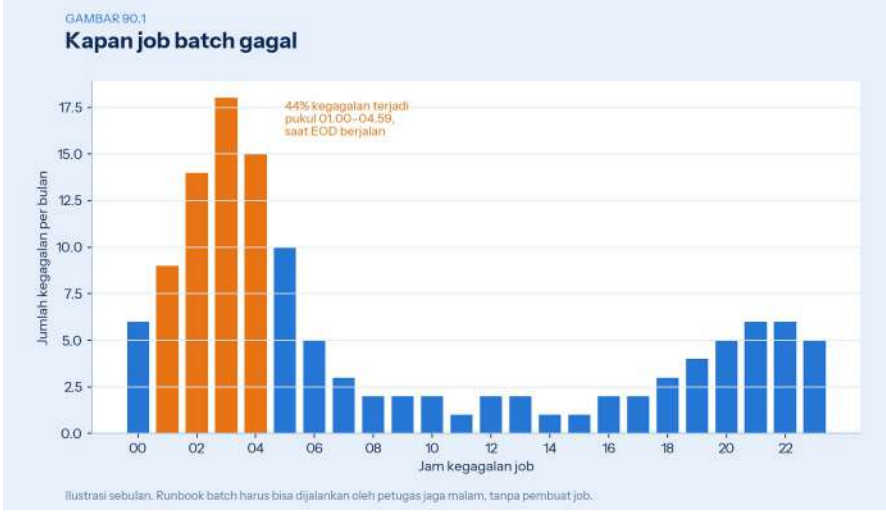
## RB-028: Job berjalan sebelum dependensinya selesai

- **Gejala:** job membaca input kosong atau data kemarin; hasil salah tanpa abend.
- **Diagnosis:** bandingkan waktu mulai job dengan waktu selesai job pendahulu di log scheduler.
- **Tindakan:** hentikan rantai job; pulihkan data yang tercemar; jalankan ulang dalam urutan benar.

- **Pencegahan:** dependensi eksplisit di scheduler dan pemeriksaan tanggal bisnis di header file input.

## 90.1 Analisis: Runbook Batch untuk Petugas Jaga Malam

Sebagian besar kegagalan batch terjadi di malam hari, saat EOD berjalan, dan saat pembuat job serta tim aplikasi sedang tidur. Runbook batch harus ditulis untuk kenyataan itu: petugas jaga malam harus bisa memutuskan dan bertindak tanpa membangunkan orang lain untuk setiap kegagalan.




*Gambar 90.1 — Kapan job batch gagal*

Keputusan utama saat job gagal hampir selalu salah satu dari empat pilihan. Runbook yang baik menentukan pilihan mana yang boleh diambil untuk setiap job, berdasarkan kekritisannya.

| Kelas job | Contoh | Boleh rerun sendiri? | Boleh dilewati? | Eskalasi |
|---|---|---|---|---|
| Jalur kritis EOD | Posting, akrual, GL | Ya, sesuai runbook restart fase | Tidak pernah | Segera bila rerun gagal |
| Pendukung EOD | Laporan internal pagi | Ya | Ya, ditandai untuk dijalankan setelah BOD | Pagi hari |
| Interface keluar | File ke regulator | Ya | Tidak, tetapi | Sebelum |

| Kelas job | Contoh | Boleh rerun sendiri? | Boleh dilewati? | Eskalasi |
|---|---|---|---|---|
| | atau mitra | | bisa ditunda dengan pemberitahuan | tenggat pengiriman |
| Housekeeping | Pembersihan, arsip | Ya | Ya | Hari kerja berikutnya |

Tabel seperti ini, ditempel di setiap runbook job atau disimpan sebagai atribut job di scheduler, menghilangkan keraguan pukul tiga pagi. Petugas jaga tidak perlu menebak apakah laporan internal boleh dilewati. Jawabannya sudah diputuskan di jam kerja, oleh orang yang memahami dampaknya.

Gabungkan ini dengan jalur kritis (Bab 22.1). Job di luar jalur kritis yang gagal memberi waktu sebesar slack-nya untuk diperbaiki tanpa menunda layanan, sehingga keputusan untuk job itu bisa lebih tenang. Job di jalur kritis membutuhkan tindakan segera, dan runbook-nya harus paling jelas dan paling sering diuji.

# Bab 91. Runbook Sistem dan Sysplex

## RB-029: Sistem terasa hang

- **Gejala:** banyak job dan pengguna TSO menunggu, console masih merespons.
- **Diagnosis:** `D GRS,C` dan `D GRS,ANALYZE,BLOCKER`; `D R,L` untuk WTOR yang menahan sumber daya; `D A,L` untuk address space yang tidak lazim.
- **Tindakan:** selesaikan pemegang sumber daya (balas WTOR, batalkan job pemblokir); jika melibatkan address space sistem, eskalasi ke IBM dengan SVC dump.
- **Pencegahan:** otomasi alert untuk contention ENQ yang berlangsung lebih dari beberapa menit.

## RB-030: Structure Coupling Facility bermasalah

- **Gejala:** pesan seri `IXC` dan `IXL` tentang kegagalan koneksi structure; subsistem pemakai melambat.
- **Diagnosis:** `D XCF,STR,STRNAME=nama` dan `D XCF,CF`; tentukan apakah CF atau link yang bermasalah.
- **Tindakan:** `SETXCF START,REBUILD,STRNAME=nama,LOCATION=OTHER` untuk memindahkan structure ke CF lain; untuk structure ber-duplex, sistem umumnya beralih otomatis.
- **Pencegahan:** dua CF, `REBUILDPERCENT` dan duplexing di CFRM policy, serta uji rebuild terencana.

## RB-031: Couple dataset alternatif hilang

- **Gejala:** `D XCF,COUPLE` menunjukkan tidak ada alternate untuk satu tipe CDS.
- **Diagnosis:** periksa volume alternate masih online dan dataset tidak rusak.
- **Tindakan:** tambahkan alternate dengan `SETXCF COUPLE,TYPE=tipe,ACOUPLE=(dsn,volser)`.
- **Pencegahan:** health check XCF aktif dan CDS primer serta alternate di storage controller berbeda.

## RB-032: Peringatan sinkronisasi waktu STP

- **Gejala:** pesan peringatan waktu, `D ETR` menunjukkan kehilangan sumber waktu.
- **Diagnosis:** panel STP di HMC untuk status Preferred, Backup, dan Arbiter server serta coupling link waktu.
- **Tindakan:** pulihkan link atau server STP; ikuti prosedur IBM untuk perubahan peran server.
- **Pencegahan:** konfigurasi STP dengan peran lengkap dan sumber waktu eksternal (NTP atau PTP) yang redundan.

## 91.1 Analisis: Melatih Runbook yang Jarang Dipakai

Runbook sistem dan sysplex adalah kebalikan runbook storage. Runbook ini jarang dipakai, tetapi saat dipakai taruhannya sangat besar: sistem tidak bisa di-IPL, CF hilang, atau sysplex terbelah. Keterampilan yang jarang dipakai cepat memudar, sehingga latihan terjadwal menjadi satu-satunya cara menjaga tim tetap siap.

| Bentuk latihan | Cara | Frekuensi yang disarankan |
|---|---|---|
| Tabletop | Tim membaca runbook bersama untuk skenario tertentu dan menjawab “apa yang kita lakukan selanjutnya?” | Bulanan, satu skenario |
| Latihan di sandbox | Skenario dijalankan nyata di sistem uji, misalnya IPL gagal karena LOADxx rusak | Setiap kuartal |
| Game day | Kegagalan disuntikkan di lingkungan mirip produksi, tim menangani tanpa tahu skenarionya | Setahun sekali atau dua kali |
| Uji DR | Skenario terbesar, dengan dampak bisnis nyata (Bab 121.1) | Sesuai ketentuan, minimal tahunan |

Tabletop terlihat sederhana, tetapi sangat efektif untuk runbook yang jarang dipakai. Setiap kali seseorang bertanya “di mana informasi ini?” atau “siapa yang berwenang memutuskan?”, ditemukan celah di runbook. Semua itu ditemukan di ruang rapat, bukan pukul tiga pagi.

**Skenario yang layak dilatih untuk runbook sistem dan sysplex:**

- IPL gagal di fase NIP; kembali ke konfigurasi sebelumnya (Bab 12.5)
- Satu CF hilang; structure di-rebuild atau pindah ke duplex (Bab 179.1)
- Satu sistem hang; SFM mengisolasi dan beban berpindah (Bab 180.1)
- Couple dataset primer rusak; beralih ke alternate

- Master catalog tidak dapat dibuka; IPL dengan katalog cadangan

Rotasikan peran dalam setiap latihan. Anggota tim yang paling senior sebaiknya tidak selalu memimpin, karena tujuan latihan adalah memastikan orang lain juga mampu. Latihan yang paling berharga adalah ketika anggota tim termuda berhasil memimpin pemulihan hanya dengan runbook.

# Bab 92. Runbook Keamanan

## RB-033: Dugaan penyalahgunaan user berprivilegi

- **Gejala:** SIEM mendeteksi aktivitas tidak lazim oleh user dengan `SPECIAL` atau `OPERATIONS`.
- **Diagnosis:** tarik SMF tipe 80 untuk user itu; bandingkan dengan tiket perubahan yang disetujui.
- **Tindakan:** amankan bukti lebih dulu (salin SMF); `ALTUSER userid REVOKE` jika tidak ada justifikasi; eskalasi ke tim keamanan dan manajemen.
- **Pencegahan:** akses darurat (*firecall*) dengan persetujuan dan log terpisah, bukan atribut permanen.

## RB-034: Database RACF primer dan backup tidak sinkron

- **Gejala:** peringatan saat `RVARY LIST` atau hasil verifikasi berbeda.
- **Diagnosis:** jalankan `IRRUT200` untuk memverifikasi struktur dan membandingkan salinan.
- **Tindakan:** jadwalkan waktu tenang; salin ulang database yang benar dengan `IRRUT200` atau `IRRUT400`; aktifkan kembali dengan `RVARY`.
- **Pencegahan:** verifikasi mingguan dan backup database RACF terjadwal.

### RB-035: Dataset produksi ditemukan terbuka untuk umum

- **Gejala:** audit menemukan profil dengan `UACC(READ)` atau lebih, atau dataset tanpa profil.
- **Diagnosis:** `LISTDSD` untuk profil; analisis unload `IRRDBU00` untuk mencari semua profil dengan UACC di atas NONE.
- **Tindakan:** `ALTDSD 'profil' UACC(NONE)` setelah memastikan akses sah dipindahkan ke group yang tepat.
- **Pencegahan:** `SETROPTS PROTECTALL` dan laporan UACC otomatis setiap bulan.

## 92.1 Analisis: Runbook Keamanan yang Menjaga Bukti

Runbook keamanan berbeda dari runbook operasional dalam satu hal penting: setiap langkah harus menjaga bukti. Runbook operasional biasa sering dimulai dengan “restart” atau “bersihkan”. Langkah seperti itu bisa menghapus jejak yang dibutuhkan untuk memahami insiden keamanan (Bab 112.1).

| Langkah umum di runbook operasional | Versi yang aman untuk insiden keamanan |
|---|---|
| Batalkan job yang mencurigakan | Ambil dump dan salinan output job lebih dulu, lalu batalkan |
| Revoke user yang disalahgunakan | Revoke, tetapi jangan hapus user; unload profilnya sebagai bukti |
| Hapus dataset yang ditanam penyerang | Salin ke lokasi terisolasi dan hitung hash, lalu hapus dari lokasi aslinya |
| Kembalikan library APF ke baseline | Simpan daftar APF saat ini sebagai bukti, lalu kembalikan |
| Purge log lama untuk membebaskan ruang | Hentikan purge; pindahkan log ke penyimpanan lain |

Selain menjaga bukti, runbook keamanan harus jelas tentang siapa yang diberi tahu dan kapan. Insiden keamanan melibatkan pihak di luar tim teknis: tim keamanan informasi, manajemen, dan dalam kondisi tertentu fungsi kepatuhan yang mengurus kewajiban pelaporan kepada regulator. Waktu pemberitahuan bisa diatur ketentuan, sehingga tidak boleh bergantung pada ingatan petugas jaga.

```
RUNBOOK KEAMANAN — langkah pertama yang sama untuk semua
skenario
  1. Catat waktu, siapa yang menemukan, dan gejala awal
  2. Beri tahu tim keamanan informasi lewat saluran
darurat yang disepakati
  3. Amankan bukti sesuai daftar Bab 112.1 sebelum
tindakan pemulihan apa pun
  4. Tindakan pembatasan (revoke, blokir alamat) hanya
dengan persetujuan tim keamanan,
     kecuali kerusakan sedang berlangsung dan harus
dihentikan segera
  5. Setiap tindakan dicatat: siapa, kapan, apa, dan di
sistem mana
```

Latih runbook keamanan bersama tim keamanan informasi dalam latihan tabletop (Bab 91.1). Latihan bersama menyamakan pemahaman tentang siapa memutuskan apa, dan menghindari kebingungan peran saat insiden nyata.

# Bab 93. Runbook Jaringan dan Middleware

## RB-036: Kartu OSA gagal

- **Gejala:** interface turun di `D TCPIP,,NETSTAT,DEVLINKS`; koneksi yang lewat kartu itu terputus.
- **Diagnosis:** `D U,,,devnum,1` dan `D M=CHP(xx)`; periksa pesan hardware di HMC.
- **Tindakan:** pastikan lalu lintas pindah ke OSA lain lewat static VIPA; setelah perbaikan, `V devnum,ONLINE` dan `V TCPIP,,START,nama_interface`.
- **Pencegahan:** minimal dua OSA per LPAR produksi di jalur fisik berbeda.

## RB-037: DVIPA tidak berpindah saat sistem gagal

- **Gejala:** klien tetap tidak terlayani meskipun sistem cadangan aktif.
- **Diagnosis:** `D TCPIP,,NETSTAT,VIPADYN` di sistem cadangan; periksa definisi `VIPABACKUP`.

- **Tindakan:** aktifkan DVIPA secara manual di sistem cadangan sesuai prosedur; perbaiki konfigurasi profile TCP/IP.
- **Pencegahan:** uji takeover DVIPA setiap kali profile TCP/IP berubah.

### RB-038: Antrean MQ terus menumpuk

- **Gejala:** `CURDEPTH` naik terus, alert kedalaman antrean.
- **Diagnosis:** `DISPLAY QSTATUS(nama) TYPE(QUEUE) IPPROCS OPPROCS CURDEPTH` untuk melihat apakah ada konsumen; `DISPLAY CHSTATUS(*)` untuk channel.
- **Tindakan:** pulihkan aplikasi konsumen atau channel; jika antrean mendekati penuh, alihkan produsen atau perbesar `MAXDEPTH` sementara.
- **Pencegahan:** trigger monitoring dan alert `QDEPTHHI` di antrean kritikal.

## 93.1 Analisis: Runbook Lintas Tim untuk Jaringan dan Middleware

Masalah jaringan dan middleware hampir selalu melibatkan lebih dari satu tim: tim mainframe, tim jaringan, tim middleware sistem terbuka, dan kadang mitra luar. Runbook untuk area ini paling sering gagal bukan di langkah teknis, tetapi di serah terima antar-tim. Setiap tim memeriksa sisinya sendiri, menyatakan “di sisi kami normal”, dan masalah terombang-ambing.

| Unsur runbook lintas tim | Isi |
|---|---|
| Pemetaan jalur | Diagram jalur end-to-end dengan pemilik setiap segmen (Bab 35.2) |
| Titik ukur bersama | Di mana masing-masing tim mengukur, dengan ID korelasi yang sama |
| Bukti “normal” | Apa yang harus ditunjukkan sebuah tim untuk menyatakan segmennya normal |
| Kontak dan eskalasi | Nama dan saluran untuk setiap tim, termasuk mitra luar |
| Pemimpin insiden | Satu orang yang memegang gambaran utuh |

| Unsur runbook lintas tim | Isi |
|---|---|
| | dan memutuskan langkah berikutnya |

Baris “bukti normal” adalah kuncinya. Menyatakan segmen sendiri normal harus disertai data: statistik koneksi, waktu respons di titik ukur, atau hasil uji dari sisi itu. Tanpa bukti, pernyataan “normal” hanya memindahkan masalah ke tim berikutnya.

Diagnosis berlapis dari bawah ke atas (Bab 39.1) juga berlaku lintas tim. Mulailah dari konektivitas dasar, lanjut ke TLS, lalu ke aplikasi. Setiap lapisan dikonfirmasi oleh tim pemiliknya dengan bukti, sebelum pindah ke lapisan berikutnya.

Setelah insiden lintas tim selesai, tinjau bersama di mana waktu terbuang. Sering kali perbaikan terbesar bukan di teknologi, melainkan di kesepakatan tentang titik ukur bersama. Setelah semua tim mengukur di titik yang sama dengan ID korelasi yang sama, pertanyaan “di mana lambatnya?” bisa dijawab dalam hitungan menit.

# Bab 94. Runbook Hardware

## RB-039: CHPID atau jalur I/O offline

- **Gejala:** pesan seri `IOS`, jalur ke device berkurang di `D M=DEV(xxxx)`.
- **Diagnosis:** `D M=CHP(xx)`; periksa *Problem Analysis* dan *Hardware Messages* di HMC.
- **Tindakan:** pastikan device masih punya jalur lain; buka layanan IBM (umumnya sudah dibuat otomatis lewat Call Home); setelah perbaikan, `V PATH(dev,chp),ONLINE`.
- **Pencegahan:** setiap device kritikal punya minimal dua jalur di adapter dan switch berbeda.

## RB-040: Kegagalan komponen prosesor atau memori

- **Gejala:** pesan hardware di HMC; sistem tetap berjalan karena spare PU atau memori redundan mengambil alih.

- **Diagnosis:** konfirmasi di HMC bahwa sparing berhasil dan Call Home terkirim.
- **Tindakan:** koordinasikan jadwal perbaikan dengan IBM; periksa apakah kapasitas cadangan masih tersedia.
- **Pencegahan:** kontrak layanan dengan waktu respons sesuai tingkat kritikal dan pemantauan pesan hardware HMC.

## 94.1 Analisis: Menangani Kegagalan Hardware Bersama IBM

Kegagalan hardware IBM Z jarang terjadi, dan ketika terjadi, sebagian besar ditangani otomatis. Prosesor yang bermasalah digantikan prosesor cadangan (sparing). Banyak komponen dapat diperbaiki tanpa mematikan mesin. Mesin juga dapat melaporkan masalahnya sendiri ke IBM lewat fasilitas dukungan jarak jauh. Peran tim bank adalah memahami dampak kegagalan terhadap kapasitas dan redundansi, lalu berkoordinasi dengan teknisi IBM.

| Pertanyaan | Mengapa penting |
|---|---|
| Apakah ada kapasitas yang hilang? | Prosesor yang gagal tanpa cadangan tersisa mengurangi kapasitas LPAR |
| Apakah ada redundansi yang hilang? | Jalur I/O, adapter jaringan, atau komponen daya yang tinggal satu |
| Apakah perbaikan bisa dilakukan tanpa gangguan? | Menentukan apakah perlu jendela pemeliharaan |
| Apakah ada dampak ke situs DR? | Mesin DR yang bermasalah melemahkan perlindungan meskipun produksi normal |
| Kapan perbaikan dijadwalkan? | Selama menunggu, redundansi berkurang |

Kehilangan redundansi adalah hal yang paling perlu diperhatikan. Satu adapter OSA yang gagal mungkin tidak berdampak apa pun karena ada adapter lain. Namun sampai adapter itu diganti, satu kegagalan berikutnya akan memutus jaringan. Periode ini harus diperlakukan sebagai risiko yang dipantau ketat, dan perubahan berisiko di area yang sama sebaiknya ditunda.

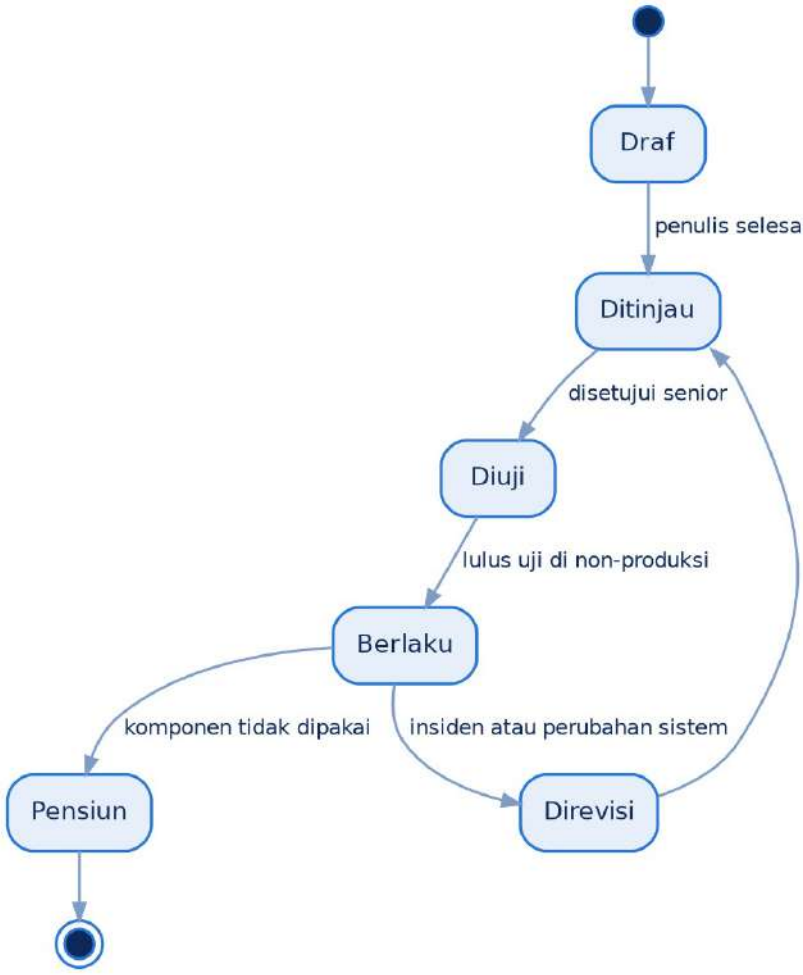
```
ALGORITMA TanganiPeringatanHardware(peringatan)
  catat peringatan dari HMC: komponen, lokasi, tingkat
```

```
keparahan
  pastikan laporan sudah terkirim ke IBM; bila belum, buka
case (Bab 176.1)
  nilai dampak: kapasitas hilang? redundansi hilang? situs
mana?
  jika redundansi hilang di komponen kritikal:
      naikkan pemantauan area itu; tunda perubahan
berisiko di area yang sama
  sepakati jadwal perbaikan dengan teknisi IBM, termasuk
apakah perlu jendela
  setelah perbaikan: verifikasi redundansi pulih (jalur,
CHPID, adapter aktif)
```

Setiap peringatan hardware sebaiknya dicatat walaupun tidak berdampak. Pola peringatan yang berulang pada komponen yang sama bisa menjadi alasan penggantian proaktif sebelum kegagalan yang lebih serius.

## Bab 95. Memelihara Katalog Runbook

Runbook yang tidak pernah diuji adalah dugaan, bukan prosedur. Kelola katalog seperti kode: berversi, ditinjau, dan diuji berkala.




Template yang dipakai seluruh katalog:

```
### RB-XXX: <judul singkat berupa gejala>
- Pemilik: <tim> · Terakhir diuji: <tanggal> · Versi: <n>
- Gejala: <apa yang dilihat operator atau pengguna>
- Diagnosis: <perintah dan apa yang dicari>
- Tindakan: <langkah berurutan, termasuk titik eskalasi>
- Pencegahan: <perubahan agar tidak terulang>
- Referensi: <message ID, bab terkait, tiket insiden>
```

- Tulis judul runbook sebagai gejala yang dilihat operator, bukan nama penyebabnya.

- Setiap insiden besar harus menghasilkan runbook baru atau revisi runbook lama.
- Uji minimal 10 runbook per kuartal di lingkungan non-produksi, bergilir.
- Tautkan setiap message ID di Bab 50 ke runbook yang relevan.

## 95.1 Analisis: Uji Pembaca Baru dan Siklus Tinjauan

Cara paling jujur menilai sebuah runbook adalah meminta orang yang belum pernah memakainya menjalankannya. Penulis runbook hampir tidak bisa menilai tulisannya sendiri, karena ia mengisi celah dengan pengetahuan yang ada di kepalanya tanpa sadar.

**Uji pembaca baru:**

1. Pilih anggota tim yang belum pernah menangani insiden jenis itu.
2. Berikan runbook dan skenario di lingkungan uji, tanpa penjelasan tambahan.
3. Amati tanpa membantu. Catat setiap kali ia berhenti, bertanya, atau menebak.
4. Setiap catatan itu adalah perbaikan runbook: langkah yang hilang, istilah yang tidak jelas, atau informasi yang tidak tersedia.

Setelah beberapa putaran, runbook yang tadinya hanya bisa dijalankan penulisnya menjadi bisa dijalankan siapa pun di rotasi jaga. Nilai katalog runbook sebenarnya terletak di situ.

| Aktivitas tinjauan | Frekuensi | Pemicu tambahan |
|---|---|---|
| Uji pembaca baru untuk runbook kritikal | Setahun sekali per runbook | Anggota baru bergabung ke rotasi jaga |
| Tinjauan isi terhadap kondisi sistem | Setiap kuartal untuk runbook yang sering dipakai (Bab 88.1) | Upgrade z/OS, subsistem, atau perubahan topologi |
| Pembaruan setelah insiden | Setelah setiap insiden yang memakai runbook | Runbook terbukti kurang atau salah |
| Pensiun runbook | Tahunan | Komponen yang dicakup sudah tidak ada |

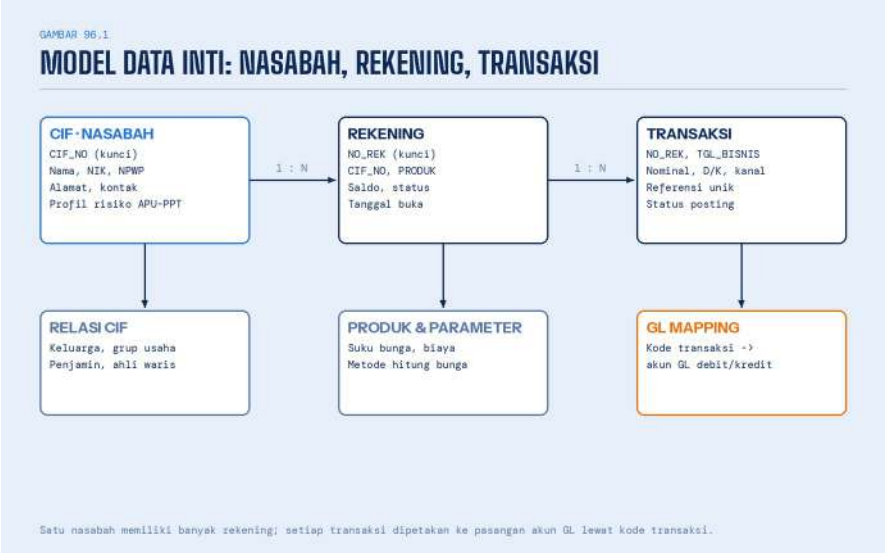
Setiap runbook sebaiknya mencantumkan tanggal terakhir diuji dan nama pengujinya di bagian atas. Informasi sederhana ini memberi tahu petugas jaga seberapa bisa ia mempercayai runbook tersebut. Runbook yang terakhir diuji tiga tahun lalu tetap berguna sebagai panduan, tetapi setiap langkahnya perlu diperiksa ulang sebelum dijalankan.

# 🏛️ Bagian XXII — Modul Core Banking

System engineer tidak mengelola logika bisnis, tetapi harus memahaminya cukup dalam untuk tahu kapan angka “terlihat salah”. Bagian ini membahas modul inti dari sudut pandang operasi: data apa yang disimpan, proses batch apa yang menyentuhnya, dan titik rawan apa yang sering memicu insiden.

## Bab 96. Customer Information File

CIF adalah sumber tunggal identitas nasabah. Semua rekening, pinjaman, dan layanan terhubung ke satu nomor CIF, sehingga kualitas CIF menentukan kualitas hampir semua laporan bank.




*Gambar 96.1 — Model data inti: nasabah, rekening, transaksi*

| Area CIF | Isi | Catatan operasional |
|---|---|---|
| Identitas | Nama, NIK, NPWP, tanggal lahir | Verifikasi NIK ke data kependudukan saat pembukaan |
| Kontak dan alamat | Alamat, telepon, email | Dipakai notifikasi dan proxy BI-FAST |

| Area CIF | Isi | Catatan operasional |
|---|---|---|
| Profil APU-PPT | Tingkat risiko, pekerjaan, sumber dana | Dasar pemantauan transaksi mencurigakan |
| Relasi | Keluarga, grup usaha, penjamin | Dasar perhitungan eksposur grup |
| Status | Aktif, meninggal, diblokir | Mempengaruhi seluruh rekening terkait |

## 96.1 Algoritma deteksi CIF ganda

CIF ganda membuat eksposur kredit dan pelaporan SLIK tidak akurat. Deteksi rutin sebaiknya dijalankan sebagai batch mingguan.

```
ALGORITMA DeteksiCIFGanda()
  kandidat <- kosong
  untuk setiap kelompok CIF dengan NIK sama dan NIK valid:
      tambahkan kelompok ke kandidat dengan skor 100
  untuk setiap kelompok dengan nama_normal dan tgl_lahir
sama:
      skor <- 70
      jika ibu_kandung sama maka skor <- skor + 20
      jika alamat_normal mirip maka skor <- skor + 10
      jika skor >= 80 maka tambahkan ke kandidat
  kirim kandidat ke unit data nasabah untuk verifikasi
manual
  -- penggabungan CIF TIDAK PERNAH otomatis
```

Penggabungan CIF memindahkan rekening, riwayat, dan relasi. Proses ini harus berjalan lewat fungsi aplikasi yang teraudit, bukan update langsung ke tabel.

## 96.2 Perlindungan data pribadi

Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi berlaku penuh sejak Oktober 2024. Bagi system engineer, konsekuensinya konkret: data CIF produksi tidak boleh disalin mentah ke lingkungan development atau UAT.

- Gunakan masking atau data sintetis saat menyalin data ke lingkungan non-produksi.

- Batasi akses dataset dan tabel CIF lewat RACF ke group yang benar-benar perlu.
- Enkripsi dataset CIF dengan pervasive encryption (Bab 31).
- Catat akses baca massal ke tabel CIF lewat audit Db2 dan SMF.

## 96.3 Analisis: Mendeteksi dan Menangani Nasabah Ganda

Satu orang yang tercatat sebagai dua nasabah berbeda di CIF menimbulkan banyak masalah. Batas transaksi dan penilaian risiko terpecah, laporan kredit ke SLIK tidak utuh, dan pemantauan APU-PPT bisa kehilangan gambaran lengkap. Duplikat biasanya muncul dari salah ketik saat pembukaan rekening, migrasi dari sistem lama, atau penggabungan bank.


*Gambar 96.2 — Mendeteksi nasabah ganda di CIF*

Pendeteksian dilakukan dengan skor kemiripan berbobot:

$$S = \sum_i w_i \times s_i \qquad \sum_i w_i = 1, \quad 0 \le s_i \le 1$$

w adalah bobot setiap atribut, dan s tingkat kemiripan atribut antara dua catatan. NIK yang sama memberi kemiripan 1. Nama dibandingkan dengan algoritma kemiripan string yang toleran terhadap salah ketik, sehingga “Siti Rahma” dan “Siti Rachma” tetap dinilai sangat mirip.

Membandingkan setiap nasabah dengan semua nasabah lain tidak mungkin dilakukan: sepuluh juta nasabah menghasilkan sekitar 50 triliun pasangan. Karena itu dipakai teknik *blocking*. Hanya catatan yang sama pada kunci sederhana tertentu, misalnya tanggal lahir atau empat digit awal NIK, yang dibandingkan. Jumlah pasangan turun menjadi jutaan, dan pekerjaan ini bisa berjalan sebagai batch mingguan.

| Hasil | Tindakan | Catatan |
|---|---|---|
| Skor tinggi | Masuk antrean peninjauan cepat | Tetap disetujui manusia sebelum digabung |
| Skor menengah | Peninjauan dengan dokumen pendukung | Sering berupa anggota keluarga dengan data mirip |
| Skor rendah | Diabaikan | Disimpan untuk menyetel bobot |

Penggabungan CIF tidak boleh otomatis penuh. Menggabungkan dua orang yang berbeda jauh lebih merusak daripada membiarkan duplikat, karena rekening, hak akses, dan riwayat kredit ikut tercampur. Proses penggabungan harus bisa dibatalkan dan tercatat lengkap untuk audit.

Pencegahan lebih murah daripada pembersihan. Pemeriksaan kemiripan saat pembukaan rekening, dengan peringatan “nasabah serupa sudah ada”, menghentikan sebagian besar duplikat sebelum tercipta.

## Bab 97. Tabungan dan Giro

Rekening tabungan dan giro (sering disebut DDA, *Demand Deposit Account*) adalah modul dengan volume transaksi terbesar. Hampir setiap transaksi kanal digital menyentuh modul ini.

| Elemen | Fungsi |
|---|---|
| Saldo buku dan saldo efektif | Saldo efektif = saldo buku dikurangi dana |

| Elemen | Fungsi |
|---|---|
| | yang diblokir atau ditahan |
| Hold / blokir | Dana ditahan untuk jaminan, sengketa, atau perintah hukum |
| Overdraft (giro) | Fasilitas cerukan dengan limit dan bunga sendiri |
| Status dormant | Rekening tanpa transaksi dalam periode tertentu |
| Biaya administrasi | Dibebankan saat EOM |

## 97.1 Perhitungan bunga harian

Bunga tabungan umumnya dihitung harian dari saldo akhir hari dan dikreditkan bulanan. Untuk basis 365 hari:

$$B_{\text{harian}} = \frac{S_{\text{akhir hari}} \times r_{\text{tahunan}}}{365}$$

$$B_{\text{bulan}} = \sum_{d=1}^{D} B_{\text{harian},\,d} \qquad B_{\text{neto}} = B_{\text{bulan}} \times (1 - t_{\text{PPh}})$$

Bunga simpanan untuk saldo di atas Rp7,5 juta dikenakan PPh final 20%. Beberapa produk memakai metode saldo terendah atau saldo rata-rata, dan suku berjenjang (*tiering*) sesuai besar saldo. Parameter ini ada di tabel produk, bukan di program.

## 97.2 Tugas batch modul tabungan dan giro

| Kapan | Proses |
|---|---|
| Setiap EOD | Akrual bunga harian, pembebasan hold yang kedaluwarsa, pembaruan saldo rata-rata |
| Setiap EOM | Kredit bunga, potong pajak, biaya administrasi, cetak atau kirim e-statement |
| Berkala | Penandaan rekening dormant, penutupan rekening bersaldo nol |

## 97.3 Analisis: Pembulatan Bunga Harian

Bunga tabungan dihitung harian dari saldo akhir hari. Hasil perhitungan hampir selalu berupa pecahan rupiah, dan cara pembulatannya, yang terlihat sepele, bisa menghasilkan selisih besar di tingkat bank.

$$b_{\text{harian}} = \frac{S \times r}{N_{\text{hari setahun}}} \qquad E_{\text{bulanan}} \approx N_{\text{rekening}} \times D \times \bar{e}$$

ē adalah rata-rata kesalahan pembulatan per rekening per hari, dan D jumlah hari dalam bulan.

| Metode | Rata-rata kesalahan per perhitungan | Akibat |
|---|---|---|
| Pemotongan (truncation) harian | Sekitar setengah satuan terkecil, selalu merugikan nasabah | Bias sistematis |
| Pembulatan ke terdekat harian | Mendekati nol rata-rata | Kesalahan saling menghapus, tetapi tetap tercatat |
| Akumulasi tanpa pembulatan, dibulatkan saat posting bulanan | Paling kecil | Membutuhkan field presisi tinggi untuk akrual |

Contoh besarnya efek: bank dengan 10 juta rekening tabungan yang memotong bunga harian ke rupiah terdekat di bawahnya menanggung bias sekitar 0,5 rupiah per rekening per hari. Dalam 30 hari, total selisihnya mendekati Rp150 juta per bulan, semuanya ke arah yang merugikan nasabah. Angka ini cukup besar untuk menjadi temuan audit dan masalah perlindungan konsumen.

Dua hal lain perlu diputuskan secara eksplisit dan terdokumentasi. Pertama, basis hari: 365 hari tetap, atau 366 hari pada tahun kabisat (Bab 106.1). Kedua, saldo yang dipakai: saldo akhir hari, saldo rata-rata, atau saldo terendah, sesuai ketentuan produk. Keduanya harus sama persis antara sistem core, laporan nasabah, dan laporan regulator.

Uji perhitungan bunga dengan kasus tepi: saldo sangat kecil, saldo sangat besar, rekening dibuka atau ditutup di tengah bulan, perubahan suku

bunga di tengah bulan, dan 29 Februari. Kesalahan di kasus tepi inilah yang biasanya lolos dari uji normal.

# Bab 98. Deposito Berjangka

Deposito menyimpan dana untuk jangka tertentu dengan bunga tetap. Tantangan operasionalnya adalah jatuh tempo: ribuan deposito dapat jatuh tempo pada tanggal yang sama dan harus diproses dalam satu malam.

| Instruksi jatuh tempo | Perilaku |
|---|---|
| ARO pokok | Pokok diperpanjang, bunga dibayar ke rekening |
| ARO pokok + bunga | Pokok dan bunga diperpanjang bersama |
| Non-ARO | Pokok dan bunga dicairkan ke rekening tujuan |

$$B_{\text{deposito}} = \frac{P \times r \times h}{365}$$

P adalah nominal, r suku bunga tahunan, dan h jumlah hari aktual jangka waktu. Jika jatuh tempo pada hari libur, pemrosesan bergeser mengikuti kalender bisnis, dan bunga untuk hari tambahan mengikuti kebijakan produk.

Penjaminan LPS berlaku hingga Rp2 miliar per nasabah per bank, dengan syarat suku bunga tidak melebihi tingkat bunga penjaminan LPS. Modul deposito harus menandai rekening yang bunganya di atas batas itu.

## 98.1 Analisis: Lonjakan Jatuh Tempo Deposito

Deposito dengan tenor tetap menciptakan beban yang bisa diramalkan dengan tepat: setiap deposito yang ditempatkan hari ini akan jatuh tempo pada tanggal yang sudah diketahui. Program promosi deposito yang sangat sukses hari ini adalah lonjakan pemrosesan EOD 1, 3, 6, atau 12 bulan kemudian.


*Gambar 98.1 — Konsentrasi jatuh tempo deposito*

Fase jatuh tempo di EOD memproses setiap deposito yang jatuh tempo: menghitung bunga terakhir, memotong pajak, lalu memperpanjang otomatis (ARO) atau mencairkan ke rekening tujuan. Durasinya sebanding dengan jumlah deposito:

$$T_{\text{fase jatuh tempo}} \approx N_{\text{jatuh tempo}} \times t_{\text{per deposito}}$$

Pada hari biasa dengan sekitar 20 ribu deposito dan 2 ms per deposito, fase ini selesai dalam sekitar 40 detik. Pada tanggal jatuh tempo program promosi dengan sekitar 410 ribu deposito, durasinya menjadi sekitar 14 menit, belum termasuk notifikasi ke nasabah dan pembukuan ke GL. Jika fase ini berada di jalur kritis (Bab 22.1), EOD pada tanggal itu ikut memanjang.

Karena tanggalnya sudah diketahui, lonjakan ini bisa direncanakan:

| Langkah | Waktu |
|---|---|
| Buat laporan proyeksi jatuh tempo 12 bulan ke depan dari tabel deposito | Setiap bulan |
| Tandai tanggal dengan jatuh tempo di atas ambang, misalnya lima kali rata-rata | Otomatis di laporan |
| Koordinasi dengan tim bisnis: apakah | Satu bulan sebelumnya |

| Langkah | Waktu |
|---|---|
| promosi perpanjangan akan menambah beban kanal | |
| Uji fase jatuh tempo dengan volume setara di UAT | Dua pekan sebelumnya |
| Pastikan kapasitas notifikasi (SMS, push, email) cukup | Satu pekan sebelumnya |

Kolaborasi dengan tim produk sejak awal membantu. Tim produk dapat menyebar tanggal penempatan promosi, atau memberi tenor yang berbeda-beda, sehingga jatuh temponya tidak menumpuk di satu hari.

# Bab 99. Kredit

Modul kredit mengelola pencairan, jadwal angsuran, pembayaran, tunggakan, dan kualitas aset. Kesalahan di modul ini langsung mempengaruhi pencadangan dan laporan ke OJK.

## 99.1 Metode perhitungan angsuran

| Metode | Ciri | Umum dipakai untuk |
|---|---|---|
| Flat | Bunga dihitung dari pokok awal sepanjang tenor | Kredit konsumer jangka pendek |
| Efektif / sliding | Bunga dari sisa pokok; angsuran menurun | Kredit modal kerja |
| Anuitas | Angsuran tetap; porsi bunga menurun, porsi pokok naik | KPR, kredit investasi |

Rumus angsuran anuitas untuk pokok P, bunga per periode i, dan jumlah periode n:

$$A = P \times \frac{i}{1 - (1 + i)^{-n}}$$

Contoh: pinjaman Rp120 juta, bunga 12% per tahun (1% per bulan), tenor 12 bulan menghasilkan angsuran tetap Rp10.661.855. Porsi bunga turun

dari Rp1,2 juta di bulan pertama menjadi sekitar Rp106 ribu di bulan terakhir.

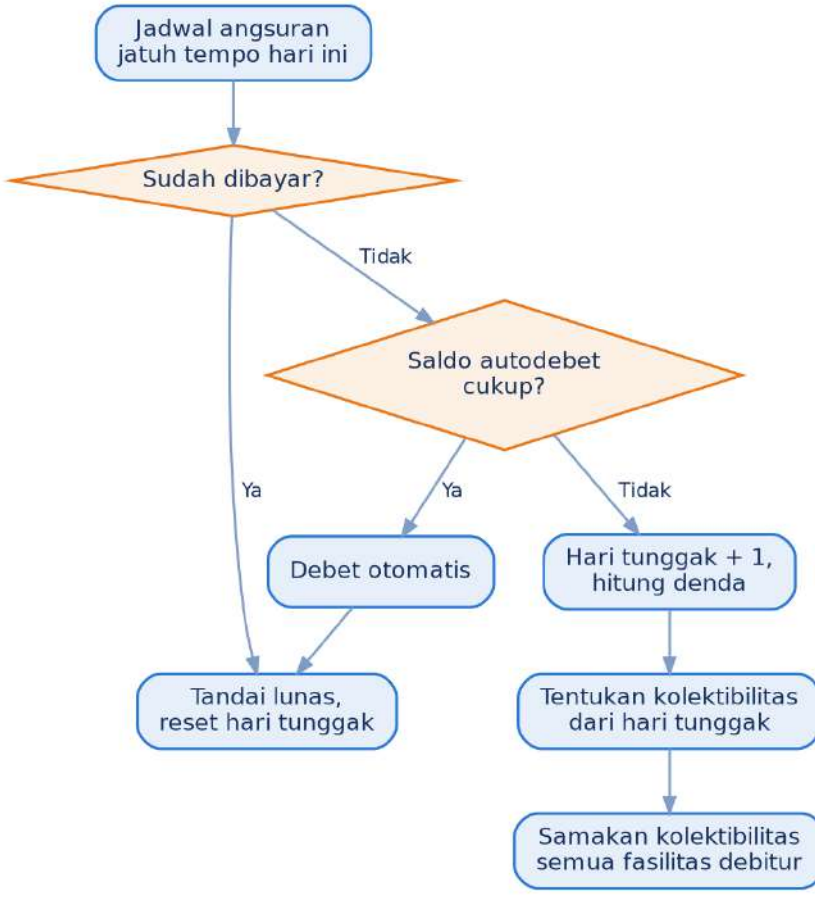

*Dihitung dengan rumus anuitas · contoh*

## 99.2 Tunggakan dan kolektibilitas

Kualitas kredit dinilai berdasarkan ketepatan pembayaran, antara lain mengacu pada POJK Nomor 40/POJK.03/2019 tentang Penilaian Kualitas Aset Bank Umum. Pembagian berdasarkan hari tunggakan yang lazim dipakai:

| Kolektibilitas | Nama | Tunggakan pokok/bunga |
|---|---|---|
| 1 | Lancar | Tidak ada tunggakan |
| 2 | Dalam Perhatian Khusus | 1 sampai 90 hari |
| 3 | Kurang Lancar | 91 sampai 120 hari |
| 4 | Diragukan | 121 sampai 180 hari |
| 5 | Macet | Lebih dari 180 hari |

Penilaian kualitas juga mempertimbangkan prospek usaha dan kinerja debitur, serta prinsip satu debitur satu kolektibilitas lintas fasilitas. Cadangan kerugian penurunan nilai (CKPN) dihitung mengikuti standar akuntansi yang berlaku (PSAK 71, yang sejak 2024 bernomor PSAK 109).




Prinsip satu debitur satu kolektibilitas berarti proses aging harus berjalan setelah semua fasilitas debitur diproses. Urutan fase EOD di modul kredit tidak boleh diubah tanpa analisis dampak.

## 99.3 Analisis: Matriks Migrasi Kolektibilitas

Fase aging kredit di EOD menentukan kolektibilitas setiap fasilitas setiap hari. Data yang dihasilkan fase ini juga menjadi bahan analisis risiko yang sangat berharga: seberapa banyak kredit yang berpindah dari satu tingkat kolektibilitas ke tingkat lain dari bulan ke bulan. Hasilnya disebut matriks migrasi atau *roll rate*.

| Dari \ Ke | Kol 1 | Kol 2 | Kol 3 | Kol 4 | Kol 5 |
|---|---|---|---|---|---|
| Kol 1 | 97,0% | 2,6% | 0,3% | 0,1% | 0,0% |
| Kol 2 | 35,0% | 45,0% | 15,0% | 3,0% | 2,0% |
| Kol 3 | 10,0% | 10,0% | 40,0% | 30,0% | 10,0% |
| Kol 4 | 5,0% | 3,0% | 7,0% | 45,0% | 40,0% |
| Kol 5 | 2,0% | 0,0% | 0,0% | 3,0% | 95,0% |

Angka dalam tabel adalah ilustrasi. Setiap baris menjumlah 100%. Baris Kol 2 misalnya berarti: dari fasilitas yang berstatus Dalam Perhatian Khusus bulan lalu, 35% kembali lancar, 45% tetap, dan 20% memburuk.

Distribusi portofolio bulan depan dapat diproyeksikan dengan perkalian vektor dan matriks:

$$\mathbf{p}_{t+1} = \mathbf{p}_t \times M$$

p adalah proporsi portofolio di setiap tingkat kolektibilitas, dan M matriks migrasi. Proyeksi ini dipakai tim risiko untuk memperkirakan kebutuhan pencadangan (CKPN) dan untuk mendeteksi perubahan perilaku pembayaran lebih awal.

Peran system engineer di sini adalah memastikan data mentahnya benar dan tersedia. Simpan snapshot kolektibilitas per fasilitas setiap akhir bulan dalam tabel riwayat yang dipartisi per bulan, dengan kunci fasilitas yang stabil walaupun terjadi restrukturisasi. Matriks migrasi yang dihitung dari data dengan kunci yang berubah-ubah akan menunjukkan pergerakan yang sebenarnya tidak terjadi.

Perubahan mendadak pada satu baris matriks, misalnya lonjakan Kol 1 ke Kol 2 dari 2,6% menjadi 6%, bisa berarti dua hal: memburuknya kondisi debitur, atau kesalahan pada fase aging. Karena itu setiap perubahan besar pada matriks sebaiknya diperiksa dari kedua sisi, bisnis dan teknis, sebelum disimpulkan.

# Bab 100. General Ledger

General Ledger adalah muara seluruh transaksi. Setiap kode transaksi di modul lain dipetakan ke pasangan akun debit dan kredit, sehingga GL yang tidak seimbang hampir selalu menunjuk ke mapping yang salah atau posting yang tidak lengkap.

| Konsep | Arti |
|---|---|
| Chart of accounts | Struktur akun: aset, liabilitas, ekuitas, pendapatan, beban |
| Mapping kode transaksi | Aturan otomatis dari transaksi ke jurnal GL |
| Rekening antar kantor (RAK) | Menjembatani transaksi yang melibatkan dua cabang |
| Akun suspense | Penampung sementara transaksi yang belum bisa diposting ke akun final |
| Revaluasi valas | Penilaian ulang posisi valuta asing dengan kurs referensi, misalnya JISDOR |

## 100.1 Algoritma balancing GL harian

```
ALGORITMA BalancingGL(tanggal)
  untuk setiap cabang c:
      selisih_c <- total_debit(c, tanggal) -
total_kredit(c, tanggal)
      jika selisih_c <> 0 maka catat_selisih(c, selisih_c)
  total_RAK <- jumlah saldo RAK semua cabang
  jika total_RAK <> 0 maka catat_selisih('RAK', total_RAK)
  untuk setiap akun suspense s:
      jika saldo(s) <> 0 dan umur(s) > batas maka
catat_peringatan(s)
  jika tidak ada selisih maka tandai GL_SEIMBANG
  selain itu: tahan ganti tanggal, jalankan RB-005
```

RAK yang tidak nol secara total menandakan transaksi antar cabang yang hanya terposting di satu sisi. Akun suspense yang terus menumpuk adalah gejala awal masalah interface, jauh sebelum GL benar-benar tidak seimbang.

## 100.2 Penutupan periode

| Periode | Proses tambahan |
|---|---|
| EOM | Kredit bunga simpanan, pembebanan biaya, amortisasi, revaluasi valas, laporan bulanan |
| Akhir kuartal | Laporan publikasi dan laporan triwulanan regulator |
| EOY | Penutupan akun laba rugi ke laba ditahan, laporan tahunan, arsip data tahunan |

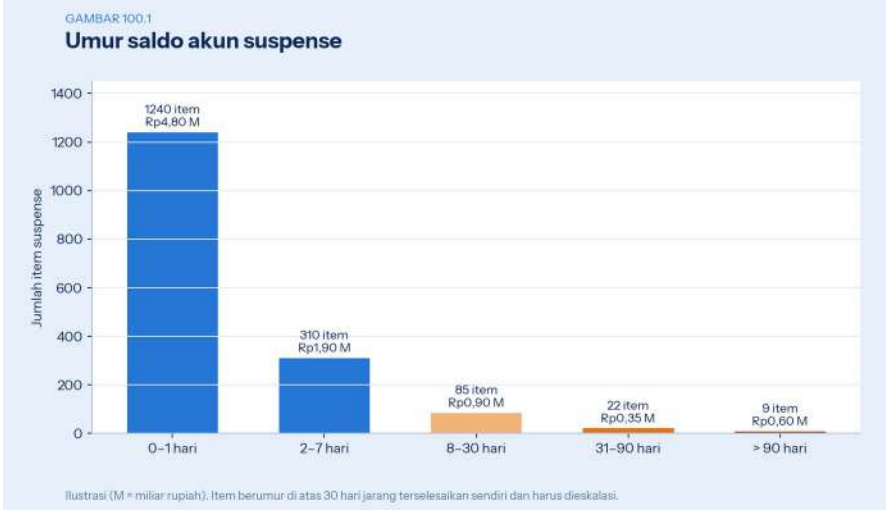
## 100.3 Analisis: Keseimbangan GL dan Umur Suspense

General Ledger yang sehat memenuhi dua persamaan sederhana setiap akhir hari. Yang pertama berlaku untuk seluruh bank, yang kedua untuk rekening antar kantor:

$$\sum \text{debit} = \sum \text{kredit} \qquad \sum_{\text{semua cabang}} \text{saldo RAK} = 0$$

Jika persamaan pertama tidak terpenuhi, ada jurnal yang hanya tercatat separuh. Jika persamaan kedua tidak terpenuhi, ada transaksi antar cabang yang sampai di satu sisi tetapi belum sampai di sisi lain. Keduanya harus diperiksa otomatis di akhir fase GL, sebelum ganti tanggal (Bab 104).

**Umur suspense.** Akun suspense menampung transaksi yang belum bisa dibukukan ke tempat akhirnya, misalnya transfer masuk dengan nomor rekening tidak dikenal. Suspense yang sehat bergerak cepat: item masuk dan keluar dalam satu atau dua hari. Suspense yang menumpuk adalah tanda masalah proses, dan juga risiko kerugian.




*Gambar 100.1 — Umur saldo akun suspense*

| Umur item | Tindakan | Pemilik |
|---|---|---|
| 0–1 hari | Penyelesaian rutin oleh tim rekonsiliasi | Operasional |
| 2–7 hari | Investigasi: hubungi mitra atau cabang | Operasional |
| 8–30 hari | Eskalasi ke kepala unit; laporan mingguan | Kepala unit |
| Lebih dari 30 hari | Eskalasi ke manajemen dan fungsi risiko; evaluasi pencadangan | Manajemen |
| Lebih dari 90 hari | Keputusan akhir: pengembalian, penghapusan, atau prosedur hukum | Komite terkait |

Dalam ilustrasi di atas, sebagian besar item berumur kurang dari seminggu, dan itu wajar. Yang perlu diperhatikan adalah item berumur lebih dari 90 hari. Jumlahnya hanya sembilan, tetapi nilainya sekitar Rp0,6 miliar. Item seperti ini jarang terselesaikan sendiri.

Dari sisi sistem, tugas system engineer adalah memastikan laporan umur suspense tersedia setiap pagi, akurat, dan tidak bergantung pada kerja

manual. Laporan itu bisa dibangun dari tabel transaksi suspense dengan satu query terjadwal, lalu dikirim ke pemilik proses masing-masing.

# 💸 Bagian XXIII — Pembayaran dan Interface

Core banking di mainframe tidak berdiri sendiri. Sebagian besar transaksinya datang dari atau menuju sistem pembayaran nasional, jaringan switching, dan kanal digital, sehingga insiden paling rumit biasanya terjadi di perbatasan antarsistem.

## Bab 101. Lanskap Sistem Pembayaran

| Sistem | Jenis | Batas nominal per transaksi | Waktu layanan |
|---|---|---|---|
| BI-FAST | Transfer ritel real-time | Hingga Rp250 juta; biaya ke nasabah maksimal Rp2.500 | 24/7 |
| SKNBI | Kliring ritel | Hingga Rp1 miliar | Jam operasional hari kerja |
| BI-RTGS | Transfer nilai besar | Di atas Rp100 juta | Jam operasional hari kerja |
| Switching ATM/debit (GPN) | Tarik tunai, transfer, belanja debit domestik | Sesuai limit bank dan kanal | 24/7 |
| QRIS | Pembayaran QR standar nasional | Sesuai ketentuan BI | 24/7 |

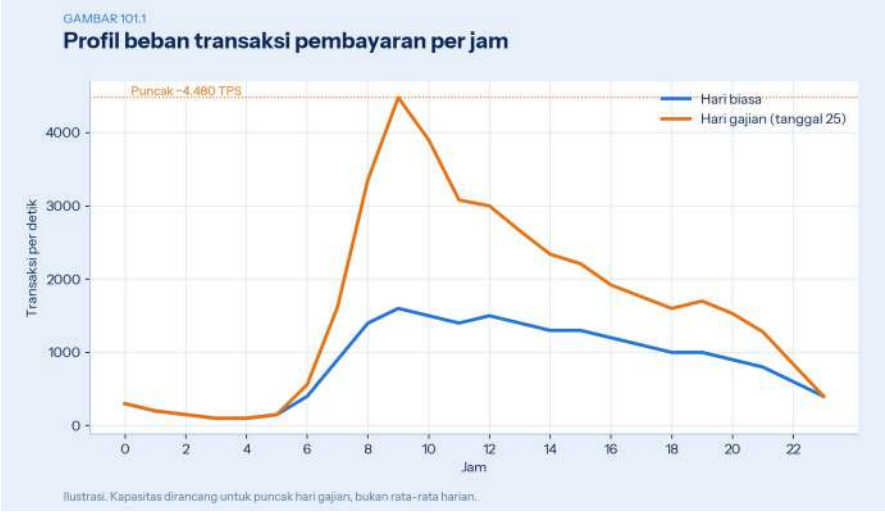
Angka di atas bersumber dari FAQ BI-FAST dan FAQ skema harga BI-FAST dari Bank Indonesia saat peluncuran. Bank boleh menetapkan limit BI-FAST lebih rendah sesuai risk appetite, dan BI meninjau batas ini secara berkala, jadi selalu cek ketentuan terbaru.

Menurut BI, BI-FAST adalah modernisasi SKNBI. Layanan transfer kredit dan debit SKNBI dialihkan bertahap ke BI-FAST, sementara SKNBI ke depan difokuskan untuk warkat cek dan bilyet giro.

| Format pesan | Dipakai untuk |
|---|---|
| ISO 8583 | Transaksi kartu, ATM, dan EDC |
| ISO 20022 | BI-FAST dan sistem pembayaran modern |
| SWIFT MT/MX | Transfer lintas negara |
| Format flat file | Kliring warkat, pembayaran massal, interface lama |

## 101.1 Analisis: Merancang Kapasitas untuk Puncak Pembayaran

Beban pembayaran tidak rata sepanjang hari atau sepanjang bulan. Hari gajian, akhir bulan, dan menjelang hari raya bisa menghasilkan puncak beberapa kali lipat dari hari biasa. Kapasitas yang dirancang dari rata-rata akan gagal justru pada hari yang paling penting bagi nasabah.




*Gambar 101.1 — Profil beban transaksi pembayaran per jam*

$$F_{\text{puncak}} = \frac{TPS_{\text{puncak}}}{TPS_{\text{puncak hari biasa}}} \qquad K_{\text{desain}} = TPS_{\text{puncak}} \times (1+g)^t \times (1+h)$$

g adalah pertumbuhan tahunan, t horizon perencanaan dalam tahun, dan h cadangan keamanan.

Dalam ilustrasi, puncak hari biasa sekitar 1.600 transaksi per detik, sedangkan puncak hari gajian sekitar 4.480. Faktor puncaknya 2,8 kali.

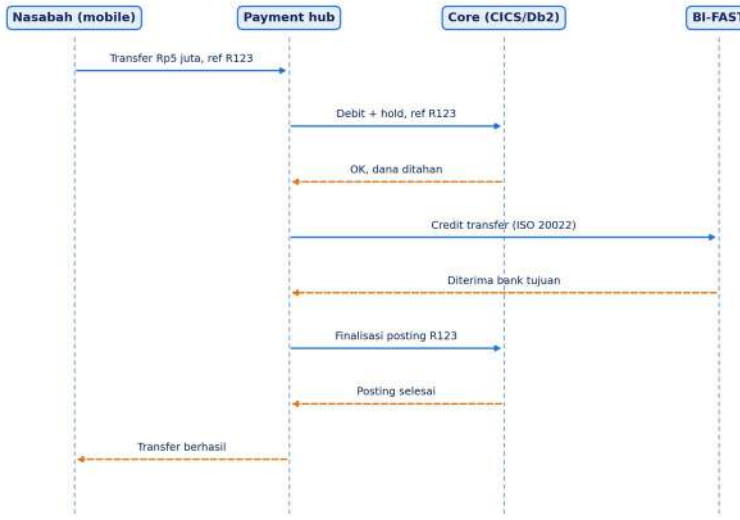
Dengan pertumbuhan 25% per tahun selama dua tahun dan cadangan 30%, kapasitas desain menjadi sekitar 4.480 × 1,5625 × 1,3 ≈ 9.100 transaksi per detik. Angka ini jauh berbeda dari kesimpulan yang diambil dari rata-rata harian.

| Komponen | Yang harus sanggup di puncak |
|---|---|
| Gateway dan z/OS Connect | Koneksi bersamaan dan TLS handshake |
| CICS | Jumlah AOR dan MXT (Bab 32.3) |
| Db2 | Lock, log, dan buffer pool |
| MQ ke payment hub | Kedalaman antrean dan laju pengurasan (Bab 34.1) |
| Kapasitas CPU dan MSU | 4HRA di jam puncak; batas capping |

Uji beban sebelum hari-hari puncak yang sudah diketahui, seperti menjelang hari raya, dengan profil yang meniru pola jam pada gambar di atas. Hasilnya menunjukkan komponen mana yang pertama kali jenuh, sehingga tambahan kapasitas bisa diarahkan tepat sasaran.

## Bab 102. Arsitektur Integrasi dan Idempotensi

Pola yang paling umum adalah *payment hub* di platform terbuka yang berbicara dengan jaringan luar, dan core banking di z/OS yang menjadi pemilik saldo. Keduanya terhubung lewat MQ atau API z/OS Connect.




Setiap langkah membawa referensi unik yang sama. Core banking menolak permintaan kedua dengan referensi yang sudah diproses, sehingga pengulangan akibat timeout tidak mendebet nasabah dua kali.

## 102.1 Algoritma penanganan timeout

Timeout adalah situasi paling berbahaya dalam pembayaran: pengirim tidak tahu apakah transaksi berhasil. Jangan pernah langsung membalik (reversal) tanpa memastikan status.

```
ALGORITMA TanganiTimeout(ref)
  status <- tanya_status(ref)                -- status
inquiry ke jaringan tujuan
  ulangi sampai status pasti atau batas_percobaan habis:
      tunggu jeda bertahap
      status <- tanya_status(ref)
  jika status = BERHASIL maka finalisasi_posting(ref)
  jika status = GAGAL    maka lepas_hold(ref); beri tahu
nasabah
  jika status tetap TIDAK_DIKETAHUI maka
      pindahkan ke antrean suspense dengan ref
      biarkan dana tetap ditahan
      selesaikan lewat rekonsiliasi (Bab 103)
```

## 102.2 Kewajiban sisi mainframe

| Kewajiban | Cara di z/OS |
|---|---|
| Idempoten per referensi | Unique index Db2 pada kolom referensi |
| Tahan dulu, posting kemudian | Hold di modul DDA sebelum pesan dikirim keluar |
| Tidak kehilangan pesan | MQ persistent message dan unit of work bersama Db2 (two-phase commit via RRS) |
| Tetap melayani saat EOD | Mode stand-in atau pemrosesan online selama batch |
| Jejak audit lengkap | Log transaksi dengan waktu, kanal, dan referensi eksternal |

## 102.3 Analisis: Merancang Penyimpanan Kunci Idempotensi

Idempotensi membutuhkan tempat untuk mengingat transaksi yang sudah diproses. Tempat itu adalah tabel kunci idempotensi: setiap permintaan membawa kunci unik, dan sebelum memproses, sistem memeriksa apakah kunci itu sudah pernah dilihat.

| Kasus | Kunci sudah ada? | Payload sama? | Respons |
|---|---|---|---|
| Permintaan baru | Tidak | – | Proses, simpan kunci dan hasilnya |
| Retry setelah timeout | Ya | Ya | Kembalikan hasil yang tersimpan tanpa memproses ulang |
| Kunci dipakai ulang untuk transaksi lain | Ya | Tidak | Tolak dengan kode konflik |
| Permintaan asli masih diproses | Ya, status “sedang diproses” | Ya | Minta klien mencoba lagi nanti |

Kunci harus disimpan setidaknya selama klien mungkin mencoba ulang. Jika kunci sudah dihapus lalu retry datang, transaksi akan diproses dua kali.

$$R_{\text{kunci}} \geq \max(\text{jendela retry semua klien}) + T_{\text{margin}} \qquad V = N_{\text{per hari}} \times R_{\text{hari}} \times s_{\text{baris}}$$

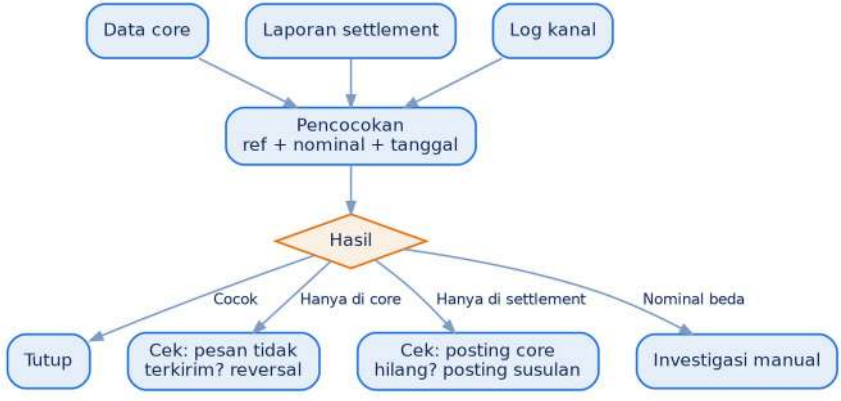
Contoh: 20 juta transaksi per hari, retensi 3 hari, dan sekitar 200 byte per baris, termasuk index, membutuhkan sekitar 12 GB. Angka ini kecil untuk Db2, tetapi tabel ini termasuk yang paling sering diakses, sehingga desainnya penting.

- **Index unik** pada kunci idempotensi, sehingga dua permintaan bersamaan dengan kunci yang sama tidak bisa sama-sama lolos.
- **Partisi per hari** agar data lama dibuang dengan menggilir partisi (Bab 152), bukan dengan DELETE jutaan baris yang membebani log.
- **Kunci dari klien, bukan dari server.** Kunci yang dibuat server tidak membantu, karena retry dari klien akan membawa kunci baru.
- **Hasil disimpan bersama kunci**, agar retry mendapat jawaban yang sama persis dengan permintaan asli.

Kunci idempotensi juga memudahkan rekonsiliasi. Setiap transaksi dari payment hub dapat dicocokkan satu per satu dengan catatan di core, dan transaksi yang sampai di satu sisi tetapi tidak di sisi lain langsung terlihat.

## Bab 103. Suspense dan Rekonsiliasi

Rekonsiliasi membandingkan catatan core banking dengan catatan pihak lain: laporan settlement dari BI atau lembaga switching, serta log kanal. Rekonsiliasi harian adalah jaring pengaman terakhir untuk transaksi yang gagal di tengah jalan.



Selisih rekonsiliasi per hari dihitung dari total yang tidak cocok:

$$\Delta = \sum \text{nominal}_{\text{core}} - \sum \text{nominal}_{\text{settlement}}$$

| Kategori selisih | Penyebab umum | Tindakan |
|---|---|---|
| Hanya di core | Pesan tidak sampai ke jaringan, atau ditolak tanpa balasan | Reversal ke nasabah setelah status dipastikan gagal |
| Hanya di settlement | Posting core gagal setelah dana berpindah | Posting susulan dari akun suspense |
| Nominal berbeda | Biaya atau kurs dihitung berbeda | Investigasi dan koreksi jurnal |
| Duplikat | Pengulangan tanpa pemeriksaan idempotensi | Reversal duplikat, perbaiki kontrol |

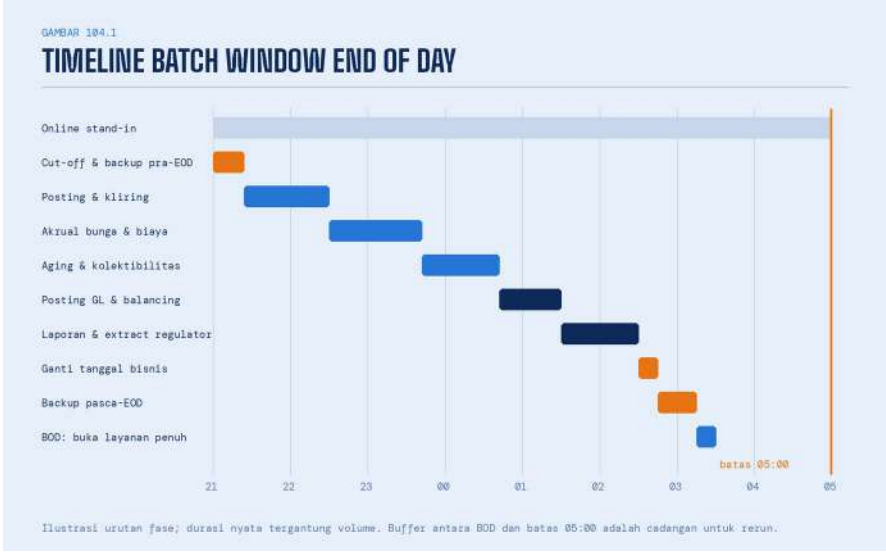
Akun suspense yang berumur lebih dari satu hari kerja harus dieskalasi. Saldo suspense yang terus tumbuh adalah sinyal awal masalah integrasi.

# 🌙 Bagian XXIV — Pendalaman End of Day

EOD adalah saat core banking paling rentan: volume besar, tenggat keras, dan setiap kesalahan berdampak ke saldo nasabah. Bagian ini melanjutkan Bab 35, 58, dan 61 menjadi panduan rancangan dan operasi EOD yang lengkap.

## Bab 104. Anatomi Batch Window

Batch window adalah rentang waktu dari cut-off sampai layanan kembali penuh. Semua fase EOD, rerun darurat, dan backup harus muat di dalamnya.




*Gambar 104.1 — Timeline batch window End of Day*

Pemakaian window dan sisa cadangannya dihitung sebagai berikut:

$$U = \frac{T_{\text{EOD}}}{T_{\text{window}}} \qquad \text{Cadangan} = T_{\text{window}} - T_{\text{EOD}}$$

Cadangan harus cukup untuk menjalankan ulang fase terpanjang. Jika fase akrual butuh 70 menit, cadangan di bawah 70 menit berarti satu kegagalan saja sudah membuat bank terlambat membuka layanan.

Durasi EOD biasanya stabil sepanjang bulan, lalu melonjak di akhir bulan karena proses bulanan menumpuk. Lonjakan inilah yang menentukan kebutuhan kapasitas, bukan rata-rata harian.


*Ilustrasi · data rekaan*

## 104.1 Analisis: Utilisasi Batch Window

Seberapa aman batch window bisa diukur dengan satu rasio sederhana:

$$U_{\text{window}} = \frac{T_{\text{EOD}}}{T_{\text{window}}} \qquad \text{syarat rerun}: \; T_{\text{EOD}} + \max_{f} T_f \le T_{\text{window}}$$

Syarat kedua penting: window harus cukup untuk menjalankan ulang fase terpanjang jika fase itu gagal di akhir. Tanpa ruang untuk satu kali rerun, satu kegagalan saja langsung berarti keterlambatan membuka layanan.

| Kondisi | T_EOD | Window | Utilisasi | Rerun fase terpanjang (70 menit) muat? |
|---|---|---|---|---|
| Hari biasa | 5,6 jam | 8 jam | 70% | Ya, sisa sekitar 1,2 jam |
| Akhir bulan | 7,4 jam | 8 jam | 92,5% | Tidak |

| Kondisi | T_EOD | Window | Utilisasi | Rerun fase terpanjang (70 menit) muat? |
|---|---|---|---|---|
| Akhir tahun | 9,0 jam | 8 jam | 112% | Tidak; window harus diperpanjang |

Tabel ini menunjukkan mengapa rata-rata harian menyesatkan. Hari biasa terlihat nyaman, tetapi akhir bulan sudah melanggar syarat rerun. Akhir tahun bahkan tidak muat sama sekali tanpa pengaturan khusus. Untuk akhir tahun, perpanjangan window biasanya disepakati dengan bisnis jauh-jauh hari. Untuk akhir bulan, perlu dipilih antara mempercepat jalur kritis (Bab 22.1) atau menyiapkan prosedur khusus yang menerima risiko keterlambatan.

Pantau utilisasi window sebagai tren bulanan. Utilisasi akhir bulan yang naik 2–3 poin persentase setiap bulan karena pertumbuhan data adalah peringatan yang jelas: dalam beberapa bulan, akhir bulan biasa pun tidak akan muat.

## Bab 105. Tabel Kontrol Fase

Tabel kontrol adalah satu-satunya sumber kebenaran tentang fase mana yang sudah selesai (Bab 58). Rancangan di bawah dipakai sebagai contoh di Db2 for z/OS.

```
CREATE TABLE BANK.EOD_KONTROL
( TGL_BISNIS     DATE          NOT NULL,
  FASE           CHAR(8)       NOT NULL,
  RUN_ID         INTEGER       NOT NULL WITH DEFAULT 0,
  STATUS         CHAR(10)      NOT NULL WITH DEFAULT
'BELUM',
  MULAI          TIMESTAMP,
  SELESAI        TIMESTAMP,
  JML_RECORD     BIGINT,
  TOTAL_KONTROL  DECIMAL(19,2),
  JOBNAME        CHAR(8),
  JOBID          CHAR(8),
  PRIMARY KEY (TGL_BISNIS, FASE)
) IN BANKDB.TSKONTRL;
```

```
CREATE UNIQUE INDEX BANK.XEODKTL1
  ON BANK.EOD_KONTROL (TGL_BISNIS, FASE);
```

Sebuah fase “mengklaim” dirinya dengan satu UPDATE atomik. Jika tidak ada baris yang berubah, berarti fase sudah selesai atau sedang dijalankan proses lain, dan job harus berhenti.

```
UPDATE BANK.EOD_KONTROL
   SET STATUS  = 'BERJALAN',
       RUN_ID  = RUN_ID + 1,
       MULAI   = CURRENT TIMESTAMP,
       JOBNAME = :JOBNAME,
       JOBID   = :JOBID
 WHERE TGL_BISNIS = :TGL
   AND FASE       = :FASE
   AND STATUS IN ('BELUM', 'GAGAL');
-- Jumlah baris terpengaruh (SQLERRD(3)) = 0  ->  jangan
jalankan fase
```

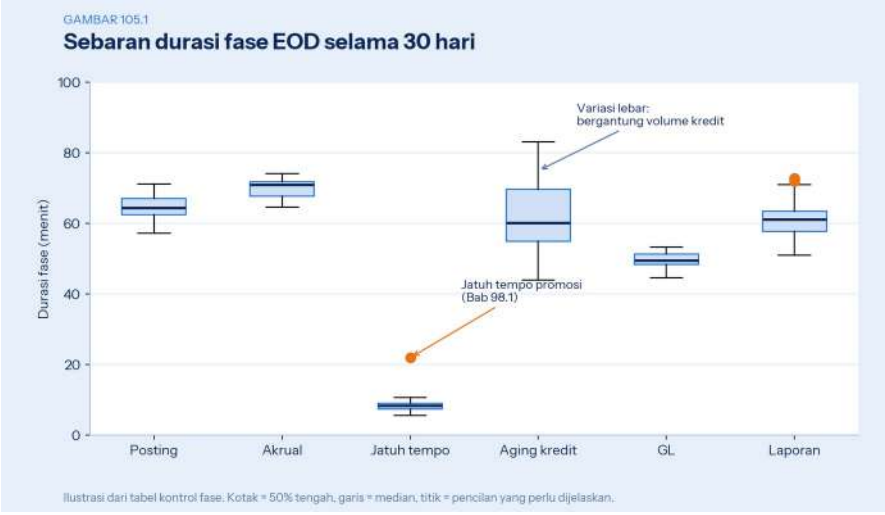
## 105.1 Teknik membuat fase aman diulang

| Teknik | Cara kerja | Cocok untuk |
|---|---|---|
| Hapus-lalu-sisip per tanggal | Hapus semua hasil fase untuk tanggal bisnis itu, lalu hitung ulang | Tabel hasil harian: akrual, laporan |
| MERGE (upsert) | Sisip jika belum ada, perbarui jika sudah | Tabel saldo dan ringkasan |
| Penanda per record | Kolom TGL_PROSES diperiksa sebelum memproses record | Posting ke rekening |
| Before image | Salinan data sebelum fase, dipulihkan bila gagal | Fase non-idempoten yang tak bisa diubah |
| Referensi unik | Unique index menolak jurnal duplikat | Posting GL |

## 105.2 Analisis: Membaca Sebaran Durasi Fase

Tabel kontrol fase mencatat waktu mulai dan selesai setiap fase setiap malam. Setelah beberapa pekan, tabel ini menjadi sumber analisis yang

sangat kaya. Bukan hanya rata-rata durasi yang penting, tetapi juga sebarannya.




*Gambar 105.1 — Sebaran durasi fase EOD selama 30 hari*

Fase dengan sebaran sempit, seperti Akrual dan GL dalam ilustrasi, dapat diprediksi dengan baik. Fase dengan sebaran lebar, seperti Aging kredit, bergantung pada volume data yang berubah-ubah. Titik pencilan, seperti Jatuh tempo pada tanggal promosi, adalah hari-hari yang perlu dijelaskan, dan biasanya bisa diramalkan bila penyebabnya diketahui (Bab 98.1).

Ambang peringatan per fase dapat dihitung langsung dari tabel kontrol:

```
SELECT FASE,
       AVG(TIMESTAMPDIFF(4, CHAR(WAKTU_SELESAI -
WAKTU_MULAI)))    AS RATA_MNT,
       STDDEV(TIMESTAMPDIFF(4, CHAR(WAKTU_SELESAI -
WAKTU_MULAI))) AS SD_MNT
FROM   EODCTL.KONTROL_FASE
WHERE  STATUS = 'SELESAI'
  AND  TGL_BISNIS > CURRENT DATE - 30 DAYS
GROUP BY FASE;
```

$$\text{Ambang}_f = \bar{T}_f + 2\,\sigma_f$$

Fase yang berjalan melewati ambangnya memicu peringatan dini, jauh sebelum EOD secara keseluruhan terlambat. Dengan dua simpangan baku,

sekitar satu dari empat puluh malam normal akan tetap memicu peringatan bila durasinya kurang lebih terdistribusi normal. Frekuensi ini cukup jarang untuk tidak mengganggu, tetapi cukup peka untuk menangkap perlambatan nyata.

Pisahkan perhitungan untuk hari biasa dan akhir bulan. Mencampur keduanya membuat ambang terlalu longgar untuk hari biasa dan terlalu ketat untuk akhir bulan.

## Bab 106. Kalender Bisnis dan Periode Khusus

Kesalahan kalender adalah penyebab insiden EOD yang paling memalukan, karena mudah dicegah. Kalender bisnis harus menjadi tabel yang dipelihara, bukan logika yang tersebar di program.

| Situasi | Risiko | Perlakuan yang disarankan |
|---|---|---|
| Libur nasional dan cuti bersama | Jatuh tempo dan tunggakan bergeser | Tabel kalender diperbarui begitu SKB diterbitkan |
| Akhir bulan jatuh di hari libur | Proses EOM dijalankan pada hari yang salah | EOM mengikuti hari kerja terakhir, akrual tetap hingga hari kalender terakhir |
| 29 Februari | Rumus basis 365 atau 366 hari tidak konsisten | Tetapkan basis hari per produk dan uji tahunan |
| Akhir tahun | Penutupan laba rugi, laporan tahunan, volume tertinggi | Latihan EOY di UAT dengan data produksi yang di-mask |
| Hari kerja berturut-turut libur panjang | Akrual beberapa hari sekaligus | Akrual per hari kalender, bukan per EOD |

```
ALGORITMA HariKerjaBerikut(tgl)
  t <- tgl + 1 hari
  selama t adalah Sabtu, Minggu, atau ada di TABEL_LIBUR:
      t <- t + 1 hari
  kembalikan t

ALGORITMA AkrualSetelahLibur(tgl_bisnis_lama,
tgl_bisnis_baru)
  untuk setiap hari kalender d dari tgl_bisnis_lama sampai
tgl_bisnis_baru - 1:
      akrual_harian(d)            -- bunga tetap berjalan
pada hari libur
```

## 106.1 Analisis: Kasus Tepi Kalender

Sebagian besar kesalahan kalender bisnis tidak muncul pada hari biasa. Kesalahan itu muncul pada tanggal-tanggal langka yang hanya terjadi sekali setahun, atau bahkan sekali dalam empat tahun. Karena jarang terjadi, tanggal-tanggal ini juga jarang diuji.

| Kasus tepi | Mengapa berbahaya | Uji yang disarankan |
|---|---|---|
| Akhir bulan jatuh pada hari libur | EOM harus dijalankan pada hari kerja sebelumnya atau sesudahnya sesuai kebijakan; akrual bisa terhitung ganda atau terlewat | Simulasikan EOM pada Sabtu dan Minggu |
| 29 Februari | Basis hari 365 atau 366; tanggal jatuh tempo “satu tahun” dari 29 Februari | Uji deposito dan kredit yang dibuka pada 29 Februari |
| Libur beruntun (hari raya, cuti bersama) | Beberapa hari kalender diproses dalam satu EOD; bunga harus dihitung untuk setiap hari kalender | Jalankan EOD yang melompati lima hari kalender |
| Libur yang ditetapkan mendadak | Kalender harus diubah dengan cepat dan aman | Prosedur perubahan kalender darurat dengan persetujuan ganda |
| Akhir tahun | EOY, pajak, laporan tahunan, dan tutup buku sekaligus | Gladi EOY di UAT dengan data setara |
| Jatuh tempo pada tanggal 31 di bulan tanpa tanggal 31 | Aturan penyesuaian tanggal harus konsisten | Deposito satu bulan yang dibuka 31 Januari |

Cara efektif menguji kasus tepi adalah menjalankan EOD di UAT dengan tanggal sistem bisnis yang disetel ke tanggal-tanggal tersebut, memakai tabel kalender yang sama dengan produksi. Karena tanggal bisnis diambil dari tabel kalender (Bab 106), bukan dari jam sistem, uji ini tidak memerlukan pengubahan jam z/OS.

Simpan daftar kasus tepi sebagai skenario regresi tetap. Setiap perubahan pada modul bunga, jatuh tempo, atau kalender harus lolos seluruh skenario ini sebelum masuk produksi. Tambahkan juga pemeriksaan tahunan: setiap November, pastikan kalender tahun berikutnya sudah

dimuat lengkap, termasuk cuti bersama yang sudah diumumkan pemerintah.

# Bab 107. Skenario EOD Bermasalah dan Runbook

## RB-041: Akrual terindikasi berjalan dua kali

- **Gejala:** total akrual hari ini sekitar dua kali proyeksi; tabel kontrol menunjukkan `RUN_ID` lebih dari 1 untuk fase akrual.
- **Diagnosis:** bandingkan jumlah record akrual per rekening untuk tanggal bisnis itu; periksa apakah fase dijalankan ulang tanpa pemulihan.
- **Tindakan:** hentikan EOD sebelum posting GL; hapus hasil akrual tanggal itu (teknik hapus-lalu-sisip) atau pulihkan before image; jalankan ulang sekali; verifikasi total.
- **Pencegahan:** klaim fase lewat UPDATE atomik (Bab 105) dan verifikasi total kontrol terhadap proyeksi.

## RB-042: Tanggal bisnis tidak maju atau maju dua kali

- **Gejala:** layanan pagi memakai tanggal kemarin, atau melompati satu hari.
- **Diagnosis:** periksa fase ganti tanggal di tabel kontrol dan tabel kalender.
- **Tindakan:** jangan membuka layanan; koreksi tanggal lewat fungsi aplikasi resmi; jika sudah ada transaksi dengan tanggal salah, koordinasikan koreksi dengan akuntansi.
- **Pencegahan:** ganti tanggal hanya boleh berjalan jika semua fase sebelumnya berstatus SELESAI dan GL seimbang.

## RB-043: EOD diproyeksikan melewati batas window

- **Gejala:** proyeksi waktu selesai melewati batas buka layanan.
- **Diagnosis:** identifikasi fase yang lambat; periksa CPU, capping, contention, dan volume dibanding biasanya.

- **Tindakan:** naikkan prioritas WLM job EOD kritikal; tunda fase non-kritikal (laporan yang bisa menyusul); aktifkan kapasitas sementara bila tersedia; putuskan pembukaan layanan bertahap bersama bisnis.
- **Pencegahan:** alert proyeksi (Bab 108) dan uji beban EOM setiap kuartal.

### RB-044: Saldo suspense pembayaran menumpuk

- **Gejala:** saldo suspense BI-FAST atau switching naik tajam dalam beberapa jam.
- **Diagnosis:** hitung transaksi dengan status tidak diketahui; periksa koneksi payment hub dan antrean MQ.
- **Tindakan:** pulihkan koneksi; jalankan status inquiry massal; selesaikan per kategori (Bab 103).
- **Pencegahan:** alert jumlah transaksi timeout per menit, bukan hanya saldo akhir hari.

### RB-045: Pemrosesan jatuh tempo deposito sangat lambat

- **Gejala:** fase jatuh tempo berjalan jauh lebih lama pada tanggal tertentu.
- **Diagnosis:** hitung jumlah deposito jatuh tempo hari itu; biasanya menumpuk setelah libur panjang atau pada tanggal promosi.
- **Tindakan:** biarkan selesai bila cadangan window cukup; jika tidak, jalankan paralel per rentang nomor rekening bila program mendukung.
- **Pencegahan:** proyeksi volume jatuh tempo tujuh hari ke depan dan desain fase yang bisa dipartisi.

### RB-046: Kolektibilitas melonjak massal dalam satu malam

- **Gejala:** ribuan debitur turun kolektibilitas sekaligus.
- **Diagnosis:** periksa tabel kalender (libur yang tidak terdaftar menambah hari tunggak) dan proses autodebet yang gagal.

- **Tindakan:** jangan kirim data ke SLIK sebelum dikonfirmasi; koreksi kalender atau jalankan ulang autodebet, lalu ulang aging.
- **Pencegahan:** verifikasi perubahan kolektibilitas harian terhadap batas wajar sebelum fase dilanjutkan.

### RB-047: Extract data regulator gagal atau terlambat

- **Gejala:** job extract abend atau file tidak terkirim sebelum tenggat pelaporan.
- **Diagnosis:** periksa step yang gagal dan validasi yang ditolak di aplikasi pelaporan.
- **Tindakan:** perbaiki data sumber lewat jalur resmi; jalankan ulang extract; informasikan unit pelaporan tentang potensi keterlambatan.
- **Pencegahan:** extract lebih awal dari tenggat dan validasi skema sebelum pengiriman (Bagian XXV).

### RB-048: EOM gagal saat kredit bunga simpanan

- **Gejala:** fase kredit bunga bulanan abend atau total pajak tidak cocok.
- **Diagnosis:** periksa parameter tarif pajak dan ambang saldo; cari rekening dengan data tidak valid.
- **Tindakan:** pulihkan titik awal fase; koreksi parameter atau data; jalankan ulang; verifikasi total bunga, pajak, dan jurnal GL.
- **Pencegahan:** uji EOM di UAT beberapa hari sebelum akhir bulan setiap kali parameter berubah.

## Bab 108. Metrik dan Pencegahan

EOD yang sehat dikelola dengan angka. Proyeksi waktu selesai yang dihitung sepanjang malam memberi waktu untuk bertindak sebelum batas terlewati.

$$T_{\text{selesai}} = T_{\text{sekarang}} + \sum_{f \in \text{sisa}} \tilde{d}_f \times \frac{V_{f,\,\text{hari ini}}}{\tilde{V}_f}$$

Tanda gelombang pada d dan V berarti median durasi dan median volume fase f dalam 30 hari terakhir. Jika proyeksi melewati batas window dikurangi cadangan, jalankan RB-043.

| Metrik | Target contoh | Tindakan jika meleset |
|---|---|---|
| Pemakaian window (U) | Di bawah 75% hari biasa, di bawah 90% EOM | Optimasi fase terpanjang atau tambah paralelisme |
| Jumlah rerun per bulan | 0–1 | Analisis akar masalah setiap rerun |
| Saldo suspense akhir hari | Mendekati nol | Perbaiki integrasi yang menyumbang suspense |
| GL seimbang sebelum ganti tanggal | 100% malam | Wajib, tanpa pengecualian |
| Selisih total kontrol vs proyeksi | Dalam toleransi yang disepakati | Tahan fase berikutnya dan investigasi |

# Bagian XXV — Pelaporan Regulator

Pelaporan regulator adalah konsumen data core banking yang paling tidak kenal kompromi: tenggatnya tetap, formatnya ketat, dan keterlambatan berujung sanksi. System engineer bertanggung jawab memastikan data keluar dari mainframe tepat waktu, lengkap, dan dapat ditelusuri.

## Bab 109. Lanskap Pelaporan

| Laporan | Penerima | Isi utama | Sumber data di core |
|---|---|---|---|
| LBUT melalui BI-ANTASENA | BI, digunakan bersama OJK dan LPS | Laporan bank umum terintegrasi berbasis metadata | GL, simpanan, kredit, valas, CIF |
| SLIK | OJK | Informasi debitur dan fasilitas kredit | Modul kredit dan CIF |
| Laporan transaksi keuangan (LTKM, LTKT, transfer dana dari/ke luar negeri) | PPATK | Transaksi mencurigakan, tunai, dan lintas negara | Log transaksi, CIF, profil APU-PPT |
| Data Single Customer View | LPS | Data simpanan per nasabah untuk penjaminan | Modul simpanan dan CIF |
| Laporan publikasi | OJK dan publik | Laporan keuangan periodik | GL |

LBUT diatur dalam PBI Nomor 21/9/PBI/2019. Tujuannya menggantikan banyak laporan terpisah dengan satu sistem berbasis metadata yang dipakai bersama BI, OJK, dan LPS. Menurut OJK, BI-ANTASENA mulai berjalan 31 Desember 2019 dan diimplementasikan penuh pada 1 Juli 2021. Periode dan tenggat tiap laporan mengikuti ketentuan terbaru masing-masing otoritas.

### 109.1 Analisis: Kualitas Data Sebelum Laporan Dikirim

Laporan regulator yang ditolak karena kesalahan format bisa diperbaiki dan dikirim ulang. Laporan yang diterima padahal isinya salah jauh lebih

berbahaya, karena kesalahannya baru ditemukan saat pemeriksaan dan bisa berujung pada sanksi. Karena itu validasi sebelum pengiriman harus berlapis.

| Lapisan | Pemeriksaan | Contoh temuan |
|---|---|---|
| Format | Struktur, tipe data, panjang field | Tanggal dengan format salah |
| Kelengkapan | Field wajib terisi, jumlah record wajar | Rekening tanpa kode sektor ekonomi |
| Konsistensi internal | Aturan antar-field dalam laporan yang sama | Kolektibilitas 1 tetapi tunggakan 120 hari |
| Rekonsiliasi | Total laporan cocok dengan GL | Total kredit di laporan berbeda dengan neraca |
| Tren | Perubahan dibanding periode lalu wajar | Jumlah debitur turun 40% dalam sebulan |

$$\text{Tingkat kesalahan} = \frac{\text{record gagal validasi}}{\text{total record}} \qquad \Delta_{\text{rekon}} = \text{Total}_{\text{laporan}} - \text{Total}_{\text{GL}}$$

Lapisan rekonsiliasi dan tren adalah yang paling sering dilewatkan, padahal keduanya menangkap kesalahan yang lolos dari validasi format. Laporan bisa sempurna secara format tetapi salah secara isi karena satu job ekstraksi berjalan dengan tanggal yang salah.

Sistem ekstraksi di mainframe (Bab 110) sebaiknya menghasilkan ringkasan kontrol bersama setiap file laporan: jumlah record, total nominal per kelompok, dan tanggal data. Ringkasan ini dibandingkan otomatis dengan GL sebelum file diserahkan ke aplikasi pelaporan. Selisih apa pun menghentikan proses dan memicu investigasi.

Simpan hasil validasi setiap periode. Riwayat tingkat kesalahan dan selisih rekonsiliasi dari bulan ke bulan menunjukkan apakah kualitas data membaik, dan menjadi bukti yang kuat saat pemeriksaan.

## Bab 110. Pipeline Data Pelaporan dari Mainframe

Pipeline pelaporan yang baik memisahkan tiga tanggung jawab: extract dari core, transformasi dan validasi, lalu pengiriman oleh aplikasi

pelaporan. Mainframe biasanya menangani tahap pertama, dan sebagian bank juga tahap kedua.

## 110.1 Prinsip data yang dituntut regulator

BI merumuskan kebutuhan data laporan sebagai lengkap, akurat, kini, dan utuh (LAKU). Setiap prinsip diterjemahkan menjadi pemeriksaan teknis di pipeline.

| Prinsip | Pemeriksaan teknis |
|---|---|
| Lengkap | Jumlah record extract sama dengan jumlah record sumber untuk tanggal yang sama |
| Akurat | Total nominal extract sama dengan saldo GL terkait (rekonsiliasi GL ke laporan) |
| Kini | Extract memakai data setelah EOD tanggal pelaporan, bukan data kemarin |
| Utuh | Tidak ada field wajib yang kosong; kode referensi valid terhadap tabel metadata |

```
ALGORITMA ValidasiExtract(laporan, tanggal)
  n_sumber  <- hitung record sumber (tanggal)
  n_extract <- hitung record file extract
  jika n_sumber <> n_extract maka GAGAL("jumlah record")
  jika total_nominal(extract) <> saldo_GL(akun_terkait,
tanggal) maka GAGAL("rekonsiliasi GL")
  jika tanggal_data(extract) <> tanggal maka
GAGAL("tanggal data")
  untuk setiap field wajib f:
      jika ada nilai kosong di f maka GAGAL("field wajib "
f)
  untuk setiap field berkode k:
      jika ada nilai k tidak ada di tabel referensi maka
GAGAL("kode tidak valid " k)
  tandai extract SIAP_KIRIM dan catat bukti validasi
```

## 110.2 Praktik operasional

- Jadwalkan extract sebagai fase pasca-EOD yang terdaftar di tabel kontrol, bukan job lepas.
- Simpan setiap file extract sebagai GDG dengan retensi sesuai kebijakan arsip, agar laporan lama bisa dibuktikan ulang.
- Catat lineage: dari tabel sumber mana dan dengan versi program apa setiap laporan dibuat.

- Tangani koreksi laporan sebagai proses resmi dengan jejak audit, bukan penggantian file diam-diam.
- Libatkan unit kepatuhan saat metadata laporan berubah; perubahan metadata sering berarti perubahan program extract.

# 🚨 Bagian XXVI — Respons Insiden Keamanan

Mainframe sering dianggap kebal, dan anggapan itulah risikonya. Serangan terhadap mainframe jarang lewat celah teknis; hampir selalu lewat kredensial yang dicuri, akses berprivilegi yang berlebihan, atau sistem terbuka yang terhubung ke core.

## Bab 111. Prinsip Penanganan

Respons insiden mengikuti fase yang umum dipakai di industri. Tabel berikut menerjemahkan setiap fase ke tindakan konkret di z/OS.

| Fase | Tujuan | Tindakan di z/OS |
|---|---|---|
| Persiapan | Siap sebelum insiden | Runbook, akses darurat teruji, SMF lengkap ke SIEM, Cyber Vault |
| Deteksi dan analisis | Tahu apa yang terjadi | Korelasi SMF 80, 30, 14/15, 119; pesan `ICH408I`; perubahan APF dan exit |
| Penahanan | Hentikan penyebaran | Revoke user, tutup port, isolasi LPAR, blokir koneksi dari sistem terbuka |
| Pemberantasan | Hapus penyebab | Cabut backdoor, kembalikan PARMLIB, ganti semua kredensial terkait |
| Pemulihan | Kembali beroperasi dengan aman | Restore data bersih, validasi, pembukaan layanan bertahap |
| Pembelajaran | Mencegah terulang | Laporan pasca-insiden, perbaikan kontrol, pembaruan runbook |

Tiga aturan yang tidak boleh dilanggar dalam insiden keamanan:

1. **Amankan bukti sebelum mengubah apa pun.** Salin SMF, SYSLOG/OPERLOG, dan unload RACF sebelum melakukan revoke atau restore.
2. **Jangan berkomunikasi lewat kanal yang mungkin disusupi.** Gunakan jalur darurat yang sudah disepakati.
3. **Libatkan tim keamanan, hukum, dan kepatuhan sejak awal.** Kewajiban pelaporan ke OJK dan otoritas lain punya tenggat sendiri.

## 111.1 Analisis: Retensi Snapshot Melawan Waktu Tinggal Penyerang

Snapshot yang tidak dapat diubah hanya berguna jika setidaknya satu snapshot dibuat sebelum penyerang mulai merusak data. Penyerang canggih sering berdiam lama di dalam jaringan sebelum bertindak, dan periode ini disebut *dwell time*. Jika retensi snapshot lebih pendek dari dwell time, semua snapshot yang tersedia mungkin sudah tercemar.

$$R_{\text{retensi}} \geq T_{\text{dwell}} + T_{\text{deteksi dan investigasi}}$$

Contoh: snapshot disimpan 7 hari, tetapi penyerang menanam perubahan tersembunyi 20 hari sebelum menjalankan enkripsi. Semua snapshot yang tersedia berasal dari masa ketika data sudah tercemar. Pemulihan harus kembali ke backup tape terisolasi yang lebih lama, dengan kehilangan data yang jauh lebih besar.

Karena kapasitas terbatas, retensi biasanya dibuat bertingkat:

| Tingkat | Frekuensi | Retensi | Tujuan |
|---|---|---|---|
| Rapat | Setiap beberapa jam | 2–3 hari | RPO kecil untuk kejadian yang cepat terdeteksi |
| Harian | Sekali sehari | Beberapa pekan | Kejadian yang terdeteksi setelah beberapa hari |
| Mingguan | Sekali seminggu | Beberapa bulan | Serangan dengan dwell time panjang |

| Tingkat | Frekuensi | Retensi | Tujuan |
|---|---|---|---|
| Tape terisolasi | Harian atau mingguan | Sesuai kebijakan | Garis pertahanan terakhir |

Pencarian snapshot bersih (Bab 114) menjadi jauh lebih cepat dengan struktur bertingkat seperti ini. Mulailah dari snapshot mingguan untuk menemukan rentang waktu yang bersih, lalu persempit ke snapshot harian dan rapat. Pencarian biner di atas beberapa ratus snapshot hanya membutuhkan beberapa kali validasi.

Kebijakan retensi ini harus ditinjau bersama tim keamanan informasi, karena perkiraan dwell time bergantung pada kemampuan deteksi bank sendiri. Semakin cepat serangan terdeteksi, semakin pendek retensi yang dibutuhkan.

## Bab 112. Deteksi dan Pengamanan Bukti

| Indikator | Sumber | Kenapa mencurigakan |
|---|---|---|
| Lonjakan `ICH408I` untuk satu user | SYSLOG, SMF 80 | Pencarian akses atau tebak kata sandi |
| User baru dengan `SPECIAL` atau `OPERATIONS` | SMF 80 | Eskalasi hak akses |
| Library ditambahkan ke APF secara dinamis | SMF 90, SYSLOG `SETPROG` | Jalur menjalankan kode authorized |
| Exit atau SVC baru | `D PROG,EXIT`, SMF 90 | Backdoor di level sistem |
| Akses baca massal ke dataset CIF | SMF 14, 42, audit Db2 | Eksfiltrasi data |
| Login dari alamat atau jam tidak lazim | SMF 30, 119 | Kredensial dicuri |
| FTP keluar berukuran besar | SMF 119 | Pencurian data |

Langkah pengamanan bukti yang dijalankan paling awal:

```
I SMF                                   Tutup dan arsipkan
dataset/logstream SMF aktif
D PROG,APF                              Simpan daftar APF
saat ini
```

```
D PROG,EXIT                              Simpan daftar exit
yang terdefinisi
D IPLINFO                                Konfigurasi IPL
saat ini
//UNLOAD   EXEC PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD SYSOUT=*
//INDD1    DD DISP=SHR,DSN=SYS1.RACFDB.PRIM
//OUTDD    DD
DSN=SEC.IR.RACF.UNLOAD.D260920,DISP=(NEW,CATLG),
//            SPACE=(CYL,(50,50)),RECFM=VB,LRECL=4096
```

Simpan salinan bukti di lokasi yang aksesnya dibatasi untuk tim investigasi, dan hitung hash-nya untuk menjaga rantai penguasaan bukti.

## 112.1 Analisis: Menjaga Rantai Bukti

Saat insiden keamanan terjadi, dorongan pertama tim teknis adalah memulihkan layanan secepat mungkin. Itu benar, tetapi ada satu hal yang harus dilakukan bersamaan: menjaga bukti. Tanpa bukti yang utuh, bank tidak bisa memastikan apa yang terjadi, data apa yang terdampak, dan apakah penyerang masih ada di dalam. Bank juga kesulitan memenuhi kewajiban pelaporan kepada regulator dan, bila ada, proses hukum.


*Gambar 112.1 — Rantai bukti insiden keamanan*

| Sumber bukti | Isi yang berguna | Risiko hilang |
|---|---|---|
| SMF 80 | Keputusan akses RACF, perubahan profil | Terhapus oleh siklus arsip biasa |
| SMF 30 | Job dan address space, siapa menjalankan apa | Sama |
| SMF 119 | Koneksi TCP/IP, FTP, dan transfer | Sama |
| SMF 14/15 | Dataset yang dibuka untuk dibaca atau ditulis | Sama |
| SYSLOG / OPERLOG | Perintah console dan pesan sistem | Retensi pendek di logstream |
| Unload database RACF | Kondisi profil dan hak akses saat kejadian | Berubah saat tim melakukan perbaikan |
| Snapshot data | Kondisi data saat kejadian | Tertimpa oleh pemulihan |

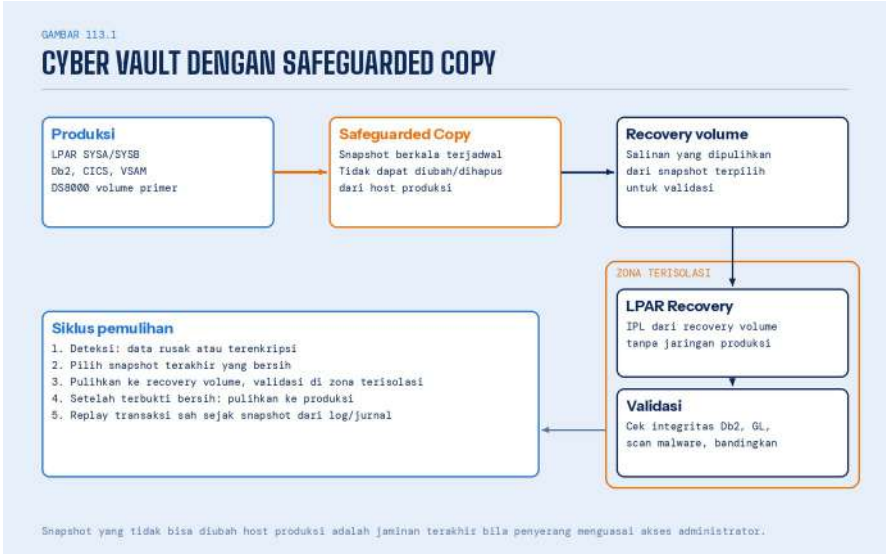
Linimasa adalah hasil akhir analisis bukti. Kejadian dari berbagai sumber diurutkan berdasarkan waktu, sehingga terlihat kapan penyerang masuk,

apa yang disentuh, dan kapan data keluar. Linimasa hanya akurat bila jam semua sistem sinkron. Di sinilah STP di mainframe dan NTP di sistem terbuka, yang merujuk sumber waktu yang sama, menjadi penting.

Siapkan prosedur pengamanan bukti sebelum insiden terjadi, dan uji dalam latihan tabletop. Pada saat insiden, tidak ada waktu untuk memikirkan dataset SMF mana yang harus disalin, atau siapa yang berwenang menghentikan siklus purge log.

## Bab 113. Ransomware, Korupsi Logis, dan Cyber Vault

Replikasi sinkron seperti Metro Mirror melindungi dari kegagalan hardware, tetapi juga menyalin data yang rusak atau terenkripsi ke situs DR dalam hitungan milidetik. Perlindungan terhadap korupsi logis membutuhkan salinan yang *tidak dapat diubah* dan dipisah waktu.




*Gambar 113.1 — Cyber Vault dengan Safeguarded Copy*

| Perlindungan | Melindungi dari | Tidak melindungi dari |
|---|---|---|
| Metro Mirror / Global Mirror | Kegagalan disk atau situs | Korupsi logis, ransomware |
| FlashCopy biasa | Kesalahan batch | Penyerang yang bisa |

| Perlindungan | Melindungi dari | Tidak melindungi dari |
|---|---|---|
| | | menghapus FlashCopy |
| Safeguarded Copy | Korupsi logis dan penghapusan oleh admin host | Kehilangan data sejak snapshot terakhir |
| Backup ke tape terisolasi (air gap) | Hampir semua skenario | Waktu pemulihan yang panjang |

Kehilangan data maksimum dalam skenario siber ditentukan oleh jarak waktu snapshot, bukan oleh replikasi:

$$RPO_{\text{siber}} \leq \Delta t_{\text{snapshot}} + T_{\text{deteksi terlambat}}$$

Jika korupsi baru disadari berhari-hari kemudian, snapshot beberapa jam terakhir juga sudah tercemar. Karena itu retensi snapshot harus lebih panjang daripada waktu deteksi terburuk.

## 113.1 Algoritma memilih snapshot bersih

```
ALGORITMA CariSnapshotBersih(daftar_snapshot)   -- urut
dari lama ke baru
  kiri <- 1; kanan <- jumlah(daftar_snapshot); terbaik <-
TIDAK_ADA
  selama kiri <= kanan:
      tengah <- (kiri + kanan) div 2
      pulihkan daftar_snapshot[tengah] ke recovery volume
      IPL LPAR recovery terisolasi
      jika validasi_bersih() maka           -- integritas
Db2, GL seimbang, tanpa malware
          terbaik <- tengah; kiri <- tengah + 1
      selain itu
          kanan <- tengah - 1
  kembalikan daftar_snapshot[terbaik]
```

Pencarian biner bekerja karena setiap snapshot setelah awal korupsi juga ikut tercemar, sehingga hasil validasi berurutan “bersih lalu kotor”. Jumlah uji pemulihan turun dari n menjadi sekitar log2(n); dengan 42 snapshot, cukup sekitar enam kali pemulihan untuk menemukan titik bersih terakhir.

## 113.2 Analisis: Frekuensi Snapshot, Retensi, dan Kapasitas

Merancang Safeguarded Copy berarti menyeimbangkan tiga hal yang saling tarik: seberapa sering snapshot dibuat (menentukan RPO), berapa lama disimpan (melindungi dari dwell time, Bab 111.1), dan berapa kapasitas yang tersedia.

$$C \approx \sum_{j=1}^{N_{\text{snapshot}}} \Delta_j$$

Δ adalah jumlah data unik yang berubah di antara dua snapshot berurutan. Jika data yang sama ditulis berulang kali di antara dua snapshot, data itu hanya dihitung sekali. Karena itu snapshot yang lebih jarang bisa lebih hemat per jam retensi. Sebaliknya, snapshot yang lebih sering menyimpan lebih banyak versi dari data yang sering berubah.

| Pilihan desain | RPO siber | Kebutuhan kapasitas | Waktu mencari snapshot bersih |
|---|---|---|---|
| Setiap jam, simpan 3 hari | Sangat kecil | Tinggi untuk data yang sering berubah | Cepat bila kejadian terdeteksi cepat |
| Setiap 4 jam, simpan 7 hari | Kecil | Menengah | Sedang |
| Harian, simpan 30 hari | Hingga satu hari | Menengah untuk retensi panjang | Cocok untuk dwell time panjang |
| Kombinasi bertingkat | Kecil untuk kejadian baru, tetap terlindung untuk yang lama | Paling efisien | Cepat dengan pencarian bertingkat |

Ukur laju perubahan nyata sebelum menetapkan desain. Laju perubahan data core banking sangat berbeda antara hari biasa, EOD, dan akhir bulan. Desain yang hanya dihitung dari rata-rata akan kehabisan kapasitas saat akhir bulan, dan snapshot tertua akan terhapus lebih cepat dari rencana. Pantau kapasitas Safeguarded seperti storage group, dengan proyeksi dan ambang peringatan (Bab 59.1).

Snapshot tidak berguna tanpa kemampuan untuk memulihkan dan memvalidasinya dengan cepat. Uji pemulihan dari snapshot ke lingkungan

terisolasi setidaknya sekali setiap kuartal, dan catat berapa lama validasi berlangsung.

# Bab 114. Runbook Insiden Keamanan

## RB-049: Kredensial user berprivilegi diduga bocor

- **Gejala:** login di luar jadwal, dari alamat asing, atau aktivitas yang tidak dikenali pemilik akun.
- **Diagnosis:** tarik SMF 30 dan 80 untuk user tersebut; bandingkan dengan tiket perubahan.
- **Tindakan:** amankan bukti; `ALTUSER userid REVOKE`; ganti kata sandi semua akun terkait; periksa perubahan yang dibuat user itu sejak waktu kompromi paling awal.
- **Pencegahan:** MFA untuk semua user berprivilegi dan akses darurat berbasis persetujuan.

## RB-050: Perubahan APF, exit, atau PARMLIB tidak sah

- **Gejala:** SIEM mendeteksi `SETPROG APF,ADD` atau perubahan member PARMLIB tanpa tiket.
- **Diagnosis:** identifikasi pelaku dari SMF; periksa isi library dan modul yang baru ditambahkan.
- **Tindakan:** cabut library dari APF secara dinamis; nonaktifkan exit; kembalikan PARMLIB dari baseline di Git; eskalasi sebagai insiden keamanan.
- **Pencegahan:** audit UPDATE ke PARMLIB dan library APF, serta perbandingan baseline otomatis setiap hari.

## RB-051: Korupsi data logis dalam skala besar

- **Gejala:** saldo tidak masuk akal, tabel terhapus, atau data terenkripsi di banyak objek sekaligus.
- **Diagnosis:** tentukan waktu paling awal kerusakan dari log Db2 dan SMF.

- **Tindakan:** hentikan replikasi yang bisa menyebarkan kerusakan bila diputuskan; jalankan algoritma snapshot bersih (Bab 113.1); pulihkan dan replay transaksi sah dari log; buka layanan setelah validasi.
- **Pencegahan:** Safeguarded Copy dengan retensi memadai dan latihan pemulihan siber tahunan.

## RB-052: Serangan tebak kata sandi ke TN3270 atau FTP

- **Gejala:** banyak `ICH408I` untuk banyak user dari alamat yang sama.
- **Diagnosis:** SMF 119 dan log firewall untuk sumber koneksi.
- **Tindakan:** blokir alamat di firewall; aktifkan pembatasan di IP filter z/OS bila tersedia; pastikan user terdampak ter-revoke otomatis dan dibuka lewat prosedur resmi.
- **Pencegahan:** TN3270 dan FTP hanya lewat TLS dari jaringan manajemen, tidak pernah dari jaringan umum.

# 🔁 Bagian XXVII — Prosedur Pemulihan Bencana Mendalam

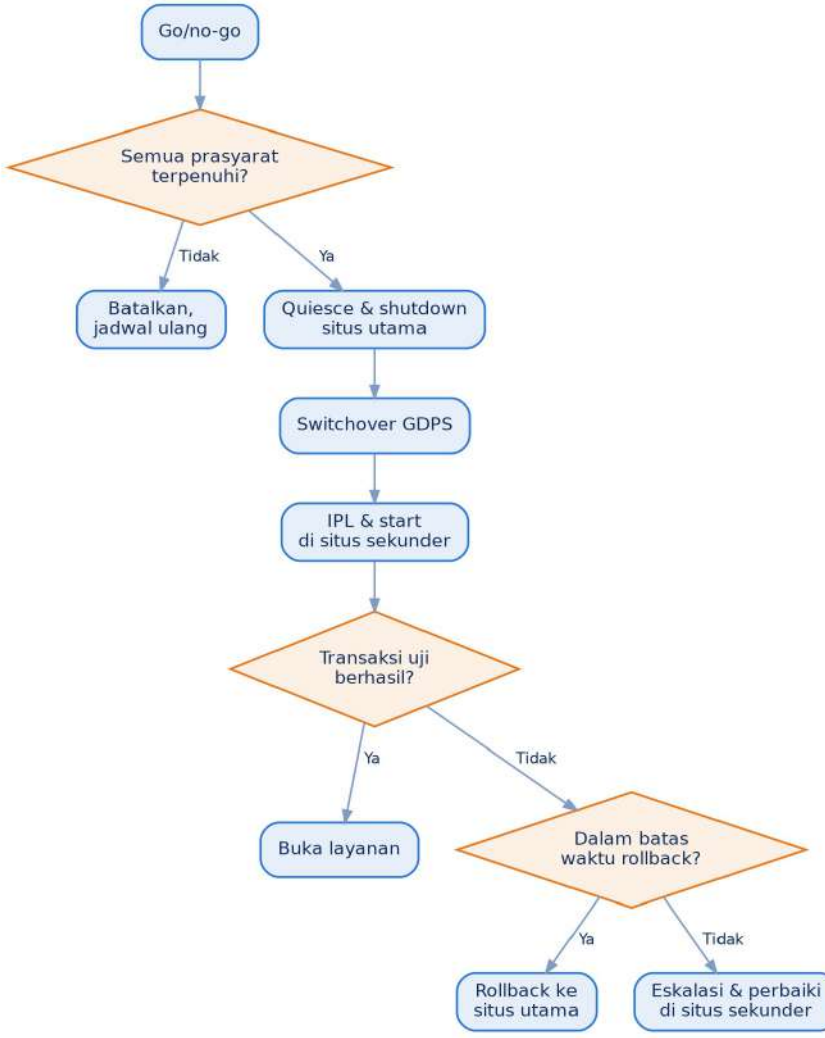
Bab 44 dan 46 membahas arsitektur dan latihan DR. Bagian ini adalah prosedurnya: langkah berurutan, pemilik tiap langkah, dan titik keputusan, dalam bentuk yang bisa dijalankan pada pukul tiga pagi oleh orang yang sedang panik.

## Bab 115. Switchover Terencana

Switchover terencana memindahkan produksi ke situs sekunder secara sengaja, misalnya untuk pemeliharaan situs utama atau uji DR. Karena direncanakan, tidak boleh ada data yang hilang (RPO nol).

| No | Langkah | Pemilik | Target durasi |
|---|---|---|---|
| 1 | Go/no-go: EOD selesai, GL seimbang, replikasi sinkron sehat | Koordinator DR | 15 menit |
| 2 | Umumkan jendela pemeliharaan ke kanal dan cabang | Operasi bisnis | Sebelum mulai |
| 3 | Hentikan penerimaan transaksi baru; alihkan kanal ke mode maintenance | Jaringan, aplikasi | 10 menit |
| 4 | Hentikan CICS, MQ, Db2 dengan urutan standar (Bab 12.4) | Sysprog, DBA | 20 menit |
| 5 | Shutdown z/OS dan keluarkan sistem dari sysplex | Sysprog | 10 menit |
| 6 | Jalankan skrip switchover GDPS: balik arah replikasi | Storage | 10 menit |

| No | Langkah | Pemilik | Target durasi |
|---|---|---|---|
| 7 | Aktifkan kapasitas DR (CBU) bila diperlukan; IPL sistem di situs sekunder | Sysprog | 25 menit |
| 8 | Start subsistem; jalankan health check dan transaksi uji | Sysprog, aplikasi | 20 menit |
| 9 | Arahkan DNS, DVIPA, dan koneksi eksternal ke situs sekunder | Jaringan | 10 menit |
| 10 | Go-live dan pemantauan intensif satu jam pertama | Semua | 60 menit |




Tetapkan batas waktu rollback sebelum mulai. Setelah batas itu lewat, jalan kembali ke situs utama sering lebih berisiko daripada memperbaiki masalah di situs sekunder.

## 115.1 Analisis: Titik Keputusan dalam Switchover Terencana

Switchover terencana memberi kemewahan yang tidak dimiliki failover: waktu untuk berhenti dan memeriksa di setiap tahap. Kemewahan itu

hanya berguna jika titik-titik pemeriksaan ditetapkan sebelumnya, lengkap dengan kriteria lanjut dan jalan kembalinya.

| Titik keputusan | Kriteria lanjut | Jalan kembali bila gagal |
|---|---|---|
| Sebelum menghentikan kanal | Semua pasangan replikasi duplex; tim lengkap; tidak ada insiden terbuka | Tunda tanpa dampak |
| Setelah aplikasi berhenti | Tidak ada transaksi menggantung; tabel kontrol bersih | Start kembali di situs utama |
| Setelah swap storage | Status replikasi di situs DR sesuai harapan | Swap kembali ke situs utama |
| Setelah IPL di situs DR | Semua sistem aktif; sysplex utuh | IPL kembali di situs utama |
| Setelah start subsistem | Db2, CICS, dan MQ sehat; tidak ada objek restricted | Hentikan dan kembali ke situs utama |
| Setelah validasi | Total kontrol cocok; transaksi uji berhasil | Kembali ke situs utama sebelum kanal dibuka |
| Buka kanal | Semua kriteria di atas terpenuhi | Setelah kanal dibuka, kembali ke situs utama adalah switchover baru |

Baris terakhir adalah titik tanpa kembali yang sebenarnya. Selama kanal belum dibuka, belum ada transaksi nasabah di situs DR, dan kembali ke situs utama masih relatif sederhana. Setelah kanal dibuka, data baru tercipta di situs DR, dan kembali ke situs utama harus dilakukan sebagai switchover terencana tersendiri.

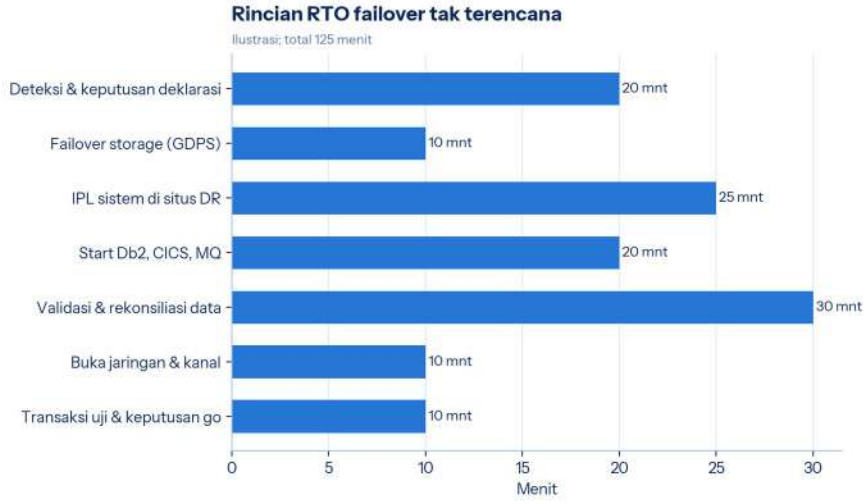
Komunikasi sama pentingnya dengan teknis. Tentukan siapa yang mengumumkan setiap titik keputusan, lewat saluran apa, dan siapa yang berhak berkata “berhenti”. Dalam switchover yang panjang, catatan waktu setiap titik juga menjadi bahan analisis linimasa di Bab 116.1.

# Bab 116. Failover Tak Terencana

Failover tak terencana dimulai dari keputusan, bukan dari perintah teknis. Kriteria deklarasi bencana dan siapa yang berwenang memutuskannya harus ditetapkan dan disetujui manajemen jauh hari.

| Kriteria deklarasi | Contoh |
|---|---|
| Situs utama tidak dapat diakses | Kebakaran, banjir, putus listrik total berkepanjangan |
| Kegagalan storage yang tak dapat dipulihkan cepat | Kerusakan array yang tidak tertangani HyperSwap |
| Estimasi pemulihan di situs utama melebihi RTO | Perbaikan hardware butuh waktu lebih lama dari target |

Rincian RTO di bawah menunjukkan pola yang hampir selalu muncul dalam latihan: langkah teknis relatif cepat, sedangkan keputusan dan validasi data memakan porsi terbesar.



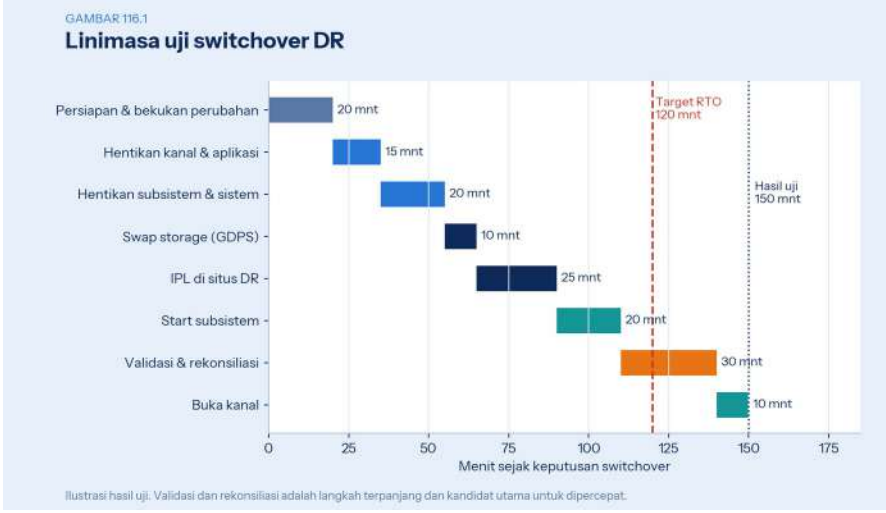

*Ilustrasi · data rekaan*

Mengurangi RTO karena itu lebih efektif lewat kejelasan wewenang deklarasi dan otomasi validasi, bukan hanya lewat teknologi replikasi yang lebih cepat.

```
ALGORITMA FailoverTakTerencana()
  catat waktu kejadian T0
  kumpulkan fakta: apa yang mati, sejak kapan, perkiraan
pemulihan
  jika kriteria deklarasi terpenuhi dan pejabat berwenang
menyetujui:
      catat waktu deklarasi
      jalankan skrip failover GDPS (kunci konsistensi di
situs sekunder)
      aktifkan CBU; IPL sistem DR dengan LOADPARM tercetak
      start Db2 (restart normal, periksa objek LPL/GRECP),
MQ, CICS
      tentukan transaksi terakhir yang ada di situs
sekunder (RPO nyata)
      validasi: GL seimbang untuk tanggal berjalan,
rekonsiliasi dengan jaringan pembayaran
      buka layanan bertahap: internal -> teller -> kanal
digital
  selain itu: lanjutkan pemulihan di situs utama, evaluasi
ulang tiap 30 menit
```

## 116.1 Analisis: Memendekkan Waktu Switchover

Uji switchover DR hampir selalu menghasilkan angka yang lebih buruk dari target di dokumen. Itu bukan kegagalan, tetapi justru tujuan uji: menemukan langkah mana yang memakan waktu dan bagaimana memendekkannya. Analisis dimulai dengan mencatat waktu mulai dan selesai setiap langkah, lalu menggambarkannya sebagai linimasa.




*Gambar 116.1 — Linimasa uji switchover DR*

Dalam ilustrasi, total waktu 150 menit, sedangkan target RTO 120 menit. Selisihnya 30 menit, dan langkah terpanjang adalah validasi dan rekonsiliasi, yaitu 30 menit.

| Langkah | Peluang percepatan | Perkiraan hemat |
|---|---|---|
| Persiapan (20 menit) | Checklist yang sudah terisi sebelum keputusan diambil | 10 menit |
| Hentikan subsistem dan sistem (20 menit) | Otomasi stop dengan urutan dependensi (Bab 9.1) | 5–10 menit |
| IPL di situs DR (25 menit) | System Recovery Boost; CLPA hanya bila perlu | 5 menit |
| Validasi dan rekonsiliasi (30 menit) | Skrip total kontrol otomatis (Bab 61.1); sebagian berjalan paralel dengan start subsistem | 15 menit |

Jika semua peluang itu terwujud, total waktu turun sekitar 35–40 menit, menjadi sekitar 110–115 menit, di bawah target. Perbaikan terbesar datang dari validasi, bukan dari langkah teknis yang terlihat paling rumit seperti swap storage dan IPL.

Simpan linimasa setiap uji switchover. Membandingkan linimasa dari tahun ke tahun menunjukkan apakah perbaikan benar-benar terjadi, dan

langkah mana yang justru memburuk, misalnya karena jumlah subsistem bertambah atau data semakin besar. Laporkan angka nyata, bukan target, kepada manajemen dan pemeriksa.

## Bab 117. Dokumen Pemulihan yang Harus Dicetak

Saat bencana, wiki internal, email, dan sistem tiket mungkin ikut mati. Simpan salinan cetak terbaru di situs DR dan di tangan koordinator DR, dan perbarui setiap kali ada perubahan.

- ☐ Daftar kontak darurat: tim internal, pejabat berwenang, IBM, vendor aplikasi, penyedia jaringan dan data center.
- ☐ Alamat IPL, LOADPARM, dan nama LPAR untuk setiap sistem di situs DR.
- ☐ Prosedur aktivasi CBU dan nomor kontrak layanan.
- ☐ Prosedur akses HMC darurat (kredensial disimpan dalam amplop tersegel sesuai kebijakan keamanan).
- ☐ Diagram jaringan situs DR, alamat DVIPA, dan perubahan DNS yang diperlukan.
- ☐ Urutan start subsistem dan dependensinya.
- ☐ Checklist switchover (Bab 115) dan failover (Bab 116).
- ☐ Daftar runbook prioritas: RB-006, RB-009, RB-011, RB-019, RB-030, RB-051.
- ☐ Formulir log kejadian untuk mencatat waktu, keputusan, dan pelaksana setiap langkah.

# 📖 Bagian XXVIII — Anatomi Insiden

Empat narasi berikut adalah fiksi yang disusun dari pola insiden yang umum di lingkungan mainframe perbankan. Nama, angka, dan detail teknis sengaja dibuat generik. Nilainya ada pada urutan keputusan: apa yang dilihat, apa yang diputuskan, dan kontrol mana yang akhirnya menyelamatkan atau gagal menyelamatkan.

## Bab 118. Narasi Pertama: Malam Ketika Akrual Hampir Berjalan Dua Kali

Pukul 22.41, step akrual bunga tabungan abend `S0C7`. Operator jaga, Dimas, melihat job merah di scheduler dan menjalankan runbook yang ia hafal: perbaiki record, lalu restart.

Masalahnya, ia me-restart job dari step pertama, bukan dari step yang gagal. Sekitar 60% rekening sudah menerima akrual sebelum abend, dan kini akan menerima akrual kedua.


*Gambar 118.1 — Garis waktu malam akrual hampir berjalan dua kali*

Yang menyelamatkan malam itu adalah satu step verifikasi kecil. Setelah fase akrual, program pembanding menghitung total akrual dan membandingkannya dengan proyeksi dari saldo kemarin. Pukul 23.12 selisihnya hampir dua kali lipat, dan job verifikasi berakhir RC 8, menahan fase GL.

Sysprog jaga, Rina, dibangunkan. Ia tidak langsung menjalankan ulang apa pun. Ia membaca tabel kontrol dan menemukan `RUN_ID` fase akrual bernilai 2, lalu memastikan belum ada jurnal yang masuk ke GL.

Pemulihannya sederhana karena fase akrual dirancang hapus-lalu-sisip per tanggal bisnis. Hasil akrual tanggal itu dihapus, fase dijalankan ulang sekali, dan total cocok pukul 01.02. Layanan dibuka tepat waktu.

| Apa yang berjalan baik | Apa yang harus diperbaiki |
|---|---|
| Verifikasi total kontrol terhadap proyeksi | Runbook S0C7 tidak menyebut titik restart yang benar |
| Fase akrual idempoten per tanggal | Scheduler mengizinkan restart dari awal tanpa peringatan |
| Sysprog membaca dulu sebelum bertindak | Fase belum memakai klaim atomik di tabel kontrol |

Perbaikan permanen: klaim fase atomik (Bab 105) dan revisi runbook dengan langkah “periksa tabel kontrol sebelum restart”.

## Bab 119. Narasi Kedua: Cabang yang Lambat Selama Tiga Pekan

Selama tiga pekan, sejumlah cabang di satu wilayah mengeluhkan layar teller lambat pada jam sibuk. Tim jaringan memeriksa link WAN dan tidak menemukan masalah; tim aplikasi memeriksa CICS dan melihat waktu respons rata-rata yang normal.

Petunjuknya justru ada di kata “rata-rata”. Ketika sysprog memecah waktu respons per kode transaksi, sebagian kecil transaksi ternyata lambat, dan semuanya berawalan `TN`.

Prefix TN diperkenalkan oleh versi baru aplikasi teller yang baru dipasang di wilayah itu. Aturan klasifikasi WLM hanya mengenal prefix lama, sehingga transaksi baru jatuh ke service class default dengan importance rendah.

Di jam biasa, perbedaan itu tidak terasa. Di jam sibuk, ketika CPU mendekati penuh, WLM dengan benar mendahulukan pekerjaan yang lebih penting, dan transaksi teller baru menunggu di belakang laporan ad hoc.

Perbaikannya satu baris aturan klasifikasi WLM dan aktivasi ulang policy. Waktu respons cabang kembali normal dalam hitungan menit.

Pelajarannya: pemantauan rata-rata menyembunyikan kelompok kecil yang menderita. Laporan per report class dan per prefix transaksi harus menjadi bagian review kinerja rutin, dan setiap rilis aplikasi yang menambah prefix transaksi wajib menyertakan perubahan WLM.

## 119.1 Analisis: Mengapa Butuh Tiga Pekan untuk Terdeteksi

Narasi kedua memperlihatkan masalah yang tidak memicu satu pun alert. Transaksi dari satu wilayah salah diklasifikasikan ke service class berprioritas rendah. Mainframe secara keseluruhan sehat, CPU tidak penuh, dan tidak ada abend. Yang ada hanya teller di satu wilayah yang menunggu lebih lama, terutama di jam sibuk.

Masalah seperti ini lolos karena pemantauan hanya melihat angka agregat. Rata-rata waktu respons seluruh bank tetap baik, karena wilayah yang terdampak hanya sebagian kecil dari total transaksi. Deteksi yang lebih baik memecah metrik per dimensi yang bermakna bagi bisnis:

| Dimensi | Contoh pemecahan | Masalah yang tertangkap |
|---|---|---|
| Wilayah atau cabang | Waktu respons per kode wilayah | Klasifikasi WLM, jaringan cabang |
| Kanal | Teller, mobile, ATM, API mitra | Masalah di gateway atau adapter tertentu |
| Jenis transaksi | Inquiry, transfer, pembayaran | Access path yang rusak untuk satu jenis transaksi |

| Dimensi | Contoh pemecahan | Masalah yang tertangkap |
|---|---|---|
| Service class | PI per service class per interval | Tujuan WLM yang meleset untuk satu kelompok |

Aturan deteksi yang sederhana sudah cukup untuk menangkap kasus ini: jika PI sebuah service class di atas 1 selama tiga interval berturut-turut pada jam kerja, kirim peringatan. Dengan interval 15 menit, masalah akan terdeteksi dalam 45 menit sejak pertama kali muncul di jam sibuk, bukan tiga pekan.

$$\text{Alert} \Leftrightarrow PI_{k,t} > 1 \text{ untuk } t = t_0, t_0 + 1, t_0 + 2$$

Pelajaran yang lebih luas: sistem yang sehat secara teknis belum tentu memberi layanan yang baik bagi setiap nasabah. Pemantauan yang baik mengukur pengalaman pengguna per kelompok yang bermakna, dan keluhan dari cabang diperlakukan sebagai data pemantauan, bukan sekadar tiket helpdesk.

## Bab 120. Narasi Ketiga: Backup yang Enam Bulan Tidak Lengkap

Setiap malam, job backup produksi selesai dengan RC 0. Enam bulan berturut-turut, laporan harian menunjukkan warna hijau.

Dalam uji restore acak triwulanan, tim diminta memulihkan satu dataset dari modul pembiayaan baru. Dataset itu tidak ditemukan di backup mana pun.

Penyebabnya ada di filter job DFSMSdss. Filter hanya mencakup `BANK.PROD.**`, sedangkan modul pembiayaan baru dibuat enam bulan sebelumnya dengan HLQ `BNKF`. Tidak ada yang salah dari sudut pandang job: ia mencadangkan persis apa yang diminta.

Tidak ada data yang hilang, karena insiden ditemukan lewat uji, bukan lewat bencana. Namun selama enam bulan, bank beroperasi dengan asumsi perlindungan yang tidak benar.

| Kontrol yang ada | Kenapa gagal mendeteksi |
|---|---|
| RC job backup | RC hanya membuktikan job berjalan, bukan cakupannya benar |
| Laporan hijau harian | Mengukur keberhasilan, bukan kelengkapan |
| Uji restore triwulanan | Berhasil menemukan masalah, tetapi terlambat enam bulan |

Perbaikan permanen: laporan cakupan backup harian yang membandingkan dataset produksi di katalog dengan dataset yang benar-benar dicadangkan, serta langkah wajib “daftarkan ke backup” di checklist pembuatan HLQ baru.

## Bab 121. Narasi Keempat: Uji Pemulihan yang Gagal Sebagian

Uji DR tahunan berjalan mulus di awal. Switchover GDPS selesai, IPL di situs sekunder berhasil, dan Db2 start tanpa masalah.

Kemudian region CICS gagal membuka sebagian besar file VSAM. Pesan menunjukkan SMSVSAM tidak dapat terhubung ke lock structure RLS.

Tim menemukan bahwa CFRM policy di situs sekunder adalah versi lama. Dua tahun sebelumnya, structure RLS dan satu group buffer pool Db2 ditambahkan ke policy di situs utama, tetapi policy DR tidak ikut diperbarui.

Uji dihentikan pada titik itu dan dinyatakan gagal sebagian. Setelah policy disinkronkan, uji ulang sebulan kemudian berhasil penuh.

Pelajarannya adalah *configuration drift*: dua situs yang seharusnya identik perlahan berbeda karena perubahan hanya diterapkan di tempat yang sedang dipakai. Solusinya adalah memperlakukan konfigurasi DR sebagai bagian dari baseline yang dibandingkan otomatis (Bab 15), bukan dokumen yang diperbarui setahun sekali menjelang uji.

## 121.1 Analisis: Cakupan Uji DR

Narasi keempat menunjukkan uji DR yang “berhasil” untuk infrastruktur tetapi gagal di tingkat aplikasi: sistem menyala, subsistem aktif, tetapi CICS gagal membuka file karena konfigurasi DR tertinggal. Uji DR yang hanya membuktikan bahwa mesin menyala memberi rasa aman yang palsu.

Cakupan uji DR bisa diukur:

$$C_{DR} = \frac{\text{fungsi bisnis kritikal yang terbukti berjalan di DR}}{\text{total fungsi bisnis kritikal}}$$

| Tingkat uji | Yang dibuktikan | Frekuensi yang disarankan |
|---|---|---|
| Komponen | Replikasi, IPL di DR, start subsistem | Setiap kuartal |
| Aplikasi | Transaksi uji per aplikasi berhasil di DR | Setiap semester |
| Proses bisnis | EOD lengkap dijalankan di DR, laporan sesuai | Setahun sekali |
| Switchover penuh | Layanan nasabah berjalan dari DR dalam periode tertentu | Setahun sekali, sesuai ketentuan |

```
ALGORITMA RencanaUjiDRTahunan(daftar_fungsi_kritikal)
  untuk setiap fungsi f:
      tentukan transaksi uji dan hasil yang diharapkan
      tandai tanggal terakhir f terbukti berjalan di DR
  prioritas <- fungsi yang belum terbukti dalam 12 bulan,
atau berubah besar sejak uji terakhir
  jadwalkan uji tingkat aplikasi untuk fungsi prioritas
setiap semester
  jadwalkan satu uji proses bisnis dan satu switchover
penuh setiap tahun
  laporkan C_DR setiap kuartal kepada manajemen
```

Fungsi yang berubah besar, misalnya aplikasi baru, integrasi baru, atau migrasi database, harus diuji di DR segera setelah masuk produksi, bukan menunggu uji tahunan berikutnya. Perubahan besar adalah saat konfigurasi DR paling mungkin tertinggal. Deteksi drift harian (Bab 15.1)

menangkap sebagian masalah ini, tetapi hanya uji nyata yang membuktikan fungsinya berjalan.

## Bab 122. Benang Merah Empat Narasi

| Pola | Narasi | Kontrol pencegah |
|---|---|---|
| Tanda hijau yang menyesatkan | Ketiga (RC 0), kedua (rata-rata normal) | Ukur hasil yang sebenarnya: cakupan, persentil, per kelompok |
| Perubahan di satu tempat, lupa di tempat lain | Kedua (WLM), keempat (CFRM DR) | Checklist perubahan yang mencakup semua konfigurasi terkait |
| Verifikasi independen menyelamatkan | Pertama (total kontrol) | Setiap fase kritikal punya pembanding yang dihitung terpisah |
| Uji menemukan apa yang tidak ditemukan pemantauan | Ketiga, keempat | Uji restore dan DR rutin dengan skenario yang dirotasi |
| Tenang dan membaca dulu | Pertama | Budaya “diagnosis sebelum tindakan” di runbook |

Hampir tidak ada insiden besar yang disebabkan satu kesalahan tunggal. Yang ada adalah satu kesalahan yang lolos dari beberapa kontrol yang semuanya tampak berfungsi.

# 📅 Bagian XXIX — Setahun dalam Hidup System Engineer

Pekerjaan sysprog mainframe punya musim. Memahami musim itu membantu merencanakan perubahan besar di waktu yang aman dan menghindarinya di waktu yang berbahaya.


*Rencana contoh · sesuaikan dengan kalender bank Anda*

Pola di atas adalah contoh. Prinsip yang umum berlaku: perubahan besar di Maret dan September, uji DR jauh dari akhir tahun, dan change freeze menjelang serta selama penutupan tahun.

## Bab 123. Kuartal Pertama

Kuartal pertama dimulai dengan sisa beban akhir tahun: laporan tahunan, audit eksternal, dan arsip data tahun sebelumnya. Setelah itu, ini adalah jendela terbaik untuk pemeliharaan besar pertama.

| Bulan | Agenda utama |
|---|---|
| Januari | Stabilisasi pasca-EOY, arsip data tahunan, pembukaan kembali perubahan setelah freeze |
| Februari | Dukungan audit, review akses tahunan, rencana pelatihan tim |

| Bulan | Agenda utama |
|---|---|
| Maret | Instalasi RSU atau upgrade z/OS dengan rolling IPL, review kapasitas kuartal |

## Bab 124. Kuartal Kedua

Kuartal kedua biasanya paling tenang dari sisi bisnis, sehingga cocok untuk uji DR besar dan proyek teknis.

| Bulan | Agenda utama |
|---|---|
| April | Tindak lanjut temuan audit, uji restore acak |
| Mei | Uji DR penuh termasuk EOD di situs sekunder |
| Juni | Laporan pasca-uji DR, review akses semester, penutupan semester pertama |

## Bab 125. Kuartal Ketiga

Kuartal ketiga adalah waktu untuk pemeliharaan besar kedua dan penyusunan anggaran tahun berikutnya.

| Bulan | Agenda utama |
|---|---|
| Juli | Pelatihan dan rotasi peran, uji Cyber Vault |
| Agustus | Proyeksi kapasitas tahun depan, persiapan RSU |
| September | Instalasi RSU kedua, review kontrak dan lisensi |

## Bab 126. Kuartal Keempat

Kuartal keempat adalah musim risiko tertinggi. Volume transaksi naik menjelang akhir tahun, dan setiap kesalahan berdampak pada penutupan buku.

| Bulan | Agenda utama |
|---|---|
| Oktober | Finalisasi anggaran dan capacity plan, |

| Bulan | Agenda utama |
|---|---|
| | knowledge transfer sebelum freeze |
| November | Uji DR kedua atau uji switchover parsial, gladi EOY di UAT |
| Desember | Change freeze, kesiapan EOY, jadwal jaga diperketat |

```
ALGORITMA KesiapanEOY()
  H-60 hari: gladi EOY di UAT dengan data produksi yang
di-mask
  H-30 hari: mulai change freeze kecuali perbaikan darurat
  H-14 hari: verifikasi kapasitas (spool, storage group,
log Db2, zFS)
  H-7 hari : cetak ulang dokumen DR, konfirmasi jadwal
jaga dan kontak vendor
  H-1 hari : backup tambahan dan snapshot Safeguarded Copy
sebelum EOY
  Hari H  : tim inti hadir, pemantauan proyeksi EOD setiap
15 menit
  H+1     : validasi penutupan laba rugi, arsip, laporan
pasca-EOY
```

## 126.1 Analisis: Mengapa Change Freeze Akhir Tahun Masuk Akal

Change freeze di akhir tahun sering dikeluhkan tim pengembang, karena terasa menghambat. Keputusan ini sebenarnya bisa dijelaskan secara kuantitatif. Jika setiap perubahan punya peluang kecil untuk memicu insiden, peluang setidaknya satu insiden naik cepat seiring jumlah perubahan:

$$P(\text{minimal satu insiden}) = 1 - (1 - p)^n$$

| Peluang insiden per perubahan (p) | Jumlah perubahan (n) | Peluang minimal satu insiden |
|---|---|---|
| 1% | 10 | sekitar 9,6% |
| 1% | 40 | sekitar 33% |
| 2% | 40 | sekitar 55% |

Desember adalah bulan dengan dampak insiden terbesar: EOM, EOY, laporan tahunan, dan puncak transaksi akhir tahun terjadi bersamaan. Mengurangi perubahan dari 40 menjadi 10 menurunkan peluang insiden dari sekitar sepertiga menjadi kurang dari sepersepuluh, tepat pada periode ketika insiden paling mahal.

Freeze yang baik tidak berarti tidak ada perubahan sama sekali.

| Jenis perubahan | Selama freeze |
|---|---|
| Perbaikan insiden produksi | Diizinkan dengan persetujuan darurat |
| Patch keamanan kritikal | Diizinkan dengan penilaian risiko |
| Perubahan regulasi dengan tenggat | Diizinkan, direncanakan jauh sebelumnya |
| Fitur baru dan perubahan kosmetik | Ditunda ke Januari |
| Perubahan infrastruktur rutin | Ditunda, kecuali untuk mencegah insiden |

Komunikasikan jadwal freeze sejak kuartal ketiga (Bab 125), agar tim pengembang dapat merencanakan rilis sebelum freeze dimulai. Setelah freeze berakhir, hindari “banjir perubahan” di pekan pertama Januari dengan menyebar rilis yang tertunda secara bertahap. Banjir perubahan kembali menaikkan n dan menghapus manfaat freeze itu sendiri.

## Bab 127. Irama Mingguan dan Harian

Irama kecil yang konsisten mencegah insiden besar. Checklist di Bab 47 adalah isinya; bab ini mengatur kapan dijalankan.

| Waktu | Kegiatan |
|---|---|
| Setiap pagi, 07.30 | Baca hasil health check (Bab 86), status EOD semalam, dan catatan jaga |
| Setiap pagi, 08.00 | Rapat singkat 15 menit: insiden semalam, perubahan hari ini, risiko |
| Siang hari | Pekerjaan terencana: perubahan, proyek, analisis kinerja |
| Sore, 16.00 | Verifikasi kesiapan EOD: kapasitas, perubahan yang masuk hari ini |

| Waktu | Kegiatan |
|---|---|
| Senin | Review perubahan pekan ini dan HOLDDATA baru |
| Rabu | Sesi berbagi pengetahuan 30 menit untuk tim |
| Jumat | Review tren kapasitas dan kinerja, persiapan jaga akhir pekan |

Irama ini terdengar birokratis, tetapi justru membebaskan. Ketika pemeriksaan rutin berjalan otomatis dan terjadwal, perhatian tim tersisa untuk masalah yang benar-benar butuh pemikiran.

# 🗂️ Bagian XXX — Manajemen Proyek Teknis

Proyek mainframe jarang gagal karena teknologinya. Mereka gagal karena prasyarat yang terlewat, pengujian yang dipangkas saat jadwal mepet, atau titik tanpa jalan kembali yang tidak disadari. Bagian ini membahas tiga proyek yang pasti dihadapi setiap tim: upgrade z/OS, migrasi CPC, dan cutover.

## Bab 128. Proyek Upgrade z/OS

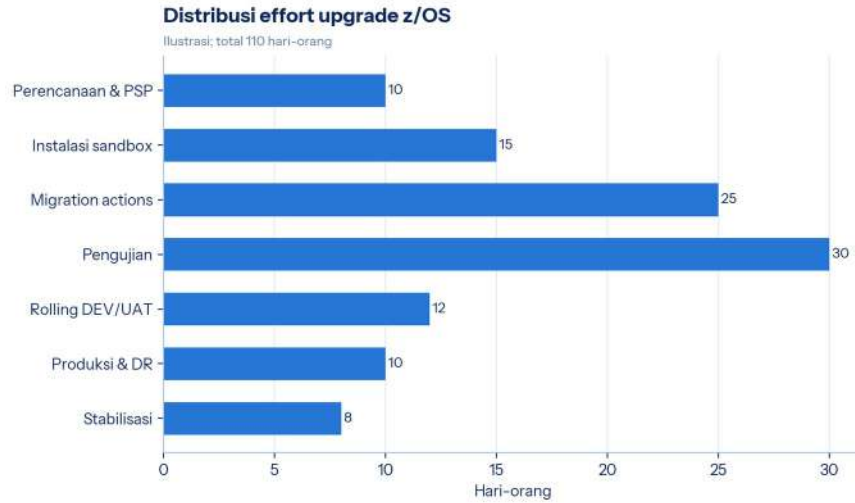
Upgrade rilis z/OS di sysplex dilakukan bergilir: satu sistem di-IPL dengan rilis baru sementara sistem lain tetap di rilis lama. Hal ini dimungkinkan karena IBM mendukung koeksistensi beberapa rilis dalam satu sysplex, dengan syarat PTF koeksistensi sudah terpasang di sistem yang lama.


*Gambar 128.1 — Rencana proyek upgrade z/OS (16 pekan)*

| Alat | Fungsi dalam proyek |
|---|---|
| z/OSMF Software Management | Menerima dan memasang paket rilis sebagai software instance |

| Alat | Fungsi dalam proyek |
|---|---|
| z/OSMF Workflows | Panduan langkah upgrade dan migration actions yang bisa dilacak |
| FIXCAT koeksistensi | Memastikan sistem lama siap hidup berdampingan dengan rilis baru |
| Health Checker (check migrasi) | Menemukan pengaturan yang harus diubah sebelum upgrade |
| PSP bucket | Informasi terbaru dan PTF wajib untuk rilis dan hardware |

Effort terbesar hampir selalu ada di migration actions dan pengujian, bukan di instalasi. Rencana yang memangkas pengujian untuk mengejar tanggal adalah rencana yang memindahkan risiko ke produksi.




*Ilustrasi · data rekaan*

## 128.1 Checklist sebelum IPL produksi pertama

- ☐ PTF koeksistensi terpasang di semua sistem yang masih memakai rilis lama.
- ☐ Semua check migrasi Health Checker bersih atau ada justifikasi tertulis.
- ☐ Produk ISV dan IBM terverifikasi kompatibel dengan rilis baru.

- ☐ SYSRES dan target library lama tetap tersedia untuk fallback.
- ☐ LOADPARM lama dan baru tercatat, prosedur IPL fallback sudah diuji di sandbox.
- ☐ Jadwal tidak bentrok dengan EOM, EOQ, atau periode freeze.

## 128.2 Analisis: Koeksistensi, Migrasi, dan Fallback

Upgrade z/OS dalam sysplex tidak dilakukan serentak. Sistem di-upgrade satu per satu lewat rolling IPL, sehingga selama beberapa pekan sistem dengan rilis lama dan baru berjalan berdampingan dalam satu sysplex. Masa ini aman hanya jika tiga hal dipahami: koeksistensi, migrasi, dan fallback.

| Konsep | Arti | Yang harus disiapkan |
|---|---|---|
| Koeksistensi | Rilis lama dan baru dapat berjalan bersama dalam satu sysplex dan berbagi data | PTF koeksistensi di sistem lama, dipasang sebelum sistem pertama di-upgrade |
| Migrasi | Tindakan yang wajib dilakukan saat pindah ke rilis baru | Daftar migration action dari dokumentasi IBM, dikerjakan dan dicentang satu per satu |
| Fallback | Kemampuan kembali ke rilis lama bila rilis baru bermasalah | SYSRES lama tetap utuh; PTF toleransi terpasang; tidak ada fungsi baru yang dipakai sebelum semua sistem di-upgrade |

Umumnya z/OS mendukung koeksistensi beberapa rilis berturut-turut dalam satu sysplex, dengan batas yang ditetapkan di dokumentasi migrasi setiap rilis. Aturan ini juga menentukan seberapa jauh sebuah sistem boleh tertinggal. Sistem yang terlalu lama tidak di-upgrade bisa memaksa upgrade dua langkah sekaligus, dengan risiko lebih besar.

Kesalahan paling berbahaya dalam masa koeksistensi adalah memakai fungsi baru terlalu cepat. Contohnya format dataset, level fungsi Db2, atau struktur couple dataset baru yang tidak dikenali rilis lama. Begitu fungsi baru dipakai, fallback menjadi tidak mungkin, karena sistem lama tidak dapat membaca data dalam format baru. Karena itu fungsi baru diaktifkan

sebagai langkah terpisah, setelah semua sistem stabil di rilis baru, dengan tiket perubahannya sendiri.

```
ALGORITMA RencanaRollingUpgrade(sistem_list)
  pasang PTF koeksistensi dan toleransi di semua sistem
lama; IPL bila perlu
  untuk setiap sistem s, mulai dari yang paling tidak
kritikal:
      IPL s dengan rilis baru; jalankan uji fungsi dan
kinerja
      pantau minimal satu siklus EOD penuh sebelum lanjut
ke sistem berikutnya
      jika ada masalah serius: IPL s kembali ke SYSRES
lama (fallback)
  setelah semua sistem stabil selama periode yang
disepakati:
      aktifkan fungsi baru satu per satu, masing-masing
dengan tiket perubahan
```

## Bab 129. Proyek Migrasi CPC

Migrasi ke generasi CPC baru mengubah fondasi fisik: prosesor, I/O, koneksi coupling, dan kriptografi. Proyek ini lebih banyak soal koordinasi dan urutan daripada soal perintah.

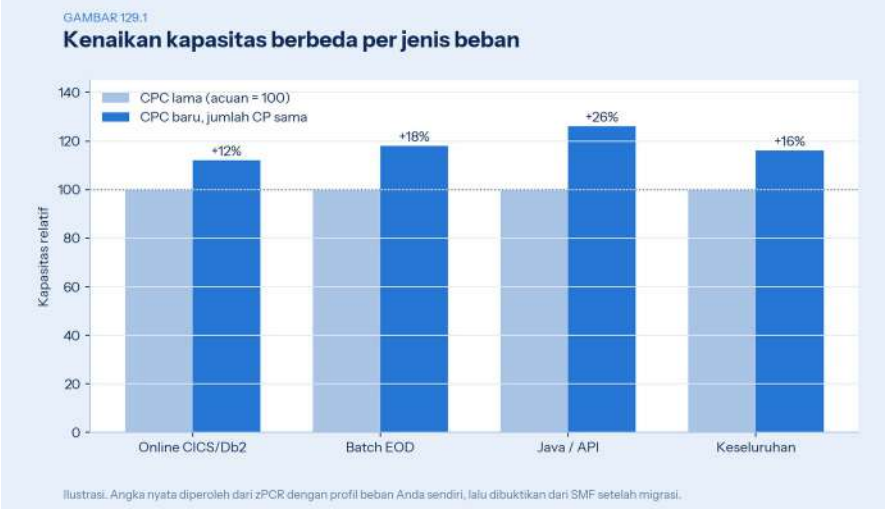
| Area | Pekerjaan utama | Risiko jika terlewat |
|---|---|---|
| Kapasitas | Sizing dengan alat perencanaan kapasitas IBM, pemetaan model kapasitas | Kapasitas kurang atau lisensi membengkak |
| I/O | IODF baru untuk tipe prosesor baru, pemetaan CHPID ke PCHID | Device tidak terlihat saat IPL |
| Coupling | Kompatibilitas tipe coupling link dengan CPC lama selama transisi | Sysplex terputus di tengah migrasi |
| Kriptografi | Migrasi master key dan konfigurasi Crypto Express | Transaksi PIN dan TLS gagal |
| STP | Peran server waktu di konfigurasi baru | Sistem kehilangan sinkronisasi waktu |
| Software | PTF minimum untuk | z/OS tidak mendukung fitur |

| Area | Pekerjaan utama | Risiko jika terlewat |
|---|---|---|
| | hardware baru lewat FIXCAT | atau gagal IPL |
| Lisensi | Kapasitas baru dan dampaknya ke MSU | Tagihan software tak terduga |

Pola yang aman adalah memindahkan LPAR satu per satu, dimulai dari sandbox dan non-produksi. CPC lama tetap siap selama beberapa pekan sebagai jalan kembali.

## 129.1 Analisis: Menyetarakan Kapasitas CPC Lama dan Baru

Pertanyaan terpenting dalam migrasi CPC adalah berapa kapasitas yang harus dibeli agar beban yang sama berjalan setidaknya sama baik, dan bagaimana tagihan software berubah. Jawabannya tidak sesederhana membandingkan jumlah prosesor atau frekuensi clock.




*Gambar 129.1 — Kenaikan kapasitas berbeda per jenis beban*

Kenaikan kapasitas generasi baru berbeda untuk setiap jenis beban. Beban yang sangat bergantung pada cache dan memori bisa mendapat manfaat berbeda dari beban komputasi murni. Karena itu IBM menyediakan alat perencanaan kapasitas, zPCR, yang memakai profil beban nyata dari SMF untuk memperkirakan kapasitas yang setara.

$$N_{\text{CP baru}} \approx N_{\text{CP lama}} \times \frac{\text{kapasitas per CP lama}}{\text{kapasitas per CP baru}} \times (1 + g)$$

g adalah pertumbuhan yang direncanakan selama umur mesin baru. Dalam ilustrasi, kenaikan keseluruhan sekitar 16% per CP. Jika pertumbuhan dua tahun ke depan diperkirakan 15%, jumlah CP yang sama hampir cukup, dengan cadangan yang tipis. Keputusan akhirnya memperhitungkan juga cadangan untuk puncak dan kebutuhan N+1 sysplex (Bab 11.1).

| Langkah | Tujuan |
|---|---|
| Kumpulkan SMF 70 dan 72 dari beberapa pekan, termasuk akhir bulan | Profil beban yang representatif |
| Jalankan zPCR untuk beberapa konfigurasi kandidat | Membandingkan pilihan secara objektif |
| Perhitungkan perubahan MSU dan dampaknya ke tagihan | Kapasitas yang lebih besar per CP bisa mengubah MSU per jam |
| Rencanakan zIIP dan IFL terpisah | Kapasitas khusus tidak dihitung ke MSU (Bab 42.2) |
| Ukur ulang setelah migrasi dari SMF | Membuktikan asumsi dan menyesuaikan defined capacity |

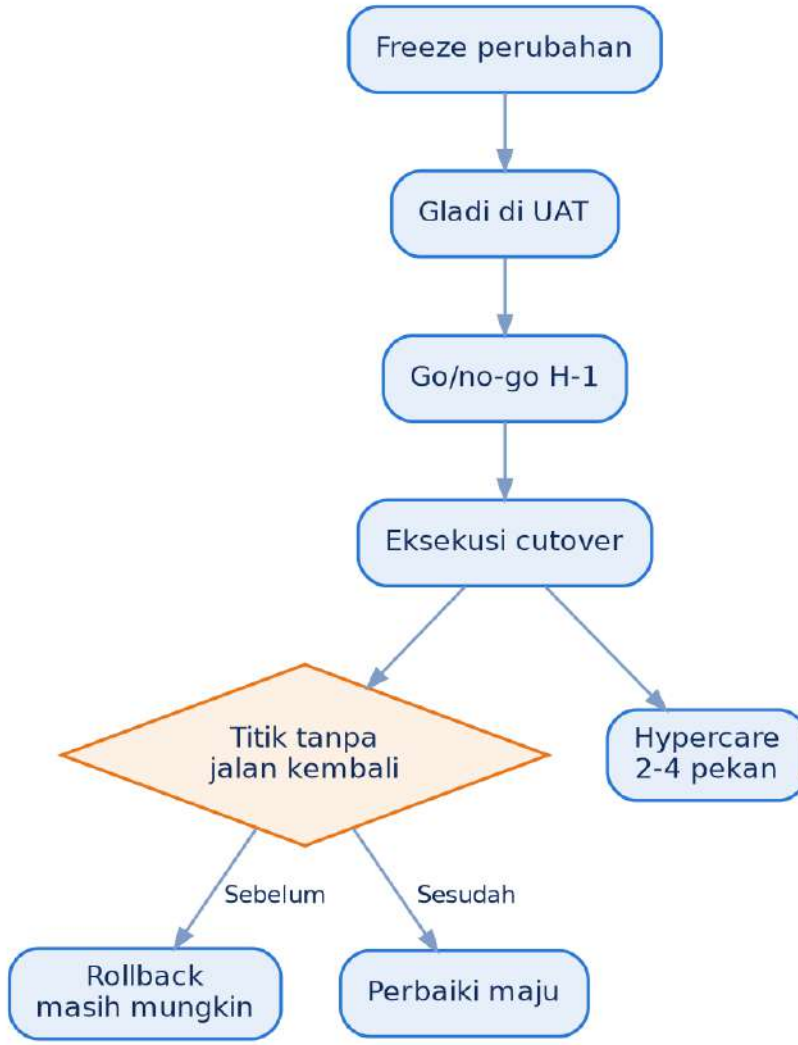
Perubahan generasi juga mengubah angka MSU untuk beban yang sama. Rencanakan bersama tim yang mengelola lisensi, dan tinjau laporan SCRT beberapa bulan pertama setelah migrasi dengan saksama (Bab 177).

## Bab 130. Cutover

Cutover adalah malam ketika rencana bertemu kenyataan. Runbook cutover ditulis per menit, dengan pemilik tiap langkah dan kriteria go/no-go yang tidak bisa ditawar.

| Elemen runbook cutover | Isi |
|---|---|
| Jam per langkah | Waktu mulai dan selesai yang direncanakan |
| Pemilik | Satu nama per langkah, bukan tim |
| Kriteria sukses | Bukti objektif: pesan, return code, hasil query |

| Elemen runbook cutover | Isi |
|---|---|
| Titik go/no-go | Siapa memutuskan dan berdasarkan apa |
| Titik tanpa jalan kembali | Langkah setelah mana rollback tidak lagi realistis |
| Rollback | Langkah pembalikan per tahap sebelum titik tanpa jalan kembali |
| Komunikasi | Kapan dan kepada siapa status dilaporkan |



Hypercare adalah periode pemantauan intensif setelah cutover, dengan tim proyek tetap siaga. Jangan membubarkan tim proyek pada malam cutover berhasil.

## 130.1 Analisis: Kriteria Go/No-Go yang Terukur

Keputusan go atau no-go dalam cutover sering diambil dalam suasana tertekan, larut malam, oleh orang-orang yang sudah bekerja berjam-jam. Keputusan seperti itu rentan bias: setelah berbulan-bulan persiapan, ada dorongan kuat untuk tetap maju. Kriteria yang terukur dan disepakati jauh sebelum malam cutover melindungi tim dari tekanan itu.

| Kriteria | Ambang go | Sumber data |
|---|---|---|
| Rekonsiliasi saldo dan jumlah rekening | Selisih nol | Laporan pembanding sistem lama dan baru |
| Transaksi uji wajib | 100% berhasil | Skrip uji terstandar |
| Waktu respons transaksi utama | Persentil 90 dalam batas SLA | Uji beban singkat dan statistik |
| Tingkat error | Di bawah ambang yang disepakati | Log aplikasi dan gateway |
| Kesiapan rollback | Prosedur sudah diuji, durasi diketahui | Hasil gladi |
| Batas waktu | Keputusan diambil sebelum titik tanpa kembali | Jadwal cutover |

```
ALGORITMA KeputusanGoNoGo(hasil)
  jika ada kriteria wajib yang gagal maka NO-GO
  jika waktu sekarang melewati batas keputusan maka NO-GO
  jika ada kriteria tidak wajib yang gagal:
      jika dampaknya dipahami, mitigasinya tersedia, dan
disetujui pemilik risiko maka GO bersyarat
      selain itu NO-GO
  selain itu GO
  catat keputusan, alasan, dan siapa yang memutuskan
```

Batas waktu keputusan sama pentingnya dengan kriteria teknis. Rollback juga membutuhkan waktu. Jika keputusan diambil terlalu larut, pilihan rollback menjadi tidak mungkin karena layanan harus dibuka pagi hari.

Tentukan titik tanpa kembali dengan menghitung mundur dari jam pembukaan layanan dikurangi durasi rollback yang sudah diuji.

NO-GO bukan kegagalan. Keputusan no-go yang diambil dengan data yang jelas adalah keberhasilan proses. Proyek bisa dijadwalkan ulang, sedangkan kerusakan data dan kepercayaan nasabah jauh lebih sulit dipulihkan.

## Bab 131. Risiko dan Rollback

Setiap proyek harus punya daftar risiko yang dinilai dari kemungkinan dan dampaknya. Risiko di pojok kanan atas matriks harus punya mitigasi yang dijalankan sebelum eksekusi, bukan rencana darurat saja.


*Contoh penilaian*

| Perubahan | Bisa dibalik? | Cara membalik |
|---|---|---|
| Rilis z/OS di satu sistem | Ya | IPL dari SYSRES lama dengan LOADPARM lama |
| IODF baru | Ya | Aktifkan IODF sebelumnya |
| PTF yang belum di-ACCEPT | Ya | RESTORE SMP/E atau kembali ke SYSRES lama |
| Function level atau mode baru di Db2 dan subsistem | Sering sulit atau tidak bisa | Uji penuh sebelum aktivasi; perlakukan sebagai titik |

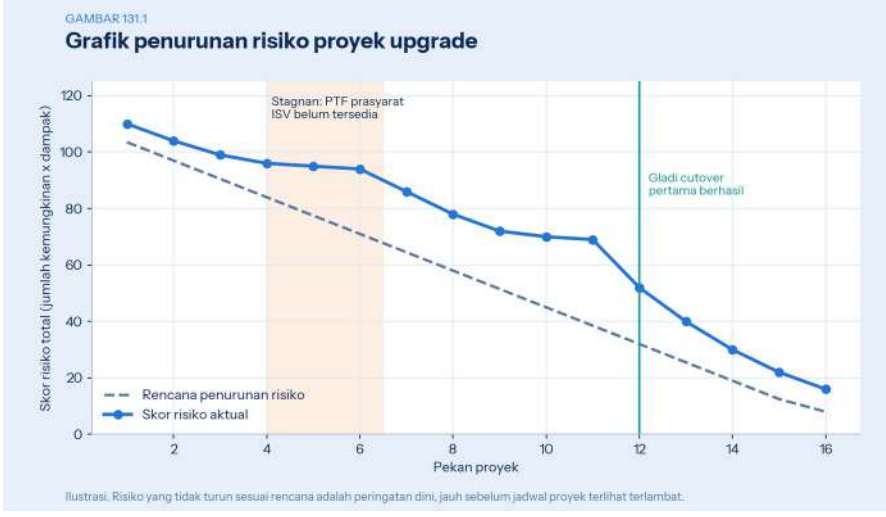
| Perubahan | Bisa dibalik? | Cara membalik |
|---|---|---|
| lain | | tanpa jalan kembali |
| Format data yang dikonversi | Tergantung | Simpan salinan data sebelum konversi |
| Migrasi master key kripto | Sulit | Ikuti prosedur resmi dan simpan key part dengan aman |

Prinsipnya: tunda langkah yang tidak bisa dibalik sampai langkah yang bisa dibalik terbukti stabil di produksi.

## 131.1 Analisis: Memantau Penurunan Risiko dan Menggladi Rollback

Matriks risiko di awal proyek sering dibuat lalu dilupakan. Padahal nilai terbesarnya muncul bila dipantau setiap pekan. Jumlahkan skor setiap risiko (kemungkinan × dampak) menjadi satu angka total, lalu gambarkan perkembangannya dari pekan ke pekan.

$$R_{\text{total}}(t) = \sum_{i} p_i(t) \times d_i(t)$$




*Gambar 131.1 — Grafik penurunan risiko proyek upgrade*

Dalam ilustrasi, skor risiko tidak turun selama pekan 4 sampai 6 karena PTF prasyarat dari vendor ISV belum tersedia. Jadwal proyek pada saat itu masih terlihat baik-baik saja. Grafik risiko memberi peringatan beberapa pekan lebih awal daripada laporan jadwal. Skor turun tajam setelah gladi cutover pertama berhasil, karena banyak risiko yang sebelumnya hanya dugaan kini sudah terbukti bisa ditangani.

**Gladi rollback.** Rencana rollback yang belum pernah dijalankan hanyalah dokumen. Gladi rollback di lingkungan uji menjawab tiga pertanyaan yang tidak bisa dijawab dari dokumen:

| Pertanyaan | Mengapa penting |
|---|---|
| Berapa lama rollback sebenarnya? | Menentukan titik tanpa kembali pada malam cutover (Bab 130.1) |
| Langkah mana yang tidak tertulis? | Setiap pertanyaan saat gladi adalah langkah yang hilang dari runbook |
| Apakah data kembali konsisten? | Total kontrol sebelum dan sesudah rollback harus sama |

Jadwalkan gladi rollback setelah gladi cutover, dengan tim yang sama yang akan bertugas pada malam sebenarnya. Catat durasi setiap langkah. Jika rollback ternyata membutuhkan 90 menit sementara rencana menyebut 30 menit, lebih baik mengetahuinya di pekan 12 daripada pukul tiga pagi pada malam cutover.

# 🧪 Bagian XXXI — Praktikum

Membaca tentang JCL tidak sama dengan menjalankan JCL yang gagal lalu memperbaikinya. Lima praktikum berikut dirancang agar bisa dijalankan dengan hak akses pengguna biasa di lingkungan belajar, tanpa menyentuh konfigurasi sistem.

## Bab 132. Menyiapkan Lingkungan Lab

| Lingkungan | Cocok untuk | Catatan |
|---|---|---|
| IBM Z Xplore | Belajar mandiri, pemula | Program belajar gratis dari IBM dengan akses z/OS jarak jauh |
| IBM Z Development and Test Environment (ZD&T) | Lab tim, pengujian aplikasi | Menjalankan z/OS di atas x86; berlisensi |
| Wazi as a Service | Pengembangan dan uji di cloud | Instance z/OS di IBM Cloud |
| LPAR sandbox perusahaan | Praktikum tingkat sysprog | Wajib terpisah dari sysplex produksi |

Aturan praktikum yang berlaku di semua lingkungan:

- Semua dataset memakai HLQ milik Anda sendiri (`&SYSUID`), tidak pernah HLQ aplikasi.
- Jangan menjalankan perintah console di sistem bersama tanpa izin pemiliknya.
- Catat setiap langkah dan hasilnya; catatan ini adalah bahan belajar paling berharga.

Setiap praktikum ditulis dengan pola: tujuan, langkah, hasil yang diharapkan, dan pertanyaan refleksi.

## Bab 133. Praktikum 1: Dataset, JCL, dan GDG

**Tujuan:** membuat GDG, menulis generasi baru dengan JCL, dan membaca hasilnya di SDSF.

**Langkah 1 — definisikan GDG (jalankan sekali):**

```
//LAB1A    JOB (LAB),'DEF
GDG',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//DEFGDG   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *,SYMBOLS=JCLONLY
  DEFINE GDG (NAME(&SYSUID..LAB.TRX) LIMIT(5) SCRATCH)
/*
```

**Langkah 2 — tulis generasi baru:**

```
//LAB1B    JOB (LAB),'TULIS
GDG',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//GEN      EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD *
0000000001 NASABAH A          0000150000
0000000002 NASABAH B          0000275000
0000000003 NASABAH C          00000A5000
/*
//SYSUT2   DD
DSN=&SYSUID..LAB.TRX(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(1,1)),RECFM=FB,LRECL=80
```

**Langkah 3:** jalankan job LAB1B tiga kali, lalu buka ISPF `=3.4` dengan pola `&SYSUID..LAB.TRX`. Buka output setiap job di SDSF panel `ST`.

**Hasil yang diharapkan:** tiga generasi `G0001V00` sampai `G0003V00`; setiap job selesai RC 0.

**Pertanyaan refleksi:**

1. Apa yang terjadi pada generasi paling lama setelah job dijalankan enam kali, dan kenapa?
2. Record ketiga berisi `A` di field nominal. Program apa yang akan abend saat membacanya sebagai packed atau zoned decimal, dan dengan kode apa?

## 133.1 Latihan Lanjutan: Batas GDG dan Kebutuhan Rerun

Kasus K-18 di Bab 188 berawal dari batas GDG yang terlalu kecil: saat EOD perlu dijalankan ulang untuk beberapa hari ke belakang, generasi yang dibutuhkan sudah terhapus. Latihan ini memperlihatkan perilaku itu secara langsung.

```
//DEFGDG   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE GENERATIONDATAGROUP -
         (NAME(HLQ.LAB.GDGTEST) LIMIT(3) NOEMPTY SCRATCH)
/*
//*------- jalankan step berikut lima kali, satu run per
"hari"
//BUAT     EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD *
DATA HARI KE-N
/*
//SYSUT2   DD
DSN=HLQ.LAB.GDGTEST(+1),DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(1,1)),RECFM=FB,LRECL=80
//*------- setelah run kelima
//LIHAT    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  LISTCAT ENTRIES(HLQ.LAB.GDGTEST) ALL
/*
```

**Yang diamati:**

1. Setelah lima run, `LISTCAT` hanya menunjukkan tiga generasi. Dua generasi tertua sudah dihapus karena opsi `SCRATCH`.
2. Ubah isi `SYSUT1` di setiap run, lalu buktikan bahwa `(0)` selalu merujuk generasi terbaru dan `(-2)` generasi tertua yang tersisa.
3. Ulangi dengan GDG baru yang memakai `EMPTY` alih-alih `NOEMPTY`, dan amati bahwa semua generasi lama dihapus sekaligus saat batas terlampaui.

**Pertanyaan refleksi:** EOD di bank Anda mungkin perlu dijalankan ulang hingga tiga hari ke belakang setelah libur panjang. Jika setiap hari kerja menghasilkan satu generasi, dan ada hari-hari dengan dua generasi karena rerun, berapa `LIMIT` minimum yang aman? Tuliskan alasannya, lalu bandingkan dengan nilai `LIMIT` GDG EOD di lingkungan kerja Anda.

## Bab 134. Praktikum 2: Backup dan Restore

**Tujuan:** mencadangkan dataset dengan DFSMSdss, menghapusnya, lalu memulihkannya.

```
//LAB2A    JOB
(LAB),'DUMP',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//DUMP     EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//OUT      DD
DSN=&SYSUID..LAB.BACKUP,DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(1,1)),UNIT=SYSALLDA
//SYSIN    DD *,SYMBOLS=JCLONLY
  DUMP DATASET(INCLUDE(&SYSUID..LAB.TRX.**))
OUTDDNAME(OUT)
/*
```

Setelah job sukses, hapus satu generasi lewat `=3.4` (perintah baris `DEL`). Lalu pulihkan:

```
//LAB2B    JOB
(LAB),'RESTORE',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//REST     EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//IN       DD DISP=SHR,DSN=&SYSUID..LAB.BACKUP
//SYSIN    DD *,SYMBOLS=JCLONLY
  RESTORE DATASET(INCLUDE(&SYSUID..LAB.TRX.**))
INDDNAME(IN) CATALOG
/*
```

**Hasil yang diharapkan:** generasi yang dihapus kembali muncul dan terkatalog; generasi yang masih ada dilewati dengan pesan bahwa dataset sudah ada.

**Pertanyaan refleksi:**

1. Apa bedanya menambahkan kata kunci REPLACE pada RESTORE? Kapan itu berbahaya?
2. Bagaimana Anda membuktikan backup ini lengkap, bukan sekadar RC 0? (Ingat Bab 120.)

## 134.1 Latihan Lanjutan: Membuktikan Restore dengan Total Kontrol

Restore yang “berhasil” menurut return code belum tentu mengembalikan data yang benar. Latihan ini menambahkan langkah pembuktian: data dipulihkan ke nama lain, lalu dibandingkan dengan aslinya memakai total kontrol (Bab 61.1).

```
//RESTORE  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DUMPIN   DD DISP=OLD,DSN=HLQ.LAB.BACKUP.DUMP
//SYSIN    DD *
  RESTORE DATASET(INCLUDE(HLQ.LAB.TRX.**)) -
          INDDNAME(DUMPIN) -
          RENAMEU((HLQ.LAB.TRX.**,HLQ.LABR.TRX.**)) -
          CATALOG
/*
//BANDING  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//ASLI     DD DISP=SHR,DSN=HLQ.LAB.TRX.HARIAN
//PULIH    DD DISP=SHR,DSN=HLQ.LABR.TRX.HARIAN
//TOOLIN   DD *
  COUNT FROM(ASLI)
  COUNT FROM(PULIH)
  STATS FROM(ASLI)  ON(30,8,PD)
  STATS FROM(PULIH) ON(30,8,PD)
/*
```

Sesuaikan posisi field nominal dengan layout file latihan Anda. RENAMEU memulihkan dataset dengan nama baru, sehingga data asli tidak tersentuh. Ini adalah praktik yang sama yang dianjurkan untuk pemulihan nyata saat ada keraguan (Bab 45.1).

**Yang harus dibuktikan:**

1. Jumlah record ASLI dan PULIH sama persis.

2. Total, minimum, dan maksimum field nominal sama persis.
3. Catat waktu restore dan ukuran data untuk memperkirakan RTO pemulihan dataset berukuran produksi.

**Pertanyaan refleksi:** jika dataset produksi seratus kali lebih besar, berapa perkiraan waktu restore-nya? Apakah angka itu masih masuk dalam RTO yang dijanjikan kepada bisnis?

# Bab 135. Praktikum 3: Mengukur Biaya CPU

**Tujuan:** melihat bagaimana opsi compiler memengaruhi CPU program yang sama.

1. Tulis program COBOL sederhana yang menjumlahkan field packed decimal dalam loop 50 juta kali.
2. Kompilasi dengan `OPT(0)`, jalankan, dan catat waktu CPU dari pesan ringkasan step `IEF032I` di job log.
3. Kompilasi ulang dengan `OPT(2)` dan ARCH setinggi yang didukung mesin lab, jalankan, dan catat lagi.
4. Ulangi masing-masing tiga kali dan hitung rata-ratanya.

$$\text{Penghematan} = \frac{\overline{CPU}_{\text{OPT(0)}} - \overline{CPU}_{\text{OPT(2)}}}{\overline{CPU}_{\text{OPT(0)}}}$$

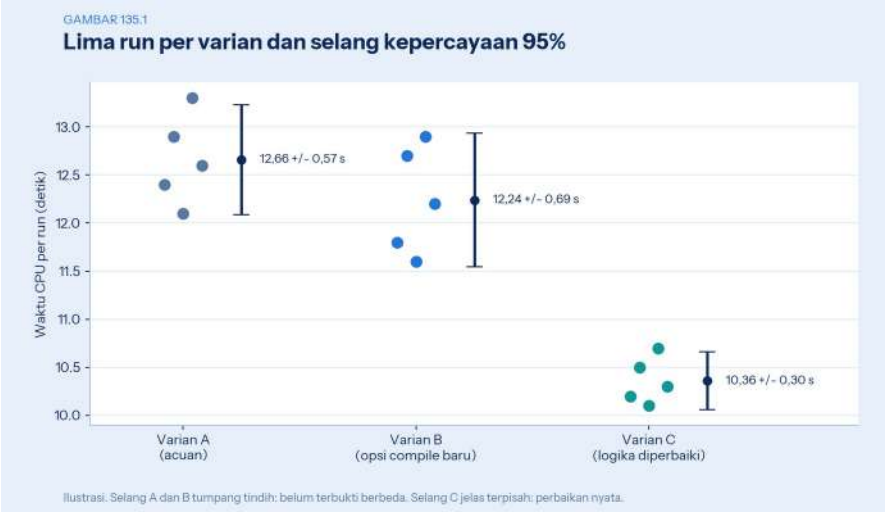
**Pertanyaan refleksi:**

1. Kenapa pengukuran harus diulang beberapa kali, dan kenapa rata-rata elapsed time bukan ukuran yang baik di sistem bersama?
2. Jika penghematan ini berlaku untuk seluruh batch EOD, bagaimana dampaknya pada 4HRA (Bab 42)?

## 135.1 Latihan Lanjutan: Membaca Hasil Benchmark dengan Benar

Waktu CPU dari satu run tidak cukup untuk menyimpulkan bahwa sebuah perubahan lebih cepat. Waktu CPU bervariasi dari run ke run karena

kondisi cache, beban LPAR lain, dan penempatan pekerjaan di prosesor. Perbedaan beberapa persen bisa hanya kebetulan.




*Gambar 135.1 — Lima run per varian dan selang kepercayaan 95%*

Jalankan setiap varian minimal lima kali dalam kondisi yang semirip mungkin, lalu hitung rata-rata dan selang kepercayaannya:

$$\bar{x} \pm t_{0,975;\, n-1} \times \frac{s}{\sqrt{n}} \qquad \text{untuk } n = 5 :\ t \approx 2{,}776$$

Dalam ilustrasi, varian B terlihat lebih cepat 0,4 detik dari varian A. Namun selang kepercayaan keduanya tumpang tindih, sehingga perbedaan itu belum terbukti. Varian C lebih cepat sekitar 2,3 detik, atau sekitar 18%, dengan selang yang terpisah jelas. Itu perbaikan nyata.

**Langkah praktikum:**

1. Jalankan program uji lima kali untuk setiap varian, bergantian (A, B, C, A, B, C, dan seterusnya), agar perubahan kondisi sistem tersebar merata.
2. Ambil waktu CPU dari SMF tipe 30 atau dari ringkasan job, bukan dari waktu elapsed.
3. Hitung rata-rata, simpangan baku, dan selang kepercayaan. Spreadsheet atau Python sudah cukup.

4. Tulis kesimpulan dalam satu kalimat: “C lebih cepat 18% (selang 95%: …)”, atau “B belum terbukti berbeda dari A”.

**Pertanyaan refleksi:** mengapa waktu elapsed tidak cocok untuk membandingkan efisiensi program di sistem yang dipakai bersama? Apa yang akan terjadi pada kesimpulan Anda bila varian B diuji siang hari dan varian A tengah malam?

## Bab 136. Praktikum 4: EOD Mini dengan Restart

**Tujuan:** merasakan restart dari step yang gagal dan memahami kenapa step harus aman diulang.

```
//LAB4     JOB (LAB),'EOD
MINI',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*        RESTART=STEP20   <- aktifkan saat menjalankan
ulang
//STEP10   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD *
FASE10 SELESAI
/*
//SYSUT2   DD DSN=&SYSUID..LAB.EODLOG,DISP=(MOD,CATLG),
//            SPACE=(TRK,(1,1)),RECFM=FB,LRECL=80
//STEP20   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  SET MAXCC = 8
/*
//CHK20    IF (STEP20.RC = 0) THEN
//STEP30   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD *
FASE30 SELESAI
/*
//SYSUT2   DD DSN=&SYSUID..LAB.EODLOG,DISP=MOD
//CHK20E   ENDIF
```

1. Jalankan job. STEP20 berakhir RC 8 (kegagalan buatan) dan STEP30 dilewati.

2. Ubah `SET MAXCC = 8` menjadi `SET MAXCC = 0`.
3. Jalankan ulang tanpa `RESTART`, lalu lihat isi `&SYSUID..LAB.EODLOG`.
4. Hapus dataset log, ulangi dari awal, dan kali ini jalankan ulang dengan `RESTART=STEP20` di kartu JOB.

**Hasil yang diharapkan:** pada percobaan pertama, baris `FASE10 SELESAI` tercatat dua kali karena `DISP=MOD`. Pada percobaan kedua hanya sekali.

**Pertanyaan refleksi:**

1. Kaitkan hasil ini dengan narasi akrual ganda di Bab 118.
2. Ubah STEP10 agar aman diulang meskipun dijalankan dari awal. Teknik apa dari Bab 105.1 yang Anda pakai?

## 136.1 Latihan Lanjutan: Kegagalan di Tiga Titik

Praktikum 4 menguji restart setelah kegagalan di tengah fase. Latihan lanjutan ini memperluasnya menjadi matriks kecil sesuai Bab 58.1: satu fase diuji di tiga titik kegagalan yang berbeda.

| Skenario | Cara menyuntikkan kegagalan | Hasil yang harus dibuktikan |
|---|---|---|
| A. Sebelum fase mulai | Batalkan job segera setelah step pertama dimulai, sebelum update tabel kontrol | Setelah restart, fase berjalan normal satu kali |
| B. Di tengah fase | Batalkan job setelah commit kedua, misalnya dengan parameter uji yang memicu `ABEND` | Tidak ada rekening yang menerima bunga dua kali |
| C. Setelah selesai, sebelum ditandai | Batalkan tepat sebelum update status `SELESAI` | Rerun tidak menambah bunga lagi |

Untuk setiap skenario, bandingkan hasil akhir dengan run acuan yang tidak pernah gagal: jumlah record di tabel akrual, total bunga, dan saldo tiga rekening sampel. Ketiga angka harus identik.

Skenario C paling sering gagal pada rancangan pertama. Jika fase mencatat akrual lalu gagal sebelum menandai dirinya selesai, rerun akan

menjalankan seluruh fase lagi. Bila tidak ada pemeriksaan “sudah diproses” per rekening, bunga akan tercatat ganda. Perbaikannya adalah checkpoint per rekening, atau memeriksa keberadaan akrual untuk tanggal bisnis yang sama sebelum menulis.

**Tugas tambahan:** tulis runbook restart untuk fase ini (Bab 52) berdasarkan apa yang Anda pelajari. Minta rekan yang belum mengikuti praktikum ini menjalankannya tanpa bantuan. Setiap langkah yang membuat dia bertanya adalah langkah yang perlu diperjelas.

## Bab 137. Praktikum 5: Melindungi Dataset dengan RACF

**Tujuan:** membuat profil dataset untuk HLQ milik sendiri dan menguji efeknya. Umumnya pengguna boleh membuat profil untuk dataset dengan HLQ yang sama dengan user ID-nya; tanyakan kebijakan lingkungan lab Anda.

```
ADDSD  'userid.LAB.RAHASIA.**' UACC(NONE)
PERMIT 'userid.LAB.RAHASIA.**' ID(rekan) ACCESS(READ)
LISTDSD DATASET('userid.LAB.RAHASIA.**') ALL
```

1. Buat dataset `userid.LAB.RAHASIA.DATA` dan isi dengan beberapa baris.
2. Minta rekan membaca dataset itu. Berhasil karena ia punya READ.
3. Minta rekan lain yang tidak terdaftar mencoba membaca. Ia menerima pesan `ICH408I`.
4. Cabut akses rekan pertama dengan `PERMIT ... ID(rekan) DELETE` dan ulangi uji.

**Pertanyaan refleksi:**

1. Apa yang terlihat di SYSLOG saat akses ditolak, dan informasi apa di pesan itu yang berguna bagi auditor?
2. Kenapa profil generik `.**` lebih disukai daripada profil diskret untuk setiap dataset?

## 137.1 Latihan Lanjutan: Profil Generik Bertingkat dan Mode WARNING

Latihan ini membuktikan dua perilaku RACF yang sering membingungkan: profil generik yang lebih spesifik selalu menang, dan mode WARNING mengizinkan akses sambil mencatat peringatan (Bab 27.3).

```
ADDGROUP (LABBACA LABAUDIT)
CONNECT (USERA) GROUP(LABBACA)
CONNECT (USERB) GROUP(LABAUDIT)

ADDSD  'HLQ.LAB.**'          UACC(NONE)
PERMIT 'HLQ.LAB.**'          ID(LABBACA)  ACCESS(READ)
ADDSD  'HLQ.LAB.RAHASIA.**' UACC(NONE)
PERMIT 'HLQ.LAB.RAHASIA.**' ID(LABAUDIT) ACCESS(READ)
SETROPTS GENERIC(DATASET) REFRESH
```

**Uji dan catat hasilnya:**

| Uji | User | Dataset | Hasil yang diharapkan |
|---|---|---|---|
| 1 | USERA | `HLQ.LAB.TRX.HARIAN` | Boleh dibaca |
| 2 | USERA | `HLQ.LAB.RAHASIA.GAJI` | Ditolak, `ICH408I` muncul |
| 3 | USERB | `HLQ.LAB.RAHASIA.GAJI` | Boleh dibaca |
| 4 | USERB | `HLQ.LAB.TRX.HARIAN` | Ditolak |

Uji 2 dan 4 memperlihatkan bahwa `HLQ.LAB.RAHASIA.**` yang lebih spesifik menggantikan `HLQ.LAB.**` sepenuhnya untuk dataset di bawahnya. Akses tidak digabung dari kedua profil. Jalankan `LISTDSD DATASET('HLQ.LAB.RAHASIA.GAJI') GENERIC` untuk melihat profil mana yang sebenarnya dipakai.

**Mode WARNING:** jalankan `ALTDSD 'HLQ.LAB.RAHASIA.**' WARNING` dan `SETROPTS GENERIC(DATASET) REFRESH`, lalu ulangi Uji 2. Akses kini diizinkan, tetapi pesan peringatan tetap tercatat. Kembalikan dengan `ALTDSD ... NOWARNING` di akhir latihan.

**Pertanyaan refleksi:** mengapa mode WARNING berguna saat migrasi ke profil baru, dan mengapa berbahaya bila tertinggal di produksi? Bagaimana cara menemukan semua profil dalam mode WARNING di sistem Anda?

# 📝 Bagian XXXII — Latihan Soal Lanjutan

Soal-soal ini menguji pemahaman lintas bab, bukan hafalan perintah. Kerjakan tanpa membuka buku terlebih dahulu, lalu periksa kunci jawaban di Bab 140.

## Bab 138. Soal Pilihan Ganda

1. Job EOD abend SB37 di step SORT. Tindakan pertama yang paling tepat adalah:
   - a. Menaikkan `REGION` job
   - b. Memeriksa ruang storage group dan alokasi SPACE output
   - c. Menambah initiator JES2
   - d. Menjalankan `RUNSTATS`
2. Zona SMP/E yang mencerminkan isi library runtime SYSRES adalah:
   - a. Global zone
   - b. Target zone
   - c. Distribution zone
   - d. HOLDDATA
3. PI sebuah service class importance 1 bernilai 1,6 selama jam sibuk. Artinya:
   -

   a. Tujuan tercapai dengan cadangan 60%
   - b. Tujuan tidak tercapai
   - c. WLM mematikan service class
   - d. CPU di bawah 60%

4. Beban berikut yang paling mungkin dieksekusi di zIIP adalah:
   - a. Program COBOL batch murni
   - b. Permintaan Db2 lewat DDF dari aplikasi Java
   - c. Perintah console operator
   - d. Program Assembler di LPA

5. Perintah untuk melihat siapa yang memegang ENQ atas sebuah dataset adalah:
   - a. `D A,L`
   - b. `D GRS,RES=(SYSDSN,nama)`
   - c. `$D J`
   - d. `D SMS,VOL(...)`

6. Field packed decimal berisi `X'40404040'`. Saat dipakai dalam `ADD`, program akan abend:
   - a. `S0C4`
   - b. `S0C7`

- c. S806
- d. S913

7. Replikasi yang **tidak** melindungi dari ransomware yang mengenkripsi data adalah:
   - a. Safeguarded Copy
   - b. Backup tape terisolasi
   - c. Metro Mirror
   - d. Snapshot dengan retensi panjang
8. Kolektibilitas debitur dengan tunggakan 100 hari adalah:
   - a. Lancar
   - b. Dalam Perhatian Khusus
   - c. Kurang Lancar
   - d. Macet
9. Batas nominal BI-FAST per transaksi yang ditetapkan BI saat peluncuran adalah:
   - a. Rp25 juta
   - b. Rp100 juta
   - c. Rp250 juta
   -

d. Rp1 miliar

10. Perubahan berikut yang paling sulit dibalik adalah:

- a. IPL ke rilis z/OS baru di satu sistem
- b. Aktivasi IODF baru
- c. Aktivasi function level baru di Db2
- d. APPLY PTF yang belum di-ACCEPT

11. Setelah menambah profil di class `OPERCMDS`, perubahan belum berlaku. Kemungkinan besar belum dijalankan:

- a. `SETROPTS RACLIST(OPERCMDS) REFRESH`
- b. `RVARY SWITCH`
- c. `SET PROG=xx`
- d. `F CATALOG,RESTART`

12. JES2 dijalankan dengan start `COLD`. Akibatnya:

- a. Semua job di antrean dan output di spool hilang
- b. JES2 lebih cepat start
- c. Hanya job yang sedang berjalan yang hilang
- d. Tidak ada dampak

## 138.1 Soal Tambahan: Analisis Kuantitatif

Sepuluh soal berikut menguji pemahaman subbab-subbab analisis. Kerjakan tanpa membuka bab terkait, lalu periksa jawabannya di akhir.

1. Sysplex dua sistem, masing-masing berjalan di 70% CPU. Apa yang terjadi bila satu sistem gagal? (a) Tidak ada masalah karena sysplex otomatis menyeimbangkan. (b) Sistem yang tersisa diminta memikul sekitar 140%. (c) WLM menambah kapasitas otomatis. (d) Beban turun separuh.
2. Waktu respons sebuah volume didominasi IOSQ. Tindakan pertama yang paling tepat? (a) Menambah cache array. (b) Menambah alias HyperPAV. (c) Memperbesar BLKSIZE. (d) Menambah jalur ke situs DR.
3. Window TCP 64 KB dan RTT 20 ms. Throughput maksimum satu koneksi kira-kira? (a) 3,3 MB/detik. (b) 33 MB/detik. (c) 125 MB/detik. (d) Tergantung kapasitas link saja.
4. Sebuah job dengan slack 35 menit dipercepat 20 menit. Pengaruhnya ke durasi EOD? (a) Berkurang 20 menit. (b) Berkurang 35 menit. (c) Tidak berubah. (d) Bertambah.
5. Total kontrol mana yang mendeteksi transaksi yang tertukar rekening tujuannya dengan nominal sama? (a) Jumlah record. (b) Total nominal. (c) Hash total nomor rekening. (d) Tidak ada.
6. Snapshot disimpan 7 hari, tetapi penyerang berdiam 20 hari sebelum merusak data. Akibatnya? (a) Snapshot terbaru aman. (b) Semua snapshot yang tersedia mungkin sudah tercemar. (c) Replikasi sinkron menyelamatkan data. (d) Tidak berpengaruh.
7. `EXPLAIN` transaksi online menunjukkan `ACCESSTYPE = R`. Artinya? (a) Akses lewat index unik. (b) Tablespace scan. (c) Akses lewat MQT. (d) Akses baca saja.
8. Permintaan datang dengan kunci idempotensi yang sudah ada, tetapi payload berbeda. Respons yang benar? (a) Proses sebagai transaksi baru. (b) Kembalikan hasil lama. (c) Tolak sebagai konflik. (d) Abaikan tanpa jawaban.
9. Channel MQ dengan RTT 20 ms dan commit 5 ms mengirim satu pesan per batch. Throughput maksimum kira-kira? (a) 40

pesan/detik. (b) 400 pesan/detik. (c) 2.000 pesan/detik. (d) Tidak terbatas.

10. Peluang insiden 1% per perubahan dan ada 40 perubahan dalam sebulan. Peluang minimal satu insiden kira-kira? (a) 1%. (b) 10%. (c) 33%. (d) 40%.

**Kunci:** 1-b (Bab 11.1), 2-b (Bab 6.3), 3-a (Bab 39.1), 4-c (Bab 22.1), 5-c (Bab 61.1), 6-b (Bab 111.1), 7-b (Bab 33.4), 8-c (Bab 102.3), 9-a (Bab 165.1), 10-c (Bab 126.1).

## Bab 139. Soal Situasi

13. Pukul 02.10, proyeksi waktu selesai EOD menunjukkan 05.40, sedangkan layanan harus buka 05.00. Susun lima langkah pertama Anda, beserta urutan dan alasannya.
14. Rekonsiliasi BI-FAST pagi ini menunjukkan 37 transaksi “hanya di core” senilai total Rp412 juta. Jelaskan kemungkinan penyebab dan tindakan per kategori.
15. Auditor meminta bukti bahwa backup produksi lengkap. Laporan RC 0 harian ditolak. Rancang bukti yang bisa diterima.
16. Uji DR berhasil di level teknis, tetapi bisnis menilai RTO 3 jam terlalu lama. Chart rincian RTO menunjukkan keputusan dan validasi memakan setengah waktu. Usulkan tiga perbaikan.
17. Sebuah transaksi CICS baru abend ASRA hanya di satu AOR dari delapan. Rancang langkah diagnosis.

## Bab 140. Kunci Jawaban

| No | Jawaban | Penjelasan singkat | Bab |
|---|---|---|---|
| 1 | b | x37 adalah masalah ruang: storage group penuh atau SPACE terlalu kecil | 16, 18 |
| 2 | b | Target zone mencatat isi target library runtime | 14 |

| No | Jawaban | Penjelasan singkat | Bab |
|---|---|---|---|
| 3 | b | PI di atas 1 berarti tujuan meleset | 40 |
| 4 | b | Akses DDF dan Java termasuk beban eligible zIIP | 4 |
| 5 | b | GRS menyimpan informasi ENQ `SYSDSN` | 48 |
| 6 | b | Nibble tanda `0` tidak sah, menghasilkan data exception | 74 |
| 7 | c | Replikasi sinkron menyalin data terenkripsi ke situs DR | 113 |
| 8 | c | 91 sampai 120 hari termasuk Kurang Lancar | 99 |
| 9 | c | Ditetapkan BI Rp250 juta per transaksi pada tahap awal | 101 |
| 10 | c | Function level baru sering tidak dapat dikembalikan | 131 |
| 11 | a | Class yang di-RACLIST butuh refresh | 28 |
| 12 | a | COLD start menghapus isi spool | 12 |

Panduan jawaban soal situasi:

- **Soal 13:** identifikasi fase lambat dan penyebabnya (RB-043); naikkan prioritas WLM job EOD kritikal; tunda fase non-kritikal seperti laporan; aktifkan kapasitas sementara bila tersedia; koordinasikan keputusan pembukaan bertahap dengan bisnis sebelum pukul 04.00.

- **Soal 14:** pastikan status tiap transaksi lewat inquiry (Bab 102.1); yang pasti gagal dibalik ke nasabah; yang tidak diketahui ditahan di suspense; periksa log payment hub untuk pesan yang tidak terkirim; laporkan pola bila ada gangguan koneksi.
- **Soal 15:** laporan cakupan yang membandingkan dataset di katalog produksi dengan dataset yang dicadangkan, ditambah bukti uji restore berkala (Bab 120).
- **Soal 16:** tetapkan kriteria dan pejabat deklarasi bencana di muka; otomasi validasi GL dan rekonsiliasi; latih tim dengan skenario bergilir (Bab 116).
- **Soal 17:** bandingkan versi program di AOR itu dengan AOR lain (`CEMT I PROG`); periksa `DFHRPL` dan waktu PHASEIN; analisis transaction dump untuk offset; periksa data atau resource yang hanya diakses AOR itu (RB-016, Bab 67).

| Skor soal 1–12 | Interpretasi |
|---|---|
| 11–12 | Siap memegang tanggung jawab jaga mandiri |
| 8–10 | Kuat di dasar; ulangi bab yang terkait jawaban salah |
| Di bawah 8 | Ulangi Bagian I sampai XIII sebelum melanjutkan |

# 🎯 Bagian XXXIII — Proyek Studi Mandiri

Proyek berikut menggabungkan beberapa bab menjadi satu hasil kerja yang bisa dipakai tim. Setiap proyek punya lingkup, hasil yang diserahkan, dan bab rujukan, lalu dinilai dengan rubrik yang sama.

## Bab 141. Proyek Tingkat Menengah

| Proyek | Lingkup | Hasil yang diserahkan | Bab rujukan |
|---|---|---|---|
| P1. Dashboard health check | Perluas skrip HLTHCHK agar menulis hasil ke dataset CSV setiap pagi, lalu tarik dengan Zowe CLI dan tampilkan tren 30 hari | Skrip REXX, JCL, skrip Zowe, contoh grafik | 26, 82–87 |
| P2. Kamus pesan internal | Ambil message ID dari OPERLOG 30 hari, hitung frekuensinya, lalu tulis arti dan tindakan untuk 50 pesan teratas | Tabel kamus pesan yang tertaut ke runbook | 50, 88 |
| P3. Laporan cakupan backup | Bandingkan dataset produksi di katalog dengan dataset yang tercatat di backup, lalu laporkan selisihnya | Job laporan harian dan prosedur tindak lanjut | 45, 120 |

# Bab 142. Proyek Tingkat Lanjut

| Proyek | Lingkup | Hasil yang diserahkan | Bab rujukan |
|---|---|---|---|
| P4. EOD mini dengan tabel kontrol | Tiga fase EOD di lab Db2 dengan klaim atomik, verifikasi total kontrol, dan uji kegagalan pada setiap fase | DDL, program atau skrip, JCL, laporan uji kegagalan | 58, 61, 105 |
| P5. Proyeksi kapasitas 12 bulan | Olah SMF 70/72 menjadi tren MSU, 4HRA puncak bulanan, dan proyeksi dengan dua skenario pertumbuhan | Laporan kapasitas dengan grafik dan rekomendasi | 41, 42 |
| P6. Otomasi checklist rolling IPL | Playbook Ansible yang menjalankan pemeriksaan sebelum dan sesudah IPL di lingkungan non-produksi | Playbook, inventori, bukti uji | 55, 60 |


## Bab 143. Rubrik Penilaian

| Kriteria | Bobot | Cukup (1) | Baik (2) | Sangat baik (3) |
|---|---|---|---|---|
| Kebenaran teknis | 30% | Berjalan pada kondisi normal | Menangani kesalahan umum | Menangani kegagalan dan hasilnya |

| Kriteria | Bobot | Cukup (1) | Baik (2) | Sangat baik (3) |
|---|---|---|---|---|
| | | | | terverifikasi independen |
| Keamanan dan kontrol | 20% | Tidak memakai hak berlebihan | Hak minimum dan terdokumentasi | Termasuk jejak audit dan pemisahan tugas |
| Keterulangan | 20% | Bisa dijalankan ulang oleh pembuat | Bisa dijalankan orang lain dengan dokumentasi | Otomatis, idempoten, dan terjadwal |
| Dokumentasi dan runbook | 20% | Ada catatan langkah | Runbook mengikuti template Bab 95 | Runbook sudah diuji orang lain |
| Komunikasi | 10% | Menjelaskan apa yang dibuat | Menjelaskan kenapa dan batasannya | Menghubungkan hasil dengan risiko bisnis |

Nilai akhir dihitung sebagai rata-rata tertimbang:

$$N = \sum_{k} w_k \times s_k \qquad \text{dengan} \sum_{k} w_k = 1,\ s_k \in \{1, 2, 3\}$$

Nilai 2,5 atau lebih menandakan hasil kerja yang siap dipakai tim. Nilai di bawah 2 berarti proyek perlu satu iterasi lagi sebelum diserahkan.

# 🔤 Bagian XXXIV — CCSID, EBCDIC/Unicode, dan Pertukaran Data

Mainframe berbicara EBCDIC, sedangkan hampir semua sistem lain berbicara ASCII atau Unicode. Sebagian besar insiden “data aneh” di interface berawal dari salah paham tentang di mana dan bagaimana konversi terjadi.

## Bab 144. Code Page dan CCSID

CCSID (*Coded Character Set Identifier*) adalah nomor yang menyatakan pengodean sebuah data. Setiap dataset, kolom Db2, pesan MQ, dan file zFS idealnya punya CCSID yang jelas.

| CCSID | Nama | Dipakai untuk |
|---|---|---|
| 37 | EBCDIC US/Kanada | Banyak aplikasi COBOL lama |
| 1047 | EBCDIC Latin-1 open systems | z/OS UNIX, C, dan banyak produk modern |
| 1140 | EBCDIC US dengan simbol euro | Varian 37 yang lebih baru |
| 819 | ISO 8859-1 (ASCII Latin-1) | Sistem terbuka lama |
| 1208 | UTF-8 | API, JSON, dan hampir semua sistem modern |
| 1200 | UTF-16 | Java dan sebagian aplikasi Windows |

CCSID 37 dan 1047 hampir sama, tetapi beberapa karakter berbeda posisi, misalnya kurung siku `[` dan `]`. Program yang menulis JSON dengan CCSID yang salah akan menghasilkan kurung yang rusak di sisi penerima.


*Gambar 144.1 — Satu record, dua pengodean*

## 144.1 Analisis: Memilih CCSID untuk Data Baru

Data lama di mainframe umumnya tersimpan dalam EBCDIC dengan CCSID 37 atau 1047. Data baru yang datang dari kanal digital semakin sering berisi karakter yang tidak ada di code page itu: huruf beraksen dari nama nasabah internasional, tanda kutip tipografis dari aplikasi mobile, atau simbol mata uang tertentu. Pilihan CCSID untuk tabel dan file baru menentukan apakah karakter-karakter itu tersimpan utuh.

| CCSID | Nama umum | Cocok untuk |
|---|---|---|
| 37 | EBCDIC Amerika Serikat | Data lama dan aplikasi COBOL klasik |
| 1047 | EBCDIC Latin-1 untuk sistem terbuka | z/OS UNIX dan file yang dipakai bersama tool Unix |
| 1140 | Seperti 37, dengan tanda euro | Data yang membutuhkan simbol euro |
| 819 | ISO 8859-1 (ASCII Latin-1) | Pertukaran dengan sistem lama berbasis ASCII |
| 1208 | UTF-8 | Data dari web, API, dan aplikasi mobile |

| CCSID | Nama umum | Cocok untuk |
|---|---|---|
| 1200 | UTF-16 | Kolom Db2 grafis Unicode |

Karakter yang tidak ada di code page tujuan diganti dengan karakter pengganti, `X'3F'` di EBCDIC atau `X'1A'` di ASCII. Penggantian ini terjadi tanpa error. Nama nasabah yang berubah menjadi “Jos? Mart?nez” di laporan adalah tanda klasiknya.

**Rekomendasi praktis:**

- Tabel Db2 baru yang menerima data dari kanal digital dibuat dengan encoding Unicode.
- Kolom nama dan alamat diberi panjang yang cukup untuk UTF-8, karena satu karakter beraksen memakan dua byte atau lebih.
- Program COBOL yang membaca kolom Unicode memakai tipe data yang sesuai atau konversi eksplisit di satu tempat.
- Tabel dan file lama tidak dikonversi massal tanpa proyek khusus, karena terlalu banyak program bergantung pada layout byte-nya.

Catat CCSID setiap tabel dan file interface di dokumentasi data (Bab 145.1). Pertanyaan “CCSID apa?” yang dijawab dari dokumen, bukan dari tebakan, mencegah sebagian besar masalah karakter lintas platform.

## Bab 145. Di Mana Konversi Terjadi

| Jalur | Mekanisme konversi | Pengaturan penting |
|---|---|---|
| FTP | Mode `ASCII` mengonversi, mode `BINARY` tidak | `SITE/LOCSITE ENCODING`, `SBDATACONN` untuk CCSID eksplisit |
| z/OS UNIX | Tag file dan auto-conversion | `chtag`, variabel `_BPXK_AUTOCVT` |
| Db2 | Konversi otomatis antara CCSID klien dan kolom | CCSID tablespace dan aplikasi |
| IBM MQ | Konversi saat GET jika diminta | Format pesan `MQFMT_STRING` dan opsi convert |
| z/OS Connect / JSON | Pemetaan copybook ke JSON | Definisi service dan CCSID |

| Jalur | Mekanisme konversi | Pengaturan penting |
|---|---|---|
| | UTF-8 | copybook |
| Program | Unicode Services atau `iconv` | Tabel konversi yang dipilih |

Prinsip yang mencegah hampir semua masalah: konversi dilakukan tepat sekali, di satu titik yang disepakati, dan CCSID sumber serta tujuan dinyatakan eksplisit, bukan dibiarkan default.

## 145.1 Analisis: Menghitung Titik Konversi di Sepanjang Jalur Data

Setiap konversi karakter adalah peluang kerusakan. Aturan praktis yang paling berguna: data dikonversi tepat sekali, di batas yang jelas antara dunia EBCDIC dan dunia ASCII atau UTF-8. Untuk menegakkan aturan itu, jalur data harus dipetakan dan titik konversinya dihitung.

| Jalur | Titik konversi | Risiko |
|---|---|---|
| Payment hub → MQ → CICS | Satu, di MQ dengan `MQGMO_CONVERT` atau di aplikasi | Rendah bila CCSID pesan diisi benar |
| File CSV mitra → FTP ASCII → dataset → program | Satu, di FTP mode teks | Karakter khusus bila code page FTP tidak sesuai |
| File → FTP ASCII → dataset → FTP ASCII kembali | Dua, bolak-balik | Karakter yang tidak ada di salah satu code page hilang |
| Db2 → CDC → Kafka → aplikasi cloud | Satu, di komponen CDC | Rendah bila CCSID kolom terdefinisi benar |
| File biner dengan field packed → FTP ASCII | Satu, tetapi seharusnya nol | Field packed rusak; wajib mode biner |

Baris terakhir adalah kesalahan klasik. File yang berisi field packed decimal atau biner tidak boleh dikonversi sama sekali, karena byte-nya bukan karakter. Konversi di file seperti ini merusak angka tanpa memicu error, dan kerusakannya baru terlihat saat program penerima menghasilkan `S0C7` atau angka yang tidak masuk akal.

$$R_{\text{jalur}} \approx 1 - \prod_{k=1}^{n} (1 - r_k)$$

r adalah peluang kesalahan di setiap titik konversi k. Rumus ini menunjukkan mengapa jumlah titik konversi harus sesedikit mungkin: setiap titik tambahan menambah peluang kerusakan, sekecil apa pun risiko per titiknya.

Dokumentasikan setiap interface dengan satu baris: format sumber, format tujuan, di mana konversi terjadi, dan CCSID yang dipakai. Dokumen satu halaman seperti ini menghemat berjam-jam investigasi saat karakter nama nasabah tiba-tiba berubah menjadi simbol aneh.

## Bab 146. Angka dan Urutan Lintas Platform

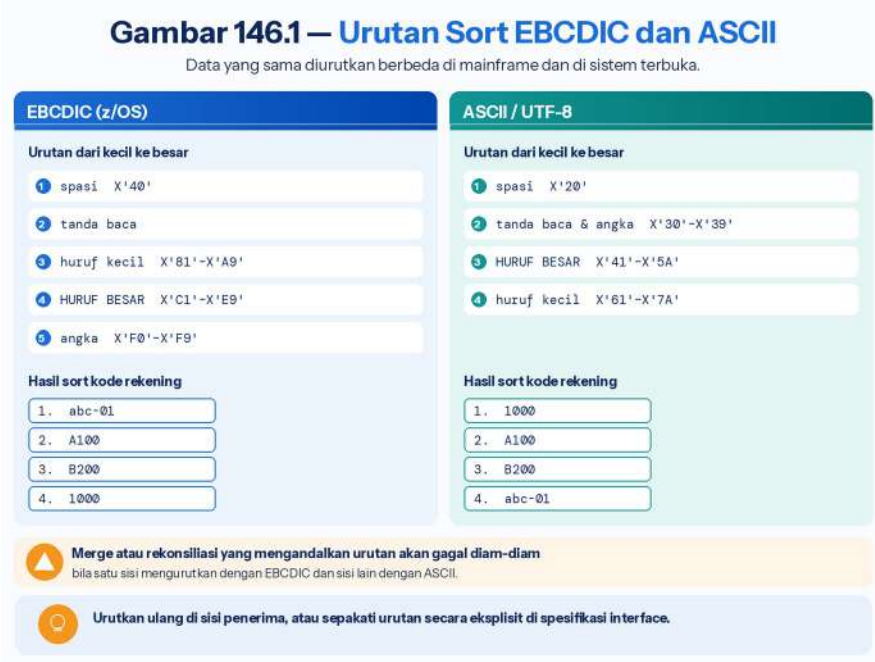
| Aspek | Mainframe | Sistem terbuka umum | Akibat jika diabaikan |
|---|---|---|---|
| Packed decimal | Asli, dipakai luas | Tidak dikenal | Nilai rusak jika dikonversi sebagai teks |
| Urutan byte biner | Big-endian | Sering little-endian (x86) | Angka biner terbaca terbalik |
| Urutan sort teks | Huruf kecil < huruf besar < angka | Angka < huruf besar < huruf kecil | Hasil sort berbeda, rekonsiliasi gagal |
| Tanda desimal | Tersirat (V di PIC) | Titik atau koma eksplisit | Nilai bergeser faktor 10 atau 100 |

Perbedaan urutan sort sering luput. File yang diurutkan di mainframe lalu dicocokkan dengan algoritma merge di sistem terbuka yang mengasumsikan urutan ASCII akan menghasilkan “selisih” palsu.

Format yang aman untuk pertukaran angka adalah teks bertanda dengan panjang tetap dan jumlah desimal yang disepakati, misalnya `+000001234.50`, atau JSON dengan angka desimal eksplisit.

## 146.1 Analisis: Urutan Sort yang Berbeda

Urutan karakter di EBCDIC berbeda dari ASCII. Perbedaan ini tidak terlihat sampai ada proses yang bergantung pada urutan: merge dua file, perbandingan berurutan untuk rekonsiliasi, atau pencarian biner di atas data yang dianggap sudah terurut.




*Gambar 146.1 — Urutan sort EBCDIC dan ASCII*

Di EBCDIC, huruf kecil berada sebelum huruf besar, dan angka berada paling akhir. Di ASCII, angka berada sebelum huruf, dan huruf besar sebelum huruf kecil. Akibatnya, daftar kode rekening yang sama akan diurutkan berbeda di kedua platform.

| Proses | Apa yang terjadi bila urutan berbeda |
|---|---|
| Merge dua file terurut (match-merge) | Record dianggap tidak berpasangan, padahal pasangannya ada di posisi lain |
| Rekonsiliasi berurutan | Selisih palsu dalam jumlah besar |
| Pencarian biner | Record yang ada tidak ditemukan |
| Laporan yang diharapkan berurutan | Urutan terlihat “acak” bagi pengguna di |

| Proses | Apa yang terjadi bila urutan berbeda |
|---|---|
| | platform lain |

Kesalahan ini berbahaya karena tidak memicu error. Program merge tetap berjalan sampai selesai, hanya saja hasilnya salah. Rekonsiliasi yang menghasilkan ribuan selisih setelah migrasi satu komponen ke sistem terbuka sering disebabkan oleh masalah ini.

Tiga cara mencegahnya:

1. **Urutkan ulang di sisi penerima** sebelum proses yang bergantung pada urutan, dengan aturan urutan platform penerima.
2. **Sepakati urutan secara eksplisit** di spesifikasi interface, misalnya “diurutkan berdasarkan nilai byte UTF-8”.
3. **Gunakan kunci numerik** yang berpadding angka nol di depan untuk kunci yang dibandingkan lintas platform. Angka murni diurutkan sama di kedua platform selama panjangnya tetap.

DFSORT dapat mengurutkan dengan urutan ASCII bila diperlukan. Namun keputusan terbaik biasanya adalah membuat penerima tidak bergantung pada urutan pengirim sama sekali.

## Bab 147. Validasi File Interface

Setiap file interface sebaiknya membawa header dan trailer yang memungkinkan penerima memeriksa kelengkapan tanpa bertanya kepada pengirim.

| Bagian | Isi |
|---|---|
| Header | Kode interface, tanggal bisnis, nomor urut file, CCSID data |
| Detail | Record data |
| Trailer | Jumlah record, total nominal (hash total), penanda akhir |

```
ALGORITMA ValidasiFileMasuk(file)
  h <- record pertama; t <- record terakhir
  jika h.kode_interface bukan yang diharapkan maka
```

```
TOLAK("interface salah")
  jika h.tanggal_bisnis <> tanggal_bisnis_berjalan maka
TOLAK("tanggal salah")
  jika h.nomor_urut <> nomor_terakhir + 1 maka TOLAK("file
hilang atau ganda")
  jika jumlah_detail <> t.jumlah_record maka TOLAK("jumlah
record")
  jika jumlah(detail.nominal) <> t.hash_total maka
TOLAK("hash total")
  untuk setiap field numerik: jika tidak valid maka
TOLAK("data numerik")
  simpan file sebagai generasi GDG baru; catat nomor_urut
```

### RB-053: Karakter rusak di file interface

- **Gejala:** nama nasabah atau alamat berisi karakter aneh; kurung atau simbol tertentu hilang.
- **Diagnosis:** lihat hex record dengan ISPF `HEX ON`; bandingkan dengan CCSID yang dinyatakan pengirim.
- **Tindakan:** proses ulang dengan tabel konversi yang benar; jika data sudah masuk ke tabel, koreksi lewat jalur resmi.
- **Pencegahan:** CCSID eksplisit di header file dan di konfigurasi transfer.

### RB-054: Rekonsiliasi menunjukkan selisih palsu karena urutan

- **Gejala:** ribuan record “tidak cocok” padahal nominal total sama.
- **Diagnosis:** periksa apakah kedua file diurutkan dengan collating sequence yang sama.
- **Tindakan:** urutkan ulang kedua file dengan urutan yang sama sebelum dicocokkan, atau cocokkan berdasarkan kunci, bukan posisi.
- **Pencegahan:** spesifikasi interface menyatakan urutan dan collating sequence secara eksplisit.

## 147.1 Analisis: Karantina File Interface

File interface yang gagal validasi menimbulkan dilema. Menolaknya berarti transaksi di dalamnya tertunda, dan nasabah bisa terdampak. Menerimanya berarti risiko data salah masuk ke core banking. Dilema ini lebih mudah diputuskan bila ada tempat ketiga: karantina.

| Keputusan | Kapan | Akibat |
|---|---|---|
| Terima | Semua validasi lolos | Diproses normal |
| Karantina | Kesalahan terbatas dan dapat diisolasi, misalnya beberapa record dengan format salah | Record bermasalah dipisah, sisanya diproses; mitra diberi tahu |
| Tolak seluruhnya | Kesalahan struktural: trailer tidak cocok, file terpotong, tanggal data salah | Tidak ada yang diproses; mitra mengirim ulang |

Aturan pentingnya: kesalahan struktural selalu berarti tolak seluruhnya. Jika jumlah record di trailer tidak cocok dengan isi file, tidak ada yang tahu record mana yang hilang. Memproses sebagian file seperti ini sama berbahayanya dengan memproses file yang salah.

```
ALGORITMA ProsesFileInterface(file)
  jika validasi struktural gagal (header, trailer, total
kontrol, tanggal) maka
      TOLAK file; beri tahu mitra; catat alasan
  selain itu:
      pisahkan record yang gagal validasi field ke dataset
karantina
      jika porsi karantina > ambang (misalnya 1%) maka
TOLAK file (terlalu banyak yang salah)
      proses record yang lolos; hitung ulang total kontrol
untuk bagian yang diproses
      kirim daftar record karantina ke mitra dan ke tim
operasional
  simpan statistik: jumlah file, jumlah ditolak, porsi
karantina per mitra
```

Statistik karantina per mitra adalah indikator kualitas data yang berguna. Mitra yang porsi karantinanya naik dari bulan ke bulan sedang mengalami masalah di sistemnya sendiri. Membicarakannya lebih awal mencegah

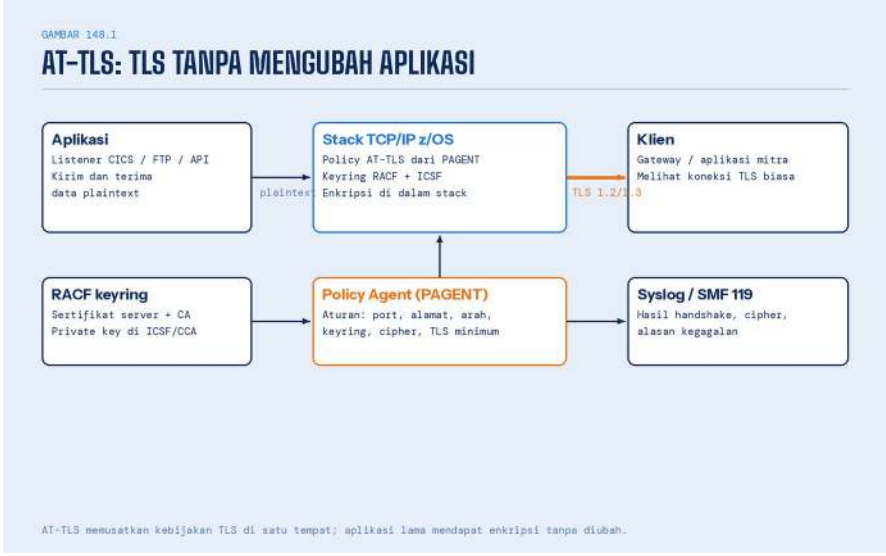
insiden yang lebih besar. Statistik ini juga membantu merumuskan perjanjian tingkat layanan dengan mitra secara objektif.

# 🔒 Bagian XXXV — Jaringan dan Keamanan Transport Mendalam

Bab 36 sampai 39 membahas jaringan dasar. Bagian ini masuk ke lapisan yang paling sering menimbulkan insiden di era API: TLS, sertifikat, dan penyaringan lalu lintas.

## Bab 148. AT-TLS

AT-TLS (*Application Transparent TLS*) menjalankan TLS di dalam stack TCP/IP z/OS berdasarkan kebijakan, sehingga aplikasi tidak perlu diubah. Kebijakan dikelola Policy Agent (PAGENT) dan bisa diperbarui tanpa restart aplikasi.




*Gambar 148.1 — AT-TLS: TLS tanpa mengubah aplikasi*

Contoh kebijakan untuk listener API di port 3000:

```
TTLSRule                        CICS_API_Inbound
{
  LocalPortRange                3000
  Direction                     Inbound
  TTLSGroupActionRef            grpAktif
```

```
  TTLSEnvironmentActionRef  envServer
}
TTLSGroupAction             grpAktif
{
  TTLSEnabled               On
}
TTLSEnvironmentAction       envServer
{
  HandshakeRole             Server
  TTLSKeyringParmsRef       ringApi
  TTLSEnvironmentAdvancedParmsRef advTls
}
TTLSKeyringParms            ringApi
{
  Keyring                   CICSPRD/APIRING
}
TTLSEnvironmentAdvancedParms advTls
{
  SSLv3                     Off
  TLSv1                     Off
  TLSv1.1                   Off
  TLSv1.2                   On
  TLSv1.3                   On
}

F PAGENT,REFRESH                    Muat ulang kebijakan
setelah perubahan
D TCPIP,,NETSTAT,TTLS               Koneksi yang
dilindungi AT-TLS
D TCPIP,,NETSTAT,TTLS,CONNDETAIL    Detail versi TLS dan
cipher per koneksi
```

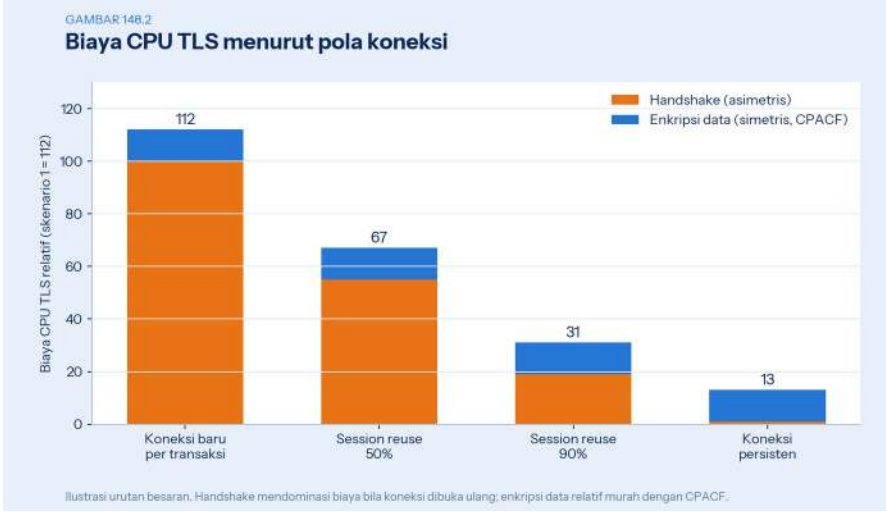
Keyring harus dimiliki user ID address space yang menjalankan listener, atau dirujuk lengkap dalam bentuk `pemilik/nama` dengan izin baca yang sesuai. Kesalahan kepemilikan keyring adalah penyebab umum handshake gagal pada konfigurasi baru.

## 148.1 Analisis: Biaya CPU TLS dan Pola Koneksi

Mengaktifkan TLS di semua jalur sering dikhawatirkan akan menaikkan CPU secara besar. Kenyataannya, biaya TLS sangat bergantung pada pola koneksi, bukan pada TLS itu sendiri. Bagian yang mahal adalah handshake, yaitu negosiasi kunci dengan kriptografi asimetris. Enkripsi data setelah

handshake memakai kriptografi simetris yang dipercepat CPACF di setiap prosesor IBM Z, sehingga relatif murah.

$$C_{\text{TLS}} \approx N_{\text{handshake penuh}} \times c_{\text{penuh}} + N_{\text{resume}} \times c_{\text{resume}} + B_{\text{data}} \times c_{\text{simetris}}$$




*Gambar 148.2 — Biaya CPU TLS menurut pola koneksi*

Aplikasi yang membuka koneksi TLS baru untuk setiap transaksi membayar handshake penuh setiap kali. Aplikasi yang memakai kembali sesi (session resumption) membayar handshake penuh hanya sesekali. Aplikasi dengan koneksi persisten, misalnya gateway yang menjaga kumpulan koneksi ke z/OS Connect, hampir tidak membayar handshake sama sekali.

| Pola | Biaya | Rekomendasi |
|---|---|---|
| Koneksi baru per transaksi | Tertinggi | Hindari untuk trafik volume tinggi |
| Session resumption | Menengah sampai rendah | Aktifkan cache sesi di AT-TLS dan di klien |
| Koneksi persisten (pooling) | Terendah | Pola terbaik untuk gateway dan middleware |

Operasi asimetris saat handshake dapat dipercepat oleh CPACF dan Crypto Express, tergantung algoritma dan konfigurasi. Sertifikat dengan kurva eliptik biasanya lebih ringan daripada RSA berukuran besar untuk tingkat keamanan yang setara.

Sebelum menyimpulkan bahwa “TLS terlalu mahal”, ukur jumlah handshake penuh per detik dari statistik AT-TLS dan bandingkan dengan jumlah transaksi. Jika rasionya mendekati satu, masalahnya adalah pola koneksi, dan perbaikannya ada di sisi klien, bukan dengan mematikan enkripsi.

# Bab 149. Keyring dan Sertifikat di RACF

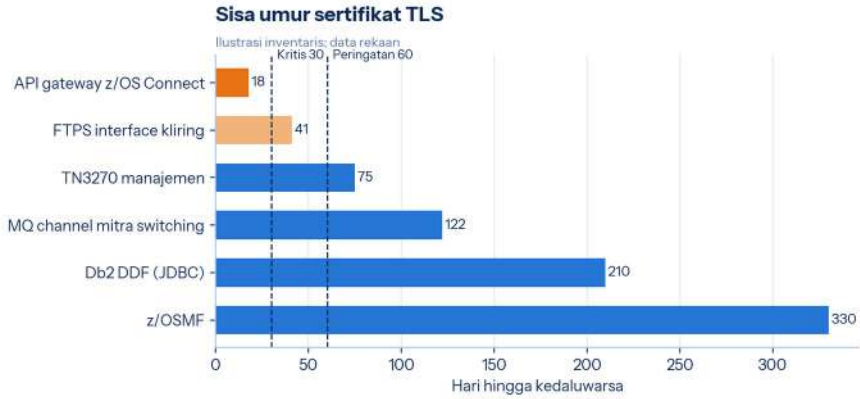
Sertifikat dan kunci privat di z/OS disimpan dan dikelola RACF, dengan kunci yang bisa dilindungi hardware lewat ICSF. Keyring mengelompokkan sertifikat yang dipakai satu aplikasi.

```
RACDCERT ID(CICSPRD) GENCERT
SUBJECTSDN(CN('api.bank.co.id') O('Bank Contoh') C('ID'))
-
         SIZE(2048) WITHLABEL('API-2026')
NOTAFTER(DATE(2027-09-30))
RACDCERT ID(CICSPRD) GENREQ(LABEL('API-2026'))
DSN('SEC.CSR.API2026')
RACDCERT ID(CICSPRD) ADD('SEC.CERT.API2026')
WITHLABEL('API-2026') TRUST
RACDCERT ID(CICSPRD) ADDRING(APIRING)
RACDCERT ID(CICSPRD) CONNECT(ID(CICSPRD) LABEL('API-2026')
RING(APIRING) DEFAULT)
RACDCERT ID(CICSPRD) CONNECT(CERTAUTH LABEL('CA-
INTERMEDIATE') RING(APIRING))
RACDCERT ID(CICSPRD) LISTRING(APIRING)
SETROPTS RACLIST(DIGTCERT DIGTRING) REFRESH
```

Urutannya: buat kunci dan sertifikat sementara, buat CSR, kirim ke CA, tambahkan sertifikat yang ditandatangani dengan label yang sama, sambungkan ke keyring bersama rantai CA-nya, lalu refresh.

## 149.1 Siklus hidup sertifikat

Sertifikat yang kedaluwarsa adalah insiden yang sepenuhnya bisa dicegah. Inventaris dengan sisa umur per sertifikat membuat pembaruan terjadwal jauh sebelum tenggat.




*Ilustrasi · data rekaan*

```
ALGORITMA PerbaruiSertifikat(sert)
  H-60: buat kunci dan CSR baru dengan label baru; kirim
ke CA
  H-45: terima sertifikat; tambahkan ke RACF; verifikasi
rantai CA lengkap
  H-30: uji di lingkungan non-produksi dengan klien
sesungguhnya
  H-14: jadwalkan perubahan; hubungkan label baru sebagai
DEFAULT di keyring
        refresh RACLIST dan kebijakan AT-TLS; uji
handshake dari mitra
  H-7 : pastikan tidak ada koneksi yang masih memakai
sertifikat lama
  Setelah kedaluwarsa: hapus sertifikat lama dari keyring
```

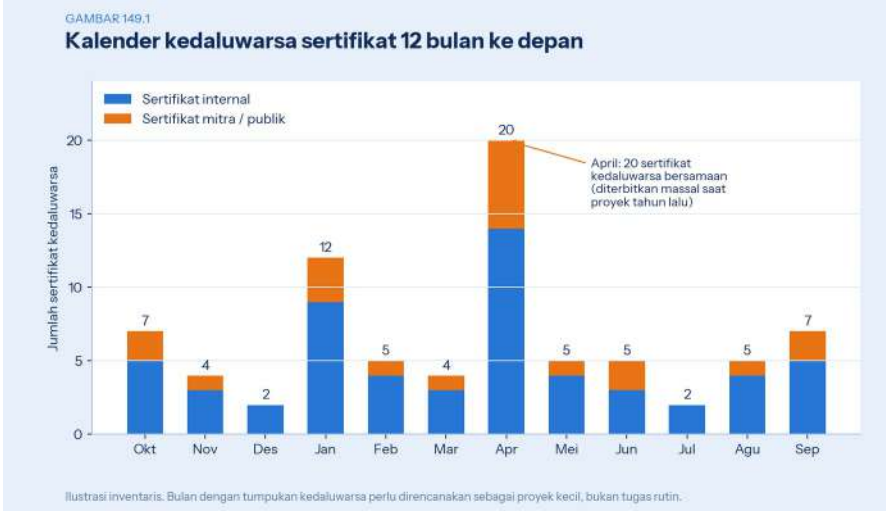
## 149.2 Analisis: Merencanakan Pembaruan Sertifikat

Sertifikat yang kedaluwarsa adalah salah satu penyebab gangguan yang paling bisa dicegah, tetapi tetap sering terjadi. Penyebabnya jarang teknis. Biasanya pembaruan dimulai terlalu lambat untuk proses yang ternyata panjang.

$$T_{\text{tenggang}} \geq T_{\text{permintaan}} + T_{\text{persetujuan CA}} + T_{\text{distribusi ke mitra}} + T_{\text{pemasangan}} + T_{\text{cadangan}}$$

Contoh untuk sertifikat yang dipakai bersama mitra switching: permintaan dan persetujuan internal 5 hari, penerbitan oleh CA 10 hari, distribusi dan pemasangan di sisi mitra 14 hari karena mengikuti jadwal perubahan

mitra, pemasangan di mainframe 3 hari, dan cadangan 14 hari. Totalnya 46 hari. Peringatan 30 hari sebelum kedaluwarsa, yang sering dipakai sebagai standar, tidak cukup untuk sertifikat jenis ini.




*Gambar 149.1 — Kalender kedaluwarsa sertifikat 12 bulan ke depan*

Kalender kedaluwarsa memperlihatkan pola yang tidak terlihat dari daftar biasa. Dalam ilustrasi, 20 sertifikat kedaluwarsa pada bulan April, karena semuanya diterbitkan bersamaan saat sebuah proyek setahun sebelumnya. Bulan seperti ini perlu direncanakan sebagai proyek kecil tersendiri, dengan jadwal yang disepakati bersama mitra jauh hari sebelumnya.

| Jenis sertifikat | Ambang peringatan pertama | Alasan |
|---|---|---|
| Internal, tanpa pihak luar | 45 hari | Proses internal saja |
| Dipakai bersama mitra | 90 hari | Mengikuti jadwal perubahan pihak lain |
| CA root atau intermediate | 180 hari | Berdampak ke banyak sertifikat dan sistem sekaligus |

Sebarkan tanggal kedaluwarsa bila memungkinkan. Sertifikat yang diterbitkan bersamaan akan kedaluwarsa bersamaan setiap tahun. Menerbitkan ulang sebagian lebih awal, untuk memecah tumpukan, adalah investasi kecil yang mengurangi risiko besar.

# Bab 150. Penyaringan Lalu Lintas IP

z/OS Communications Server menyediakan IP filtering sebagai lapisan pertahanan tambahan di belakang firewall jaringan. Aturan filter mengizinkan hanya alamat dan port yang memang dibutuhkan.

| Zona | Siapa boleh masuk | Port contoh |
|---|---|---|
| Manajemen | Workstation sysprog dan operator lewat jump host | TN3270 TLS, SSH, z/OSMF |
| Aplikasi | Payment hub, API gateway, server aplikasi | Listener CICS, z/OS Connect, Db2 DDF, MQ |
| Replikasi dan DR | Sistem di situs sekunder | Port replikasi dan monitoring |
| Lainnya | Tidak ada | Semua ditolak secara default |

Prinsip yang dipakai sama dengan firewall mana pun: tolak secara default, izinkan secara eksplisit, catat penolakan, dan tinjau aturan secara berkala.

## 150.1 Analisis: Merancang Aturan Penyaringan yang Bisa Dirawat

Aturan penyaringan IP di z/OS dievaluasi berurutan, dan aturan pertama yang cocok menentukan hasilnya. Seiring waktu, daftar aturan cenderung tumbuh: setiap mitra baru, setiap aplikasi baru, setiap pengecualian sementara menambah satu atau dua baris. Tanpa disiplin, daftar ini berubah menjadi ratusan baris yang tidak dipahami siapa pun.

| Prinsip | Penerapan |
|---|---|
| Tolak kecuali diizinkan | Aturan terakhir menolak semua lalu lintas yang tidak cocok dengan aturan sebelumnya |
| Satu aturan satu tujuan | Setiap aturan punya komentar: aplikasi, pemilik, tiket, tanggal |
| Kelompokkan per zona | Mitra, kanal internal, manajemen, dan replikasi DR di blok terpisah |
| Aturan paling spesifik di atas | Pengecualian sempit ditempatkan sebelum aturan umum |
| Catat penolakan | Lalu lintas yang ditolak dicatat untuk |

| Prinsip | Penerapan |
|---|---|
| | investigasi dan penyetelan |
| Tinjau berkala | Aturan tanpa lalu lintas selama beberapa bulan diusulkan untuk dihapus |

Peninjauan berkala adalah bagian yang paling sering terlewat. Aturan “sementara” untuk migrasi dua tahun lalu masih ada, dan membuka port ke alamat yang kini dipakai sistem lain. Statistik hit per aturan menunjukkan aturan mana yang tidak pernah dipakai. Setiap aturan yang tidak dipakai adalah permukaan serangan yang tidak memberi manfaat apa pun.

```
ALGORITMA TinjauAturanFilter(aturan_list,
statistik_90_hari)
  untuk setiap aturan a:
      jika a tidak punya komentar pemilik atau tiket maka
tandai "TANPA PEMILIK"
      jika hit(a) = 0 selama 90 hari maka tandai "KANDIDAT
HAPUS"
      jika a membuka rentang alamat atau port yang sangat
luas maka tandai "TERLALU LUAS"
      jika a tertutup sepenuhnya oleh aturan lain di
atasnya maka tandai "TIDAK TERJANGKAU"
  kirim daftar kepada pemilik masing-masing; hapus setelah
konfirmasi atau tenggat
```

Ubah aturan lewat tiket perubahan dan uji di lingkungan non-produksi dengan profil yang sama. Kesalahan urutan satu baris saja dapat memutus kanal pembayaran seluruh bank.

# Bab 151. Runbook Transport

## RB-055: Handshake TLS gagal setelah perubahan kebijakan

- **Gejala:** mitra tidak bisa terhubung setelah AT-TLS diperbarui; pesan kegagalan handshake di syslog.
- **Diagnosis:** `D TCPIP,,NETSTAT,TTLS,CONNDETAIL` untuk koneksi yang berhasil; bandingkan versi TLS dan cipher yang didukung mitra.

- **Tindakan:** kembalikan kebijakan sebelumnya dan F `PAGENT,REFRESH`; koordinasikan versi dan cipher minimum dengan mitra sebelum mencoba lagi.
- **Pencegahan:** uji perubahan kebijakan di non-produksi dengan klien setiap mitra.

## RB-056: Rantai sertifikat tidak lengkap

- **Gejala:** sebagian klien berhasil, sebagian menolak sertifikat server.
- **Diagnosis:** `RACDCERT ... LISTRING` untuk memastikan sertifikat CA perantara tersambung ke keyring.
- **Tindakan:** sambungkan CA perantara ke keyring; refresh RACLIST; uji ulang dari klien yang gagal.
- **Pencegahan:** checklist pembaruan sertifikat mencakup verifikasi rantai penuh dari sisi klien.

# 🧬 Bagian XXXVI — Db2 Lanjutan

Bagian XVII membahas operasi Db2 sehari-hari. Bagian ini membahas fitur yang menentukan apakah database core banking tetap cepat setelah sepuluh tahun data menumpuk.

## Bab 152. Partisi Tabel

Tabel riwayat transaksi tumbuh tanpa henti. Partisi berdasarkan rentang tanggal membuat data lama bisa diarsip, di-REORG, dan dipulihkan per partisi, tanpa menyentuh data yang sedang aktif.

| Jenis universal table space | Cara membagi | Cocok untuk |
|---|---|---|
| Partition-by-range (PBR) | Nilai kolom kunci, misalnya tanggal | Riwayat transaksi, jurnal GL |
| Partition-by-growth (PBG) | Otomatis saat partisi penuh | Tabel sedang tanpa pola arsip |

```
CREATE TABLE BANK.TRX_HIST
( TGL_BISNIS  DATE          NOT NULL,
  NO_REK      CHAR(16)      NOT NULL,
  NO_URUT     BIGINT        NOT NULL,
  NOMINAL     DECIMAL(17,2) NOT NULL,
  DK          CHAR(1)       NOT NULL,
  KODE_TRX    CHAR(4)       NOT NULL )
  PARTITION BY RANGE (TGL_BISNIS)
  ( PARTITION 1 ENDING AT ('2026-01-31'),
    PARTITION 2 ENDING AT ('2026-02-28'),
    PARTITION 3 ENDING AT ('2026-03-31') )
  IN BANKDB.TSTRXH;

-- Setiap awal bulan: partisi tertua dikosongkan dan
dipakai ulang untuk bulan baru
ALTER TABLE BANK.TRX_HIST
  ROTATE PARTITION FIRST TO LAST
  ENDING AT ('2026-04-30') RESET;
```

Sebelum rotasi, pastikan data partisi tertua sudah diarsip dan diverifikasi. RESET menghapus isi partisi tersebut.

## 152.1 Analisis: Menentukan Ukuran dan Jumlah Partisi

Partisi yang dirancang dengan baik membuat tabel riwayat transaksi yang tumbuh tanpa batas tetap mudah dikelola. Rancangan yang buruk menghasilkan partisi yang terlalu besar untuk di-REORG dalam window, atau terlalu banyak partisi kecil yang merepotkan. Perhitungannya sederhana dan sebaiknya dilakukan sebelum tabel dibuat.

$$N_{\text{partisi}} = \frac{D_{\text{retensi}}}{D_{\text{per partisi}}} \qquad V_{\text{partisi}} = N_{\text{baris per periode}} \times s_{\text{baris}} \times (1 + f_{\text{index dan ruang bebas}})$$

Contoh: tabel riwayat transaksi menyimpan 60 bulan data, dipartisi per bulan, sehingga ada 60 partisi aktif. Setiap bulan ada sekitar 30 juta baris dengan rata-rata 200 byte. Dengan tambahan 30% untuk ruang bebas, satu partisi sekitar 30 juta × 200 × 1,3 ≈ 7,8 GB. `DSSIZE` 16 GB memberi ruang untuk pertumbuhan volume beberapa tahun.

| Pilihan periode | Kelebihan | Kekurangan |
|---|---|---|
| Harian | Partisi kecil, REORG cepat, pembuangan data harian presisi | Ribuan partisi untuk retensi panjang; overhead pengelolaan |
| Bulanan | Keseimbangan yang umum untuk data transaksi | Partisi bulan berjalan menjadi “panas” |
| Tahunan | Partisi sedikit | Partisi terlalu besar; REORG dan recovery lama |

Dua aturan sering terlupa. Pertama, kunci partisi harus sesuai dengan cara data dibuang: jika retensi dihitung per bulan, partisi per bulan memungkinkan data lama dibuang dengan `ROTATE PARTITION` tanpa `DELETE` massal. Kedua, query online harus menyertakan predikat pada kolom kunci partisi, agar Db2 hanya membaca partisi yang relevan. Query tanpa predikat itu akan memindai seluruh 60 partisi.

Uji rencana partisi dengan volume data setara produksi di lingkungan uji: berapa lama REORG satu partisi, berapa lama `ROTATE`, dan apakah query utama benar-benar hanya menyentuh partisi yang diharapkan (lihat `EXPLAIN`, Bab 33.4).

# Bab 153. Materialized Query Table

MQT menyimpan hasil query agregat sebagai tabel fisik. Laporan harian yang menjumlahkan jutaan transaksi bisa membaca ringkasan yang sudah dihitung, bukan tabel detail.

```
CREATE TABLE BANK.MQT_RINGKAS_HARIAN AS
( SELECT TGL_BISNIS, KODE_TRX, DK,
         COUNT(*)     AS JML,
         SUM(NOMINAL) AS TOTAL
    FROM BANK.TRX_HIST
   GROUP BY TGL_BISNIS, KODE_TRX, DK )
  DATA INITIALLY DEFERRED
  REFRESH DEFERRED
  MAINTAINED BY SYSTEM
  ENABLE QUERY OPTIMIZATION;

REFRESH TABLE BANK.MQT_RINGKAS_HARIAN;   -- dijalankan
sebagai fase pasca-EOD
```

Dengan `ENABLE QUERY OPTIMIZATION`, optimizer dapat memakai MQT secara otomatis untuk query yang cocok, tergantung register khusus seperti `CURRENT REFRESH AGE`. MQT yang tidak di-refresh berarti laporan membaca data lama, jadi jadwalkan `REFRESH` di tabel kontrol EOD.

## 153.1 Analisis: Kapan MQT Menguntungkan

Materialized Query Table menyimpan hasil kueri agregat yang mahal, sehingga kueri berikutnya bisa membaca hasil yang sudah jadi. Manfaatnya nyata, tetapi ada biaya: MQT harus diperbarui (refresh), dan datanya hanya sesegar refresh terakhir.

$$\text{Hemat bersih per hari} = Q \times (c_{\text{tanpa MQT}} - c_{\text{dengan MQT}}) - R \times c_{\text{refresh}}$$

Q adalah jumlah kueri per hari yang dapat memakai MQT, c biaya CPU per kueri, R jumlah refresh per hari, dan c refresh biaya satu kali refresh.

Contoh: kueri dashboard ringkasan saldo per cabang dijalankan 2.000 kali sehari, masing-masing memakan 4 detik CPU. Dengan MQT, biayanya turun menjadi 0,05 detik. Refresh memakan 300 detik CPU dan dilakukan empat kali sehari. Hemat bersihnya sekitar 2.000 × 3,95 − 4 × 300 = 6.700

detik CPU per hari, hampir dua jam CPU. MQT jelas menguntungkan dalam kasus ini.

Sebaliknya, kueri yang hanya dijalankan beberapa kali sehari tidak sebanding dengan biaya refresh. MQT dengan refresh terlalu sering untuk data yang berubah cepat bisa lebih mahal daripada menjalankan kueri aslinya.

| Pertanyaan | Bila jawabannya ya |
|---|---|
| Apakah kueri agregat yang sama dijalankan sangat sering? | Kandidat kuat MQT |
| Apakah pengguna menerima data yang tertinggal beberapa jam? | MQT dengan refresh terjadwal cocok |
| Apakah data harus selalu terkini hingga detik terakhir? | MQT kurang cocok; pertimbangkan index atau desain lain |
| Apakah refresh bisa dijalankan di luar jam sibuk? | Biaya refresh tidak bersaing dengan transaksi online |

Optimizer Db2 hanya memakai MQT yang di-refresh tertunda bila register `CURRENT REFRESH AGE` mengizinkan. Pastikan pengaturannya sesuai dengan kebutuhan kesegaran data, lalu buktikan lewat `EXPLAIN` bahwa kueri benar-benar diarahkan ke MQT.

## Bab 154. Objek Besar (LOB)

LOB menyimpan data besar seperti e-statement PDF atau hasil pindai dokumen. Db2 menaruh LOB di LOB table space terpisah, kecuali bagian kecil yang disimpan inline di baris.

| Pertimbangan | Rekomendasi |
|---|---|
| Ukuran total LOB | Proyeksikan pertumbuhan; LOB cepat mendominasi kapasitas |
| Backup dan recovery | Sertakan LOB table space dalam set recovery tabel induknya |
| Logging | Pertimbangkan dampak log untuk insert LOB besar |
| Alternatif | Dokumen besar yang jarang diakses sering |

| Pertimbangan | Rekomendasi |
|---|---|
| | lebih murah disimpan di object storage, dengan referensi di Db2 |

## 154.1 Analisis: Merencanakan Penyimpanan LOB

Semakin banyak dokumen nasabah disimpan secara digital: hasil pindai KTP, formulir pembukaan rekening, tanda tangan, dan dokumen kredit. Menyimpannya sebagai LOB di Db2 menjaga dokumen tetap terikat pada data nasabah dan ikut dalam backup dan pemulihan yang sama. Pilihan ini perlu dihitung, karena volumenya bisa sangat besar.

$$V_{\text{LOB}} = N_{\text{nasabah}} \times \bar{d} \times \bar{s}_{\text{dokumen}}$$

d adalah rata-rata jumlah dokumen per nasabah. Contoh: 5 juta nasabah dengan rata-rata 3 dokumen berukuran 400 KB menghasilkan sekitar 6 TB, belum termasuk pertumbuhan dan salinan backup. Angka ini jauh lebih besar dari seluruh data transaksi banyak bank.

| Keputusan | Pilihan | Pertimbangan |
|---|---|---|
| Di mana dokumen disimpan | LOB di Db2, atau penyimpanan objek dengan referensi di Db2 | Konsistensi dan kesederhanaan di Db2; biaya per TB lebih rendah di penyimpanan objek |
| LOB kecil | Inline LOB | Data kecil disimpan di baris utama, akses lebih cepat |
| Logging | Log penuh, atau tanpa log untuk LOB tertentu | Tanpa log menghemat log secara besar, tetapi pemulihan bergantung pada image copy |
| Kompresi | Kompresi di aplikasi atau storage | Pindaian gambar biasanya sudah terkompresi; manfaatnya kecil |
| Retensi | Mengikuti ketentuan penyimpanan dokumen | Dokumen nasabah yang sudah tutup rekening punya masa simpan tertentu |

Perhatikan dampaknya ke operasi lain. Image copy tablespace LOB berukuran terabyte memakan waktu dan ruang. REORG menjadi pekerjaan besar. Replikasi ke DR membawa volume yang sama. Karena itu banyak arsitektur memilih pendekatan campuran: metadata dan dokumen kecil yang sering diakses disimpan di Db2, sedangkan pindaian besar disimpan di penyimpanan objek yang dilindungi, dengan kunci referensi di Db2.

Apa pun pilihannya, pastikan dokumen ikut dalam rencana pemulihan dan uji DR. Nasabah dan regulator akan mempertanyakan dokumen yang hilang, bukan hanya saldo.

# Bab 155. Kendali Sumber Daya Kueri

Satu query analitik yang ditulis buruk dari aplikasi pelaporan dapat menghabiskan CPU yang seharusnya melayani transaksi. Db2 menyediakan beberapa lapis pembatasan.

| Mekanisme | Fungsi |
|---|---|
| Resource Limit Facility (RLF) | Membatasi waktu CPU per query dinamis berdasarkan tabel batas |
| Service class WLM untuk DDF | Memberi prioritas rendah pada beban analitik dari luar |
| Profile table Db2 | Memantau atau membatasi thread dan koneksi per aplikasi atau alamat |
| Timeout thread idle | Membersihkan koneksi DDF yang ditinggalkan |

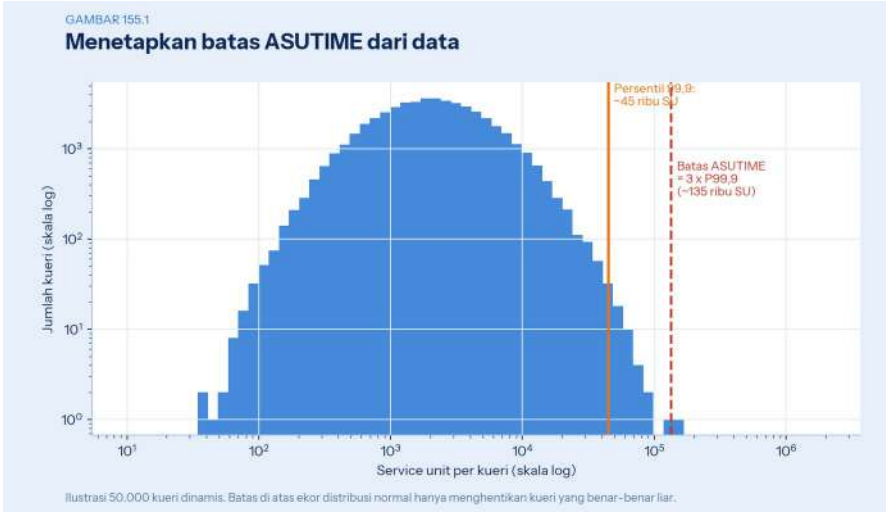
```
-START RLIMIT ID=01       Aktifkan tabel batas DSNRLST01
-DIS RLIMIT               Status RLF
-START PROFILE            Muat ulang profile table
```

Pisahkan beban analitik dari OLTP secara arsitektur bila memungkinkan: replikasi ke platform analitik (Bab 56) lebih aman daripada membatasi query di database transaksi.

## 155.1 Analisis: Menetapkan Batas ASUTIME dari Data

Resource Limit Facility melindungi Db2 dari kueri dinamis yang liar, misalnya kueri ad hoc tanpa predikat yang memindai tabel riwayat

bertahun-tahun di jam sibuk. Batas yang terlalu ketat menghentikan pekerjaan sah. Batas yang terlalu longgar tidak melindungi apa-apa. Batas yang tepat ditetapkan dari distribusi pemakaian nyata.




*Gambar 155.1 — Menetapkan batas ASUTIME dari data*

$$\text{ASUTIME} = k \times P_{99,9}(\text{service unit per kueri})$$

Dalam ilustrasi, 99,9% kueri dinamis memakai kurang dari sekitar 45 ribu service unit. Dengan faktor k = 3, batasnya sekitar 135 ribu service unit. Batas ini tidak menyentuh pekerjaan normal, tetapi menghentikan kueri yang pemakaiannya jauh di luar pola.

| Kelompok pengguna | Pendekatan batas |
|---|---|
| Aplikasi online lewat DDF | Ketat; kueri online seharusnya ringan dan dapat diprediksi |
| Pengguna analitik ad hoc | Sedang; dipisah per jam sibuk dan di luar jam sibuk |
| Batch terjadwal | Biasanya tanpa batas RLF, dikendalikan lewat WLM dan jadwal |
| Pengguna teknis (DBA) | Longgar, tetapi diaudit |

Kueri yang melewati batas dihentikan dengan `SQLCODE -905`. Pastikan aplikasi dan pengguna memahami arti kode ini, dan sediakan jalur resmi

untuk kueri berat yang sah, misalnya dijadwalkan di luar jam sibuk atau dijalankan di salinan data untuk analitik.

Tinjau distribusi setiap kuartal. Pertumbuhan data membuat kueri normal perlahan menjadi lebih mahal, dan batas yang dulu tepat bisa mulai menghentikan pekerjaan sah. Setiap kali `-905` terjadi, catat kuerinya. Riwayat ini menunjukkan apakah batas perlu disesuaikan atau kuerinya yang perlu diperbaiki.

## Bab 156. REORG Berbasis Real-Time Statistics

Menjalankan REORG untuk semua objek setiap pekan membuang CPU. Real-Time Statistics (RTS) mencatat perubahan sejak REORG terakhir, sehingga REORG hanya dijalankan untuk objek yang membutuhkannya.

```
SELECT DBNAME, NAME, PARTITION, TOTALROWS,
       REORGINSERTS, REORGDELETES, REORGUNCLUSTINS,
       DECIMAL(100.0 * (REORGINSERTS + REORGDELETES)
               / NULLIF(TOTALROWS, 0), 7, 2) AS PCT_UBAH
  FROM SYSIBM.SYSTABLESPACESTATS
 WHERE DBNAME = 'BANKDB'
   AND TOTALROWS > 100000
 ORDER BY PCT_UBAH DESC
 FETCH FIRST 20 ROWS ONLY;
```

$$\text{kandidat REORG jika } \frac{\text{REORGINSERTS} + \text{REORGDELETES}}{\text{TOTALROWS}} > 10\% \text{ atau } \frac{\text{REORGUNCLUSTINS}}{\text{TOTALROWS}} > 10\%$$

Ambang 10% adalah titik awal yang umum. Db2 juga menyediakan stored procedure `DSNACCOX` yang memberi rekomendasi utilitas berdasarkan RTS, dan hasilnya bisa dipakai untuk membangun job REORG dinamis.

### 156.1 Analisis: Merencanakan Window REORG

Real-Time Statistics memberi tahu objek mana yang perlu di-REORG. Pertanyaan berikutnya adalah kapan dan dengan cara apa, agar REORG tidak mengganggu transaksi online maupun batch EOD.

$$T_{\text{REORG}} \approx \frac{V_{\text{objek}}}{\text{laju unload} \;+\; \text{reload} \;+\; \text{rebuild index}} + T_{\text{log apply}} + T_{\text{switch}}$$

Laju ini paling baik diukur dari REORG sebelumnya untuk objek yang serupa, lalu dipakai untuk memperkirakan objek lain. Dengan SHRLEVEL CHANGE, aplikasi tetap bisa membaca dan menulis selama sebagian besar proses. Namun di akhir ada fase penerapan log dan pengalihan ke salinan baru, yang membutuhkan akses eksklusif sebentar (drain). Jika ada transaksi panjang yang memegang objek, drain bisa gagal atau menunggu.

| Pilihan | Ketersediaan selama REORG | Cocok untuk |
|---|---|---|
| SHRLEVEL NONE | Objek tidak tersedia | Objek kecil di window pemeliharaan |
| SHRLEVEL REFERENCE | Hanya baca | Objek yang jarang ditulis |
| SHRLEVEL CHANGE | Baca dan tulis, dengan drain singkat di akhir | Objek online kritikal |

**Aturan penjadwalan:**

- Jangan jalankan REORG objek yang dipakai fase EOD di dalam window EOD (kasus K-05).
- Untuk tabel berpartisi, REORG satu partisi pada satu waktu, dimulai dari partisi yang paling membutuhkan menurut RTS (Bab 152.1).
- Jadwalkan fase drain di jam dengan transaksi paling sedikit, dan tetapkan batas waktu tunggu drain agar REORG menyerah dengan aman daripada menahan transaksi.
- Sediakan ruang untuk salinan bayangan. REORG SHRLEVEL CHANGE membutuhkan ruang tambahan kira-kira sebesar objek yang di-REORG.

Catat durasi dan hasil setiap REORG. Riwayat ini memperbaiki perkiraan untuk REORG berikutnya dan menunjukkan objek yang pertumbuhannya membuat REORG semakin lama, yang merupakan kandidat untuk partisi ulang atau pengarsipan.

# ✍️ Bagian XXXVII — Esai Teknis

Tiga esai pendek ini bukan prosedur. Isinya cara berpikir yang saya harap tertinggal setelah detail perintah terlupakan.

## Bab 157. Mengapa Integritas Dibangun ke Dalam, Bukan Ditambahkan

Mainframe bertahan puluhan tahun di perbankan bukan karena cepat, tetapi karena setiap lapisannya dirancang untuk tidak kehilangan satu rupiah pun. Log Db2 ditulis sebelum commit dikonfirmasi. Two-phase commit menyatukan Db2, MQ, dan CICS dalam satu unit kerja. Coupling Facility memastikan dua sistem tidak memperbarui record yang sama bersamaan.

Integritas semacam ini tidak bisa ditambahkan belakangan. Aplikasi yang dirancang tanpa idempotensi tidak menjadi aman hanya karena dipasang di mainframe. Fase EOD yang tidak bisa diulang tidak menjadi aman hanya karena ada backup.

Pelajarannya bagi system engineer: ketika meninjau desain baru, tanyakan dulu “apa yang terjadi jika langkah ini dijalankan dua kali, atau berhenti di tengah?” sebelum bertanya “seberapa cepat?”. Pertanyaan pertama menentukan apakah sistem bisa dipercaya; pertanyaan kedua hanya menentukan berapa biayanya.

## Bab 158. Ekonomi Keandalan

Setiap angka sembilan tambahan pada target ketersediaan memangkas downtime yang diizinkan sepuluh kali lipat, dan biasanya melipatgandakan biaya untuk mencapainya.


*Dihitung dari 525.600 menit per tahun*

Ketersediaan dapat didekati dari rata-rata waktu antar-kegagalan dan rata-rata waktu pemulihan:

$$A = \frac{MTBF}{MTBF + MTTR}$$

Rumus ini menunjukkan dua jalan: membuat kegagalan lebih jarang, atau membuat pemulihan lebih cepat. Di mainframe, MTBF hardware sudah sangat tinggi, sehingga investasi terbaik biasanya ada di sisi MTTR: runbook yang teruji, otomasi, dan wewenang keputusan yang jelas.

Biaya satu jam gangguan bagi bank dapat dirumuskan sebagai:

$$C_{\text{gangguan}} = T \times (R_{\text{hilang/jam}} + K_{\text{operasional/jam}}) + K_{\text{pemulihan}} + K_{\text{reputasi dan regulasi}}$$

Komponen terakhir paling sulit diukur tetapi sering paling besar. Gangguan layanan perbankan yang lama bisa berujung pemeriksaan regulator dan hilangnya kepercayaan nasabah yang tidak tercermin di laporan keuangan bulan itu.

Keputusan investasi keandalan yang baik membandingkan biaya perbaikan dengan penurunan biaya gangguan yang diharapkan, bukan mengejar angka sembilan demi angka itu sendiri.

# Bab 159. Tentang Dokumentasi yang Sungguh Dipakai

Hampir setiap tim punya dokumentasi. Sedikit yang punya dokumentasi yang dibuka pada pukul tiga pagi.

Dokumentasi yang dipakai punya tiga ciri. Pertama, ditulis untuk orang yang sedang panik: langkah pendek, perintah yang bisa disalin, dan hasil yang diharapkan di setiap langkah. Kedua, bisa ditemukan dalam sepuluh detik lewat nama gejala, bukan nama komponen. Ketiga, pernah diuji oleh orang selain penulisnya.

Buku ini mencoba mengikuti ketiga prinsip itu lewat runbook bertemplat, indeks per kategori, dan praktikum. Namun buku tetaplah titik awal. Dokumentasi yang paling berharga adalah yang ditulis tim Anda sendiri, setelah insiden Anda sendiri, dalam bahasa yang tim Anda pakai sehari-hari.

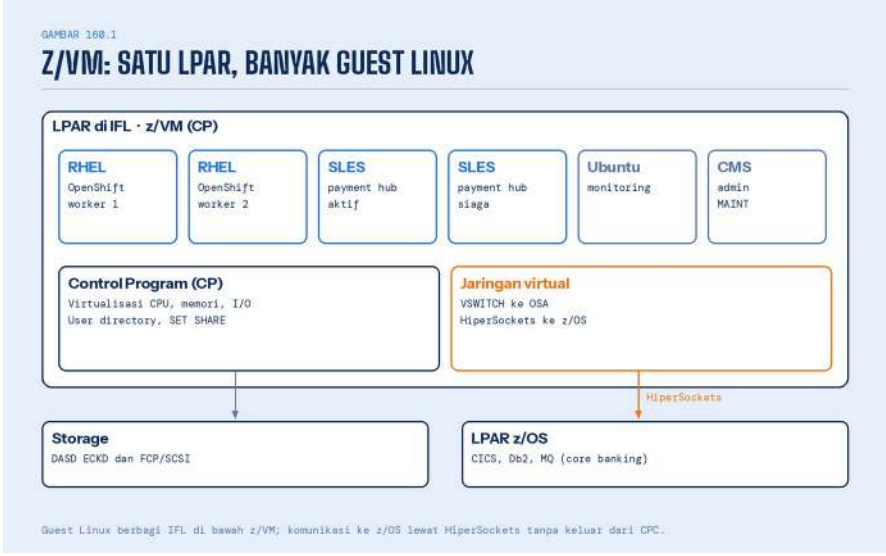
Jika setelah membaca buku ini Anda menulis satu runbook baru dan mengujinya bersama rekan, buku ini sudah berhasil.

# 🐧 Bagian XXXVIII — z/VM dan Linux on IBM Z

Banyak bank menjalankan payment hub, middleware, dan layanan mikro di Linux on IBM Z, berdampingan dengan z/OS di mesin yang sama. System engineer z/OS perlu memahami lingkungan ini cukup dalam untuk berkolaborasi dan mendiagnosis masalah di perbatasannya.

## Bab 160. Arsitektur z/VM

z/VM adalah hypervisor IBM Z yang menjalankan ratusan mesin virtual (guest) di satu LPAR. Komponen utamanya adalah CP (*Control Program*) yang memvirtualisasi hardware, dan CMS sebagai lingkungan administrasi ringan.




*Gambar 160.1 — z/VM: satu LPAR, banyak guest Linux*

Setiap guest didefinisikan di user directory. Contoh entri untuk guest Linux produksi:

```
USER LNXPH01 ******** 8G 16G G
  INCLUDE LNXDFLT
```

```
CPU 00 BASE
CPU 01
IPL 0200
NICDEF 0600 TYPE QDIO LAN SYSTEM VSWPROD
MDISK 0200 3390 0001 10016 LXP001 MR
MDISK 0201 3390 0001 10016 LXP002 MR
```

Kata sandi ditampilkan sebagai placeholder. Di lingkungan produksi, otentikasi guest sebaiknya dikelola external security manager seperti RACF for z/VM, bukan kata sandi di directory.

| Perintah CP | Fungsi |
|---|---|
| `QUERY NAMES` | Guest yang sedang login |
| `INDICATE LOAD` | Beban CPU dan paging sistem |
| `INDICATE USER LNXPH01` | Pemakaian sumber daya satu guest |
| `QUERY ALLOC PAGE` | Pemakaian ruang paging |
| `QUERY VSWITCH DETAILS` | Status virtual switch dan uplink OSA |
| `SET SHARE LNXPH01 RELATIVE 300` | Menaikkan porsi CPU relatif guest |

## 160.1 Analisis: Metrik Kesehatan z/VM

z/VM menjalankan banyak guest di atas sumber daya bersama. Kesehatannya tidak cukup dilihat dari CPU saja. Empat kelompok metrik memberi gambaran yang lengkap.

| Metrik | Sumber | Tanda masalah | Tindakan |
|---|---|---|---|
| Pemakaian prosesor (IFL) per LPAR | `INDICATE LOAD`, Performance Toolkit | Mendekati jenuh di jam puncak | Tinjau prioritas guest (SHARE), tambah IFL |
| Laju paging z/VM | Performance Toolkit | Paging tinggi dan terus-menerus | Kurangi memori guest yang berlebihan (Bab 162.1) |
| Pemakaian page space | `QUERY ALLOC PAGE` | Terisi lebih dari sekitar separuh | Tambah page space; paging blok z/VM paling efisien bila page space longgar |
| Keadaan tunggu | Performance Toolkit | Guest sering | Tinjau overcommit |

| Metrik | Sumber | Tanda masalah | Tindakan |
|---|---|---|---|
| guest | | menunggu CPU atau halaman | dan SHARE |

Page space z/VM perlu perhatian khusus. z/VM menulis halaman ke page space dalam blok, dan cara ini paling efisien bila tersedia banyak ruang kosong yang berdekatan. Karena itu page space umumnya dijaga jauh dari penuh. Page space yang penuh dapat membuat seluruh sistem z/VM berhenti, dan semua guest di atasnya ikut berhenti.

$$\text{Rasio overcommit memori} = \frac{\sum \text{memori virtual semua guest aktif}}{\text{memori nyata LPAR}}$$

Rasio overcommit yang sehat bergantung pada pola beban (Bab 163). Guest yang sibuk bersamaan membutuhkan rasio yang lebih rendah daripada guest yang sibuk di waktu berbeda. Pantau rasio ini bersama laju paging. Rasio yang sama bisa sehat di satu lingkungan dan bermasalah di lingkungan lain.

Kumpulkan metrik z/VM ke platform pemantauan yang sama dengan z/OS (Bab 183), agar masalah di guest Linux dapat dikorelasikan dengan kondisi z/VM dan LPAR di bawahnya.

## Bab 161. SSI dan Live Guest Relocation

Single System Image (SSI) menggabungkan beberapa sistem z/VM menjadi satu cluster yang berbagi directory dan sumber daya. Live Guest Relocation (LGR) memindahkan guest Linux yang sedang berjalan ke anggota cluster lain tanpa menghentikannya.

```
VMRELOCATE TEST LNXPH01 TO VMB    Periksa kelayakan relokasi
VMRELOCATE MOVE LNXPH01 TO VMB    Pindahkan guest yang sedang berjalan
QUERY SSI                         Status anggota cluster
```

LGR memungkinkan pemeliharaan z/VM atau hardware tanpa downtime aplikasi Linux, mirip rolling IPL di Parallel Sysplex.

## 161.1 Analisis: Merencanakan Live Guest Relocation

Live Guest Relocation memindahkan guest Linux yang sedang berjalan dari satu anggota z/VM SSI ke anggota lain, tanpa menghentikan aplikasi. Kemampuan ini membuat pemeliharaan z/VM tidak lagi berarti gangguan layanan. Namun relokasi hanya berjalan lancar bila guest memenuhi syarat dan waktunya dipilih dengan tepat.

| Langkah | Perintah atau pemeriksaan | Tujuan |
|---|---|---|
| Uji kelayakan | `VMRELOCATE TEST guest TO anggota` | Menemukan hambatan: device yang tidak tersedia di tujuan, kapasitas memori, dan sebagainya |
| Periksa kapasitas tujuan | Memori dan prosesor anggota tujuan | Guest yang pindah tidak membuat anggota tujuan kewalahan |
| Pilih waktu | Jam dengan perubahan memori guest yang rendah | Relokasi lebih cepat dan jeda akhir lebih singkat |
| Relokasi | `VMRELOCATE MOVE guest TO anggota` | Pemindahan sebenarnya |
| Verifikasi | Aplikasi dan koneksi guest normal | Memastikan tidak ada efek samping |

Lama relokasi terutama ditentukan oleh ukuran memori guest dan seberapa cepat memori itu berubah. Memori disalin ke anggota tujuan dalam beberapa putaran, dan halaman yang berubah selama penyalinan harus disalin lagi. Guest dengan beban tulis memori yang sangat tinggi membutuhkan lebih banyak putaran. Di akhir proses, guest dihentikan sejenak untuk menyalin sisa halaman. Jeda ini biasanya singkat, tetapi bisa terasa oleh aplikasi yang sangat sensitif terhadap latensi.

Untuk pemeliharaan satu anggota SSI, relokasikan guest secara bertahap, bukan sekaligus. Mulailah dengan guest yang paling tidak kritikal, amati dampaknya, lalu lanjutkan. Guest yang tidak memenuhi syarat relokasi harus dijadwalkan untuk restart biasa dalam jendela pemeliharaan, dan daftar guest seperti ini sebaiknya diperiksa jauh sebelum hari pemeliharaan.

Relokasi juga berguna di luar pemeliharaan, misalnya untuk menyeimbangkan beban saat satu anggota terlalu sibuk. Namun relokasi yang terlalu sering menandakan kapasitas anggota yang tidak seimbang, yang lebih baik diperbaiki lewat perencanaan kapasitas (Bab 163).

# Bab 162. Linux on IBM Z dari Sudut Pandang Operasi

Linux on IBM Z memakai arsitektur `s390x`. Distribusi yang umum adalah Red Hat Enterprise Linux, SUSE Linux Enterprise Server, dan Ubuntu Server.

| Area | Perintah atau alat | Catatan |
|---|---|---|
| Disk ECKD | `lsdasd`, `dasdfmt`, `fdasd` | DASD yang sama jenisnya dengan z/OS |
| Disk SCSI lewat FCP | `lszfcp`, `zfcp` | Umum untuk kapasitas besar |
| Aktivasi device | `lszdev`, `chzdev` | Mengaktifkan device secara persisten |
| Jaringan | driver `qeth` | OSA dan HiperSockets |
| Perintah CP dari Linux | `vmcp` | Misalnya `vmcp query userid` |
| Boot | `zipl` | Boot loader untuk IBM Z |
| Kriptografi hardware | `lszcrypt` | Akses Crypto Express |

```
lsdasd                          # daftar DASD dan statusnya
chzdev -e dasd 0.0.0201         # aktifkan DASD 0201 secara persisten
vmcp query cpus                 # lihat CPU virtual dari dalam Linux
lszcrypt -V                     # status kartu kriptografi
```

## 162.1 Analisis: Ukuran Guest dan Pemeliharaan Linux on Z

Kesalahan paling umum dalam mengoperasikan Linux on Z adalah memberi setiap guest memori sebanyak mungkin “agar aman”. Di server fisik biasa, kebiasaan ini tidak merugikan. Di z/VM, kebiasaan ini justru merusak efisiensi konsolidasi.

Linux memakai memori yang tidak terpakai sebagai cache file. Dari sisi Linux, memori terlihat hampir penuh. Dari sisi z/VM, setiap halaman yang disentuh guest harus dilayani dengan memori nyata. Guest yang diberi 32 GB padahal hanya membutuhkan 8 GB akan mengisi sisanya dengan cache, dan mengambil memori yang bisa dipakai guest lain.

| Pendekatan | Hasil |
|---|---|
| Memori guest besar “untuk jaga-jaga” | Paging z/VM naik, semua guest melambat |
| Memori guest sesuai kebutuhan kerja plus cadangan kecil | Overcommit sehat, kinerja stabil |
| Menambah memori saat dibutuhkan (hotplug, bila didukung) | Fleksibel tanpa pemborosan permanen |

Ukur kebutuhan nyata dari pemakaian aplikasi, bukan dari angka memori terpakai di Linux. Tanda guest terlalu kecil adalah swapping di dalam Linux. Tanda guest terlalu besar adalah cache file yang sangat besar dan paging tinggi di z/VM.

**Pemeliharaan.** Linux on Z mengikuti siklus patch distribusinya, yang jauh lebih sering daripada siklus RSU z/OS. Live Guest Relocation (Bab 161) memungkinkan pemeliharaan anggota z/VM tanpa menghentikan guest, tetapi patch kernel Linux sendiri tetap membutuhkan restart guest. Rencanakan jendela pemeliharaan Linux secara bergilir, sehingga layanan yang berjalan di beberapa guest tetap tersedia selama sebagian guest di-restart.

| Aktivitas | Frekuensi umum |
|---|---|
| Patch keamanan distribusi | Bulanan, atau segera untuk celah kritikal |
| Restart guest untuk kernel baru | Mengikuti patch kernel, bergilir |
| Pemeliharaan z/VM (RSU) | Beberapa kali setahun, dengan relokasi guest |
| Tinjauan ukuran guest | Setiap kuartal dari data pemakaian |

# Bab 163. Konsolidasi dan Kinerja

Nilai ekonomi Linux on IBM Z datang dari konsolidasi: banyak beban kecil berbagi sedikit IFL. Rasio overcommit perlu dijaga agar konsolidasi tidak berubah menjadi kontensi.

$$\text{Rasio overcommit CPU} = \frac{\sum \text{CPU virtual semua guest aktif}}{\text{IFL fisik}} \qquad \text{Rasio memori} = \frac{\sum \text{memori virtual guest}}{\text{memori LPAR}}$$

| Jenis beban | Kecocokan di Linux on IBM Z |
|---|---|
| Payment hub dan middleware dekat core | Sangat cocok: latensi rendah lewat HiperSockets |
| Database terbuka dengan data sensitif | Cocok: pervasive encryption dan isolasi LPAR |
| OpenShift untuk layanan mikro bank | Cocok, terutama yang memanggil core |
| Beban GPU atau analitik berat | Kurang cocok; lebih tepat di platform lain |

Rasio yang aman sangat bergantung pada pola beban. Mulailah konservatif di produksi, pantau dengan Performance Toolkit for z/VM, lalu naikkan bertahap berdasarkan data, bukan asumsi.

## 163.1 Analisis: Ekonomi Konsolidasi Linux ke IBM Z

Konsolidasi server Linux ke IFL sering dinilai dari satu sisi saja: harga hardware. Perhitungan yang lebih lengkap melihat tiga komponen biaya sekaligus, dan dua di antaranya sering lebih besar daripada hardware.

$$\text{Rasio konsolidasi} = \frac{\sum \text{core x86 terpasang}}{N_{\text{IFL}}} \qquad \text{Core efektif} = \sum(\text{core} \times \text{utilisasi rata} - \text{rata})$$

Banyak server x86 berjalan dengan utilisasi rendah, misalnya 10–20%, karena setiap aplikasi diberi server sendiri untuk isolasi. Dua ratus core dengan utilisasi rata-rata 15% hanya melakukan kerja setara sekitar 30 core yang sibuk penuh. z/VM dapat menjalankan banyak guest di atas IFL dengan utilisasi tinggi, karena beban guest yang puncaknya berbeda waktu saling mengisi.

| Komponen biaya | Pengaruh konsolidasi | Catatan |
|---|---|---|
| Hardware dan fasilitas | Lebih sedikit server fisik, | Perlu dibandingkan dengan |

| Komponen biaya | Pengaruh konsolidasi | Catatan |
|---|---|---|
| | ruang, dan energi | biaya IFL dan memori |
| Lisensi software per core | Sering turun besar karena jumlah core berkurang | Periksa ketentuan lisensi vendor untuk IBM Z |
| Operasi | Lebih sedikit server fisik untuk dikelola; tetapi butuh keahlian z/VM | Pelatihan dan perubahan proses |

Komponen lisensi per core sering menjadi faktor penentu. Software database atau middleware komersial yang dilisensikan per core bisa jauh lebih mahal daripada hardware-nya sendiri. Mengurangi jumlah core yang perlu dilisensikan memberi penghematan terbesar, dan besarnya bergantung sepenuhnya pada ketentuan lisensi masing-masing vendor.

Faktor kesetaraan antara core x86 dan IFL tidak bisa diambil dari tabel umum. Nilainya bergantung pada jenis beban, generasi prosesor di kedua sisi, dan pola utilisasi. Karena itu keputusan konsolidasi sebaiknya didasarkan pada uji coba dengan beban nyata, misalnya memindahkan beberapa aplikasi perwakilan dan mengukur pemakaian IFL-nya selama beberapa pekan, sebelum dibuat proyeksi untuk seluruh portofolio.

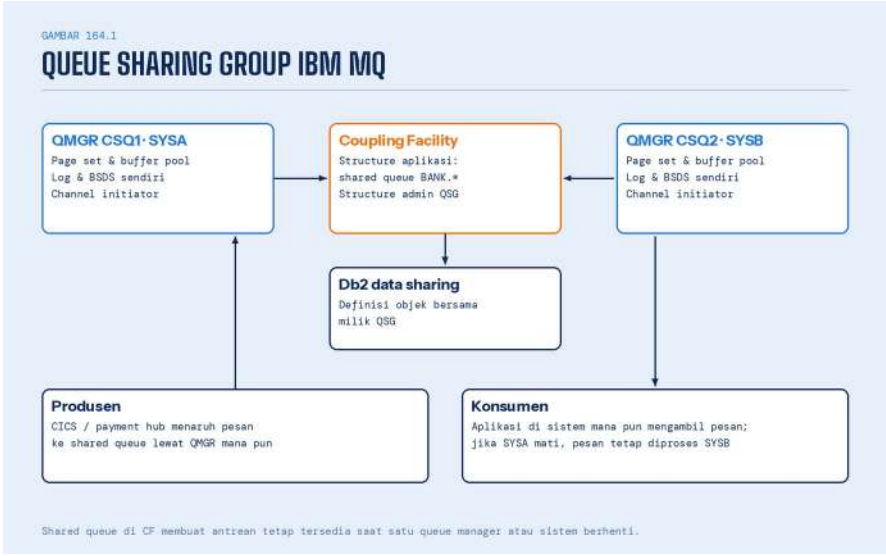
# 📨 Bagian XXXIX — IBM MQ dan IMS Mendalam

MQ adalah jalan raya pesan antara core banking dan dunia luar, sedangkan IMS masih menyimpan data inti di sebagian bank yang lebih tua. Keduanya jarang menjadi sorotan sampai berhenti, lalu tiba-tiba menjadi pusat insiden.

## Bab 164. Arsitektur Queue Manager di z/OS

Queue manager MQ di z/OS berjalan sebagai subsistem dengan dua address space: `xxxxMSTR` untuk inti queue manager dan `xxxxCHIN` untuk channel initiator.

| Komponen | Fungsi | Yang dipantau |
|---|---|---|
| Page set | Dataset VSAM tempat pesan disimpan | Persentase terpakai |
| Buffer pool | Cache pesan di memori | Rasio halaman yang harus ditulis ke page set |
| Storage class | Pemetaan queue ke page set | Queue besar terisolasi di page set sendiri |
| Log aktif, arsip, dan BSDS | Pemulihan pesan persisten | Offload log berjalan lancar |
| Channel initiator | Komunikasi dengan queue manager lain dan klien | Status channel, jumlah koneksi |




*Gambar 164.1 — Queue sharing group IBM MQ*

```
+CSQ1 DISPLAY USAGE PSID(*)                    Pemakaian page
set
+CSQ1 DISPLAY QSTATUS(BANK.*) TYPE(QUEUE) CURDEPTH IPPROCS
OPPROCS
+CSQ1 DISPLAY CHSTATUS(*) STATUS               Status semua
channel
+CSQ1 DISPLAY QMGR DEADQ                       Nama dead-
letter queue
+CSQ1 DISPLAY GROUP                            Anggota queue
sharing group
```

## 164.1 Analisis: Kapasitas Log dan Page Set Queue Manager

Queue manager MQ di z/OS menyimpan pesan persisten di page set atau CF structure, dan mencatat setiap perubahan pesan persisten di log. Keduanya harus diukur dari volume nyata. Log yang terlalu kecil memaksa offload archive terus-menerus. Page set yang terlalu kecil membuat antrean penuh sebelum MAXDEPTH tercapai.

$$V_{\text{log per hari}} \approx N_{\text{persisten}} \times (\bar{s}_{\text{pesan}} + o_{\text{put}} + o_{\text{get}}) \qquad V_{\text{page set}} \geq \text{MAXDEPTH} \times \bar{s}_{\text{pesan}} \times (1 + h)$$

Saat pesan persisten ditulis (put), isinya dicatat di log. Saat pesan diambil (get), log mencatat catatan yang jauh lebih kecil. o adalah overhead catatan log, dan h cadangan ruang.

Contoh: 5 juta pesan persisten per hari dengan ukuran rata-rata 2 KB menghasilkan sekitar 10 GB log per hari dari isi pesan saja, ditambah overhead. Active log harus cukup menampung beberapa jam aktivitas puncak, agar offload ke archive tidak tertinggal dan pemulihan dapat membaca dari active log yang lebih cepat.

| Komponen | Yang diukur | Tanda terlalu kecil |
|---|---|---|
| Active log | Laju penulisan log di jam puncak | Offload archive terus-menerus; peringatan log hampir penuh |
| Archive log | Retensi untuk pemulihan | Pemulihan page set tidak bisa mencapai titik yang dibutuhkan |
| Page set / structure | Kedalaman antrean terburuk × ukuran pesan | Error ruang penuh sebelum `MAXDEPTH` tercapai |
| Buffer pool | Pesan yang sering diakses dan antrean yang cepat bergerak | I/O page set tinggi, kinerja turun |

Pisahkan antrean dengan pola yang berbeda ke page set atau buffer pool berbeda, misalnya antrean transaksi real-time dan antrean pengiriman file batch. Antrean batch yang menumpuk berjuta pesan tidak boleh mendesak antrean real-time keluar dari buffer.

Pesan non-persisten tidak dicatat di log. Untuk data yang memang boleh hilang saat restart, misalnya notifikasi informasi, pesan non-persisten menghemat log secara besar. Keputusan persisten atau tidak harus diambil bersama pemilik aplikasi, bukan diubah diam-diam oleh tim infrastruktur.

## Bab 165. Channel dan Keandalan Pesan

| Jenis channel | Kegunaan |
|---|---|
| Sender / receiver | Antar-queue manager, satu arah |
| Server-connection (SVRCONN) | Aplikasi klien, misalnya payment hub |
| Cluster sender / receiver | Antar-anggota cluster MQ |

Keandalan pesan ditentukan oleh beberapa keputusan desain:

| Keputusan | Pilihan untuk transaksi keuangan |
|---|---|
| Persistensi | Pesan persisten, tercatat di log |
| Syncpoint | Put dan get dalam unit kerja bersama Db2 |
| Pesan racun | `BOTHRESH` dan `BOQNAME` untuk memindahkan pesan yang terus gagal diproses |
| Pesan tak terkirim | Dead-letter queue dengan handler `CSQUDLQH` yang terjadwal |
| Kedaluwarsa | Hanya untuk pesan informasi, tidak untuk instruksi keuangan |

## 165.1 Analisis: Throughput Channel MQ

Channel MQ mengirim pesan dalam batch. Setiap batch diakhiri sinkronisasi dengan pihak penerima: pesan dianggap aman hanya setelah penerima mengonfirmasi. Karena itu throughput channel jarak jauh sangat dipengaruhi waktu pulang-pergi jaringan dan ukuran batch.

$$\text{Throughput}_{\text{maks}} \approx \frac{\text{BATCHSZ}}{RTT + T_{\text{commit}} + T_{\text{transfer batch}}}$$

Contoh: RTT 20 ms dan waktu commit log di kedua sisi 5 ms. Dengan `BATCHSZ` 50 dan batch yang selalu terisi penuh, throughput maksimum sekitar 50 / 0,025 ≈ 2.000 pesan per detik. Jika setiap pesan dikirim sebagai batch tersendiri, throughput turun menjadi sekitar 40 pesan per detik, walaupun jaringan dan CPU masih lega.

Batch hanya terisi penuh bila pesan cukup banyak yang menunggu. Pada beban rendah, channel mengirim batch kecil agar pesan tidak tertahan. Parameter `BATCHINT` menentukan berapa lama channel menunggu untuk mengisi batch. Nilai yang lebih besar menaikkan throughput dengan sedikit latensi tambahan, sehingga cocok untuk aliran batch, tetapi tidak untuk pembayaran real-time.

| Parameter | Pengaruh | Pertimbangan |
|---|---|---|
| BATCHSZ | Jumlah pesan maksimum per sinkronisasi | Lebih besar menaikkan throughput; lebih banyak pesan dikirim ulang bila batch gagal |
| BATCHINT | Waktu menunggu untuk mengisi batch | Nol untuk latensi terendah; lebih besar untuk throughput |
| HBINT | Interval detak saat channel diam | Mendeteksi koneksi mati lebih cepat |
| DISCINT | Waktu channel diam sebelum diputus | Menghemat sumber daya untuk mitra yang jarang mengirim |

Untuk aliran bervolume tinggi ke situs jauh, pertimbangkan beberapa channel paralel, misalnya dipisah per jenis pesan. Satu channel dengan RTT tinggi akan selalu dibatasi rumus di atas, berapa pun kapasitas jaringannya.

## Bab 166. Keamanan MQ

MQ dilindungi RACF lewat class khusus, ditambah aturan channel di dalam MQ sendiri.

| Kontrol | Fungsi |
|---|---|
| Class MQCONN | Siapa boleh terhubung ke queue manager |
| Class MQQUEUE | Siapa boleh put atau get di queue tertentu |
| Class MQCMDS, MQADMIN | Siapa boleh memberi perintah dan mengelola objek |
| Aturan CHLAUTH | Memblokir atau memetakan koneksi berdasarkan alamat, user, atau sertifikat |
| TLS channel (SSLCIPH) | Enkripsi dan otentikasi antar-queue manager |

Channel SVRCONN tanpa aturan CHLAUTH dan tanpa TLS adalah celah yang sering ditemukan audit. Setiap SVRCONN produksi harus dibatasi pada alamat dan identitas yang diharapkan.

## 166.1 Analisis: Daftar Periksa Audit Keamanan MQ

MQ menghubungkan core banking dengan payment hub, switching, dan sistem mitra. Karena itu MQ adalah salah satu pintu masuk yang paling menarik bagi penyerang. Audit keamanan MQ secara berkala memastikan pintu itu dijaga sama ketatnya dengan pintu lain.

| Area | Kontrol | Cara memeriksa |
|---|---|---|
| Otentikasi channel | Channel authentication aktif; aturan default menolak koneksi yang tidak dikenal | Tampilkan aturan `CHLAUTH` dan pastikan ada aturan tolak umum |
| Identitas | `MCAUSER` tidak berprivilegi; pemetaan identitas lewat aturan `USERMAP` atau `SSLPEERMAP` | Tinjau definisi channel server dan receiver |
| Enkripsi | TLS pada semua channel ke luar LPAR; cipher sesuai kebijakan | Periksa `SSLCIPH` setiap channel |
| Otorisasi | Profil RACF di class `MQCONN`, `MQQUEUE`, `MQADMIN`, dan `MQCMDS` | Laporan akses per antrean produksi |
| Perintah administrasi | Hanya tim MQ yang boleh mengubah objek | Profil `MQCMDS` dan audit penggunaannya |
| Antrean sistem dan dead-letter | Akses dibatasi; isinya dipantau | Siapa yang dapat membaca dan menulis DLQ |

Dua temuan paling sering dalam audit MQ:

- **Channel tanpa TLS di jaringan internal**, dengan anggapan jaringan internal aman. Pesan pembayaran berisi nomor rekening dan nominal. Siapa pun yang bisa menyadap jaringan internal dapat membacanya.
- **`MCAUSER` kosong atau berprivilegi tinggi**, sehingga pesan dari mitra dimasukkan dengan hak akses yang terlalu luas. Satu mitra yang disusupi bisa menulis ke antrean yang seharusnya tidak dijangkaunya.

Periksa juga isi dead-letter queue secara teratur. Pesan yang berakhir di sana sering berasal dari percobaan koneksi yang tidak sah atau aplikasi yang salah konfigurasi. Keduanya perlu diketahui lebih awal.

Lakukan audit ini setiap tahun dan setiap kali ada mitra baru. Simpan hasilnya bersama checklist hardening z/OS (Bab 182) sebagai satu paket bukti keamanan.

## Bab 167. IMS untuk System Engineer z/OS

IMS terdiri dari dua bagian: IMS DB untuk database hierarkis dan IMS TM untuk transaksi. Keduanya dapat dipakai bersama atau terpisah, misalnya IMS DB diakses dari CICS.

| Komponen IMS | Fungsi |
|---|---|
| Full-function DB, HALDB | Database hierarkis umum, HALDB untuk ukuran sangat besar |
| Fast Path DEDB | Database berkinerja tinggi untuk volume transaksi besar |
| Region MPP / IFP | Menjalankan transaksi online |
| Region BMP | Batch yang berbagi database dengan online |
| DBRC dan RECON | Mencatat log, image copy, dan status database untuk recovery |
| OLDS, WADS, SLDS | Log online, write-ahead, dan log sistem arsip |

```
/DISPLAY ACTIVE                  Region dan transaksi
aktif
/DISPLAY DB CIFDB                Status database
/START DB CIFDB                  Mulai database yang
berhenti
/DBR DB CIFDB                    Keluarkan database dari
online untuk recovery
QUERY DB NAME(CIFDB) SHOW(ALL)   Perintah type-2 lewat
Operations Manager
```

Recovery IMS hampir selalu dijalankan lewat DBRC, yang tahu image copy dan log mana yang dibutuhkan. Jangan menyusun JCL recovery IMS secara manual jika DBRC bisa membuatkannya.

## RB-057: Dead-letter queue MQ menumpuk

- **Gejala:** kedalaman DLQ naik terus.
- **Diagnosis:** jelajahi pesan di DLQ untuk membaca header alasan (reason code) dan queue tujuan asli.
- **Tindakan:** perbaiki penyebab (queue penuh, queue tidak ada, izin kurang); jalankan DLQ handler dengan aturan yang sesuai untuk mengirim ulang.
- **Pencegahan:** DLQ handler terjadwal dan alert kedalaman DLQ di atas nol untuk queue kritikal.

## RB-058: Channel MQ berstatus RETRYING

- **Gejala:** pesan ke mitra menumpuk di transmission queue.
- **Diagnosis:** `DISPLAY CHSTATUS(nama)`; periksa jaringan, sertifikat TLS, dan status queue manager mitra.
- **Tindakan:** pulihkan penyebab; channel akan terhubung lagi otomatis sesuai interval retry, atau `START CHANNEL` manual.
- **Pencegahan:** pemantauan kedalaman transmission queue dan masa berlaku sertifikat channel.

## RB-059: Page set MQ hampir penuh

- **Gejala:** peringatan pemakaian page set tinggi; put gagal dengan alasan penyimpanan penuh.
- **Diagnosis:** `DISPLAY USAGE PSID(*)`; cari queue yang menumpuk di page set itu.
- **Tindakan:** pulihkan konsumen queue yang menumpuk; perluas page set bila dikonfigurasi dapat bertambah.
- **Pencegahan:** storage class terpisah untuk queue yang bisa menumpuk besar dan alert pemakaian page set.

## RB-060: Database IMS berhenti setelah BMP abend

- **Gejala:** transaksi yang memakai database gagal; `/DISPLAY DB` menunjukkan status berhenti.

- **Diagnosis:** baca pesan abend BMP dan status database di DBRC.
- **Tindakan:** bila IMS sudah melakukan backout dinamis dengan benar, `/START DB`; bila perlu batch backout, jalankan sesuai petunjuk DBRC sebelum database dibuka.
- **Pencegahan:** checkpoint berkala di program BMP dan prosedur restart BMP yang teruji.

## 167.1 Analisis: Waktu Pemulihan Database IMS

Pemulihan database IMS mengikuti pola yang sama dengan database lain: pulihkan salinan terakhir, lalu terapkan perubahan sejak salinan itu dibuat. DBRC mencatat semua bahan yang dibutuhkan, yaitu image copy, log, dan change accumulation, sehingga langkah pemulihan dapat dihasilkan secara otomatis.

$$T_{\text{pulih}} \approx T_{\text{restore image copy}} + T_{\text{apply perubahan}} \qquad T_{\text{apply}} \propto \text{volume log sejak image copy}$$

Semakin lama jarak sejak image copy terakhir, semakin banyak log yang harus diterapkan, dan semakin lama pemulihan. Change Accumulation membantu dengan meringkas perubahan dari banyak log menjadi satu set yang hanya berisi perubahan terakhir untuk setiap bagian database. Menerapkan ringkasan ini jauh lebih cepat daripada membaca ulang semua log.

| Strategi | Pengaruh ke waktu pemulihan | Biaya |
|---|---|---|
| Image copy lebih sering | Lebih sedikit log yang perlu diterapkan | Lebih banyak waktu dan ruang untuk image copy |
| Change Accumulation terjadwal | Penerapan perubahan jauh lebih cepat | Job tambahan dan ruang untuk hasilnya |
| Image copy online bersamaan dengan transaksi | Tidak perlu menghentikan database | Butuh perencanaan konsistensi |
| HALDB dengan partisi | Pemulihan per partisi, tidak seluruh database | Desain dan pengelolaan partisi |

Contoh perencanaan: database dengan image copy mingguan dan log harian yang besar bisa membutuhkan waktu pemulihan berjam-jam pada

akhir pekan, karena log enam hari harus diterapkan. Change Accumulation harian memangkas waktu penerapan itu secara besar, tanpa menambah frekuensi image copy.

Uji pemulihan database IMS di lingkungan uji secara berkala, dan catat waktunya. Bandingkan dengan RTO yang dijanjikan untuk layanan yang memakai database itu. Jika waktu pemulihan nyata melebihi RTO, frekuensi image copy atau jadwal Change Accumulation perlu disesuaikan sebelum pemulihan itu benar-benar dibutuhkan.

# 🎛️ Bagian XL — WLM Mendalam dan Analisis SMF dengan Python

Bab 40 memperkenalkan WLM dan Bab 41 memperkenalkan SMF. Bagian ini memperdalam keduanya sampai ke rumus yang dipakai WLM dan cara mengolah data SMF sendiri di luar alat komersial.

## Bab 168. Rumus di Balik WLM

WLM mengukur keberhasilan setiap service class dengan Performance Index (PI). Rumusnya berbeda untuk tiap jenis tujuan.

$$PI_{\text{response time}} = \frac{\text{waktu respons aktual}}{\text{target waktu respons}} \qquad PI_{\text{velocity}} = \frac{\text{velocity target}}{\text{velocity aktual}}$$

Velocity mengukur seberapa jarang pekerjaan tertunda karena menunggu sumber daya yang dikelola WLM:

$$\text{Velocity} = \frac{\text{sampel using}}{\text{sampel using} + \text{sampel delay}} \times 100$$

Contoh: service class batch memiliki 300 sampel *using* dan 450 sampel *delay* dalam satu interval. Velocity aktualnya 40. Jika targetnya 50, PI = 50 / 40 = 1,25, artinya tujuan meleset dan WLM akan mencoba memberi sumber daya lebih bila importance-nya cukup tinggi.

Untuk tujuan persentil, misalnya “90% transaksi di bawah 0,3 detik”, PI dihitung dari waktu respons pada persentil tersebut, bukan rata-rata.

### 168.1 Analisis: Memilih Tujuan Velocity

Tujuan velocity dipakai untuk pekerjaan yang tidak punya waktu respons yang jelas, seperti batch, started task, dan server. Memilih angkanya lebih sulit daripada memilih tujuan waktu respons, karena velocity tidak langsung dirasakan pengguna. Pendekatan yang andal adalah menurunkannya dari pengukuran.

$$\text{Velocity} = \frac{\text{sampel using}}{\text{sampel using} + \text{sampel delay}} \times 100$$

**Langkah menentukan tujuan:**

1. Ukur velocity yang dicapai service class itu selama beberapa pekan, terutama pada periode ketika kinerjanya dinilai memuaskan oleh pemiliknya.
2. Ambil nilai yang biasa tercapai pada periode sibuk yang memuaskan, misalnya 55 sampai 65.
3. Tetapkan tujuan sedikit di bawahnya, misalnya 50. Dengan begitu PI berada di bawah 1 saat kondisi baik, dan naik di atas 1 saat pekerjaan benar-benar tertahan.
4. Tinjau setelah perubahan besar.

Tujuan yang terlalu tinggi, misalnya 90 untuk batch, hampir tidak pernah tercapai di sistem yang sibuk. WLM akan terus mencoba membantu service class itu, sehingga pekerjaan yang lebih penting ikut terganggu. Tujuan yang terlalu rendah membuat WLM tidak pernah memprioritaskannya, bahkan saat pekerjaan itu mulai terlambat.

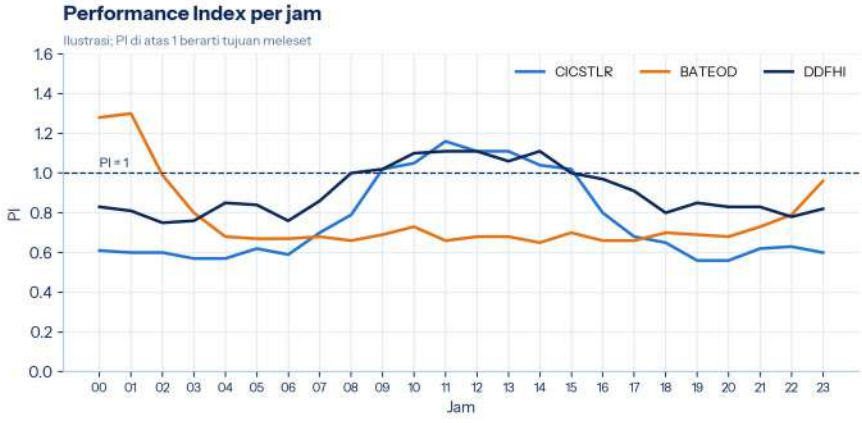
| Perubahan | Pengaruh ke velocity | Tindakan |
|---|---|---|
| Migrasi ke CPC dengan lebih banyak atau lebih cepat prosesor | Delay CPU turun, velocity yang tercapai naik | Ukur ulang dan sesuaikan tujuan setelah stabil |
| Penambahan zIIP | Beban eligible punya jalur tambahan | Tinjau service class yang memakai Java atau DDF |
| Beban baru di sistem | Kontensi naik, velocity turun | Tinjau importance dan tujuan |
| Perubahan pola I/O | Delay device berubah | Periksa apakah velocity masih mencerminkan kinerja |

Velocity adalah alat untuk WLM, bukan metrik untuk pemilik bisnis. Laporkan kepada pemilik bisnis dalam bentuk yang mereka pahami, misalnya “batch EOD selesai pukul 03.40, target 04.00”, dan gunakan velocity sebagai tuas di balik layar.

# Bab 169. Period, Resource Group, dan Pola Harian

| Fitur WLM | Fungsi | Contoh pemakaian |
|---|---|---|
| Multi-period | Tujuan berubah setelah pekerjaan memakai sejumlah service unit | Query DDF pendek dilayani cepat; query panjang turun prioritas |
| Resource group | Batas atas atau bawah kapasitas untuk sekelompok service class | Membatasi beban analitik agar tidak mengganggu OLTP |
| Importance | Urutan pengorbanan saat sumber daya kurang | OLTP importance 1, batch umum 4 |
| Report class | Pelaporan terperinci tanpa memengaruhi prioritas | Satu report class per aplikasi atau kanal |

Pola PI sepanjang hari memperlihatkan kapan sistem kekurangan sumber daya dan siapa yang dikorbankan. Pada ilustrasi berikut, transaksi teller dan API meleset di jam puncak siang, sedangkan batch EOD meleset di tengah malam.




*Ilustrasi · data rekaan*

Jika PI importance 1 meleset di jam puncak, masalahnya kapasitas atau klasifikasi, bukan tuning WLM. Menaikkan importance tidak menciptakan CPU baru.

| Kesalahan umum | Akibat |
|---|---|
| Terlalu banyak service class importance 1 | WLM tidak bisa membedakan yang benar-benar penting |
| Target velocity tidak realistis | PI selalu di atas 1, WLM terus mengejar yang tak tercapai |
| Transaksi baru tidak diklasifikasi | Jatuh ke service class default (lihat Bab 119) |
| Resource group dipasang lalu dilupakan | Beban tercekik bertahun-tahun tanpa ada yang ingat alasannya |

## 169.1 Analisis: Menentukan Durasi Period untuk DDF

Kueri Db2 yang datang lewat DDF sangat beragam. Sebagian besar ringan, misalnya inquiry saldo dari API. Sebagian kecil berat, misalnya kueri laporan dari alat analitik. Service class dengan beberapa period memisahkan keduanya secara otomatis: pekerjaan dimulai di period pertama dengan prioritas tinggi, lalu turun ke period berikutnya bila sudah memakai service unit melebihi durasi yang ditentukan.

Durasi period pertama menentukan pembagian itu. Aturan praktis yang sering dipakai: pilih durasi sehingga sebagian besar pekerjaan, misalnya 80–90%, selesai di period pertama.

$$D_1 \approx P_{85}(\text{service unit per pekerjaan DDF})$$

| Period | Isi | Tujuan contoh | Importance |
|---|---|---|---|
| 1 | Kueri ringan, sebagian besar pekerjaan | 90% selesai < 0,5 detik | 2 |
| 2 | Kueri menengah | Velocity 40 | 3 |
| 3 | Kueri berat yang jarang | Velocity 20 atau discretionary | 5 |

Jika durasi period pertama terlalu panjang, kueri berat ikut menikmati prioritas tinggi dan bisa mengganggu kueri ringan. Jika terlalu pendek, sebagian kueri ringan yang sedikit lebih besar dari biasanya terlempar ke period kedua, dan waktu responsnya memburuk tanpa alasan yang jelas.

Ukur distribusi service unit per pekerjaan DDF dari SMF, lalu tinjau setiap kuartal. Penambahan API baru atau perubahan pola pemakaian analitik dapat menggeser distribusi.

**Resource group** melengkapi period. Jika pengguna analitik ad hoc sesekali menjalankan kueri yang sangat besar, resource group dengan batas kapasitas mencegah mereka menghabiskan CPU di jam sibuk. Kombinasi period dan resource group, ditambah batas ASUTIME di RLF (Bab 155.1), memberi tiga lapis perlindungan bagi beban online.

# Bab 170. Mengambil Data SMF

Data SMF yang disimpan di logstream diambil dengan `IFASMFDL`. Contoh berikut mengambil record tipe 30 dan 70 sampai 79 untuk tanggal 20 September 2026 (hari ke-263).

```
//SMFDUMP  EXEC PGM=IFASMFDL
//SYSPRINT DD SYSOUT=*
//OUTDD1   DD
DSN=CAP.SMF.D2026263,DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(200,100)),RECFM=VBS,LRECL=32760
//SYSIN    DD *
  LSNAME(IFASMF.PLEXP.DATA,OPTIONS(DUMP))
  OUTDD(OUTDD1,TYPE(30,70:79))
  DATE(2026263,2026263)
  START(0000) END(2359)
/*
```

Untuk dianalisis di luar mainframe, unduh dataset dengan mode biner dan tetap menyertakan RDW (misalnya perintah FTP `quote site rdw` sebelum `get` dalam mode binary). Tanpa RDW, batas antar-record hilang.

## 170.1 Analisis: Volume dan Retensi SMF

SMF adalah sumber data terpenting untuk kinerja, kapasitas, keamanan, dan penagihan lisensi. Volumenya juga besar, dan bisa tumbuh tanpa disadari ketika jenis record baru diaktifkan atau aplikasi bertambah. Perencanaan volume dan retensi mencegah dua masalah: data penting hilang karena ruang habis, dan biaya penyimpanan yang membengkak untuk data yang tidak pernah dipakai.

$$V_{\text{total}} = \sum_{t} N_t \times \bar{s}_t \times D_{\text{retensi},\,t}$$

N adalah jumlah record tipe t per hari, s rata-rata ukurannya, dan D retensi untuk tipe itu.

| Kelompok record | Kegunaan | Retensi online yang umum | Arsip |
|---|---|---|---|
| 70–79 (RMF) | Kinerja dan kapasitas | Beberapa bulan | Ringkasan bertahun-tahun untuk tren |
| 30 | Job dan address space | Beberapa bulan | Sesuai kebutuhan audit |
| 80, 81, 83 | Keamanan | Sesuai kebijakan keamanan | Sesuai regulasi |
| 89, 70 untuk SCRT | Penagihan sub-capacity | Sampai laporan bulanan dikirim | Sesuai ketentuan kontrak |
| 110 (CICS), 101 (Db2) | Kinerja transaksi | Beberapa pekan | Ringkasan |
| 119 | Jaringan dan transfer | Beberapa bulan | Sesuai kebijakan keamanan |

Record detail bervolume tinggi seperti SMF 110 dan 101 jarang dibutuhkan dalam bentuk mentah setelah beberapa pekan. Ringkasannya, misalnya per transaksi per jam, jauh lebih kecil dan cukup untuk analisis tren. Pola ini memberi retensi panjang dengan biaya kecil: data mentah untuk investigasi jangka pendek, data ringkas untuk tren jangka panjang.

Pantau volume SMF harian sebagai metrik tersendiri. Lonjakan volume yang tiba-tiba sering menandakan perubahan konfigurasi, misalnya audit yang terlalu luas diaktifkan (Bab 30.1), atau masalah aplikasi yang menghasilkan record berlebihan. Dengan perekaman berbasis logstream, pastikan data dibongkar secara teratur dengan `IFASMFDL` ke dataset harian sebelum logstream mencapai batasnya.

## Bab 171. Parser Header SMF dengan Python

Setiap record SMF diawali header standar: RDW (panjang), flag, tipe record, waktu (seperseratus detik sejak tengah malam), tanggal packed

`0cyydddF`, dan ID sistem. Parser berikut membaca header itu dan menghitung jumlah record per sistem, tanggal, jam, dan tipe.

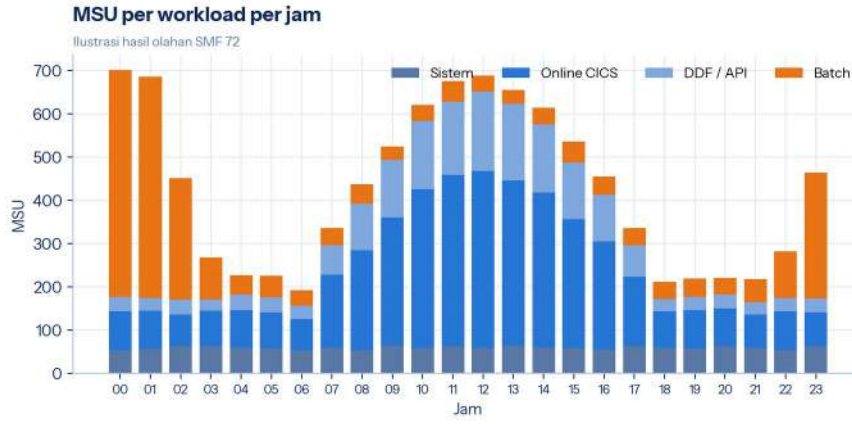
```
import struct, collections, datetime

def tanggal_smf(b):
    h = b.hex()                       # contoh '0126263f': c=1, yy=26, ddd=263
    c, yy, ddd = int(h[1]), int(h[2:4]), int(h[4:7])
    return datetime.date(1900 + 100 * c + yy, 1, 1) + datetime.timedelta(days=ddd - 1)

def baca_smf(path):
    with open(path, 'rb') as f:
        while True:
            rdw = f.read(4)
            if len(rdw) < 4:
                break
            panjang, _ = struct.unpack('>HH', rdw)
            isi = f.read(panjang - 4)
            tipe = isi[1]
            waktu = struct.unpack('>I', isi[2:6])[0]   # seperseratus detik
            tgl = tanggal_smf(isi[6:10])
            sid = isi[10:14].decode('cp037')           # EBCDIC
            yield sid, tgl, waktu // 360000, tipe

hitung = collections.Counter(baca_smf('CAP.SMF.D2026263.bin'))
for (sid, tgl, jam, tipe), n in sorted(hitung.items()):
    print(sid, tgl, f'{jam:02d}', tipe, n)
```

Kode ini sudah diuji dengan data sintetis. Beberapa hal yang perlu diperhatikan saat memakai data nyata:

- Record yang disimpan sebagai *spanned* (VBS) dapat terpecah dalam beberapa segmen; kolom kedua RDW menandai segmen dan harus digabung sebelum diproses.
- Isi setelah header tiap tipe record memakai struktur *triplet* (offset, panjang, jumlah) yang berbeda per tipe dan versi; rujuk *z/OS MVS System Management Facilities* untuk tata letaknya.

- Untuk analisis produksi berkelanjutan, alat seperti IBM Z Common Data Provider atau produk pihak ketiga lebih tepat daripada parser buatan sendiri.

Setelah field tipe 72 diurai, pemakaian CPU dapat dikelompokkan per workload dan per jam. Hasilnya menjelaskan siapa yang menyusun puncak harian:




*Ilustrasi · data rekaan*

## 171.1 Analisis: Memvalidasi Hasil Parser SMF

Parser SMF buatan sendiri sangat berguna, tetapi juga berbahaya bila hasilnya dipercaya tanpa diuji. Kesalahan kecil dalam membaca offset, tipe data, atau record bersegmen dapat menghasilkan angka yang terlihat wajar tetapi salah. Keputusan kapasitas yang diambil dari angka seperti itu ikut salah.

| Lapisan validasi | Cara | Yang ditangkap |
|---|---|---|
| Jumlah record per tipe | Bandingkan dengan ringkasan dari utilitas dump SMF | Record terlewat atau terbaca ganda |
| Rentang waktu | Record pertama dan terakhir sesuai periode yang diminta | Masalah konversi tanggal dan zona waktu |
| Nilai yang dikenal | Bandingkan beberapa angka dengan laporan RMF resmi | Offset atau tipe data yang salah |

| Lapisan validasi | Cara | Yang ditangkap |
|---|---|---|
| | untuk interval yang sama | |
| Data sintetis | Record buatan dengan nilai yang sudah diketahui | Kesalahan logika parser |
| Record bersegmen | Uji record yang melebihi satu blok | Record panjang yang terpotong |

Record SMF bisa lebih panjang dari satu blok fisik dan disimpan dalam beberapa segmen. Parser yang tidak menangani segmen akan membaca potongan pertama sebagai record utuh dan menghasilkan data acak dari sisa blok. Ini adalah kesalahan paling umum pada parser buatan sendiri.

```
ALGORITMA ValidasiParser(file_smf, hasil_parser,
ringkasan_utilitas)
  untuk setiap tipe t: bandingkan jumlah record
hasil_parser[t] dengan ringkasan_utilitas[t]
  periksa bahwa semua stempel waktu berada dalam periode
file
  ambil tiga interval acak; bandingkan CPU per service
class dengan laporan RMF resmi
  toleransi: selisih kecil karena pembulatan; selisih
besar = CACAT
  jalankan ulang uji dengan data sintetis setiap kali
parser diubah
```

Simpan uji-uji ini bersama kode parser di Git, dan jalankan setiap kali parser diubah. Parser yang diuji seperti ini bisa diandalkan untuk laporan kapasitas yang dibawa ke manajemen (Bab 172).

## Bab 172. Dari Data ke Keputusan Kapasitas

Data SMF baru bernilai jika menghasilkan keputusan. Proyeksi sederhana dengan pertumbuhan majemuk bulanan sudah cukup untuk rapat anggaran pertama:

$$MSU_t = MSU_0 \times (1 + g)^t$$

MSU nol adalah puncak 4HRA bulan ini, g adalah pertumbuhan bulanan rata-rata dari 12 bulan terakhir, dan t jumlah bulan ke depan.

```
ALGORITMA ReviewKapasitasBulanan()
  ambil puncak 4HRA bulanan 12 bulan terakhir dari SMF 70
(per LPAR)
  g <- rata-rata pertumbuhan bulanan
  proyeksikan 12 bulan ke depan dengan skenario g dan 1,5
x g
  bulan_kritis <- bulan pertama proyeksi melewati 85%
kapasitas terpasang
  jika bulan_kritis kurang dari lead time pengadaan + 3
bulan maka
      ajukan penambahan kapasitas atau optimasi (zIIP,
rekompilasi, jadwal batch)
  catat asumsi dan bandingkan dengan realisasi bulan
berikutnya
```

## 172.1 Analisis: Menyajikan Proyeksi Kapasitas dengan Skenario

Proyeksi kapasitas dengan satu angka tunggal hampir pasti meleset, dan kesan yang tertinggal adalah tim kapasitas “salah hitung”. Proyeksi dengan beberapa skenario jauh lebih jujur dan lebih berguna bagi pengambil keputusan, karena memperlihatkan rentang kemungkinan dan kapan keputusan harus diambil.

| Skenario | Asumsi | Kapan kapasitas habis (contoh) | Keputusan yang dipicu |
|---|---|---|---|
| Rendah | Pertumbuhan transaksi 10% per tahun; tidak ada produk baru | 34 bulan lagi | Tidak ada tindakan tahun ini |
| Dasar | Pertumbuhan 20% per tahun; satu produk digital baru | 18 bulan lagi | Masukkan penambahan kapasitas ke anggaran tahun depan |
| Tinggi | Pertumbuhan 30% per tahun; akuisisi portofolio bank lain | 10 bulan lagi | Siapkan opsi kapasitas sementara (misalnya kapasitas on demand) |

$$T_{\text{habis}} = \frac{\ln\left(K_{\text{maks}}/K_{\text{sekarang}}\right)}{\ln(1+g)}$$

Rumus ini memberi waktu, dalam tahun, sampai kapasitas habis bila pemakaian tumbuh secara majemuk dengan laju g per tahun. Contoh: pemakaian puncak sekarang 65% dari kapasitas yang tersedia, dan batas aman 85%. Dengan g = 20%, waktu tersisa sekitar ln(85/65) / ln(1,2) ≈ 1,5 tahun, atau sekitar 18 bulan.

Sajikan juga pemicu keputusan yang terukur, bukan hanya tanggal. Contohnya: “jika puncak 4HRA bulanan melewati 75% selama dua bulan berturut-turut, pengadaan dimulai.” Pemicu seperti ini bisa dipantau bulanan dan menghindari perdebatan panjang tentang skenario mana yang benar.

Perbarui proyeksi setiap kuartal dengan data terbaru (Bab 170.1 dan 171.1). Bandingkan proyeksi sebelumnya dengan kenyataan. Riwayat akurasi ini membangun kepercayaan manajemen terhadap proyeksi berikutnya.

# 🤝 Bagian XLI — Operasi Tim, Vendor, dan Lisensi

Keandalan mainframe bergantung sama besarnya pada cara tim bekerja seperti pada teknologinya. Bagian ini membahas jaga dan eskalasi, komunikasi saat insiden, postmortem, hubungan dengan IBM dan vendor, serta sisi biaya yang ikut menentukan keputusan teknis.

## Bab 173. Jaga dan Eskalasi

| Tingkat | Definisi | Contoh | Respons awal | Eskalasi |
|---|---|---|---|---|
| S1 | Layanan utama berhenti atau data terancam | Core banking tidak bisa diakses, EOD gagal mendekati batas | 15 menit | Kepala TI dan bisnis segera; buka kasus IBM Sev 1 bila relevan |
| S2 | Layanan terganggu berat, ada solusi sementara | Satu kanal lambat, satu sistem sysplex turun | 30 menit | Manajer operasi dalam 1 jam |
| S3 | Dampak terbatas | Job non-kritikal gagal, peringatan kapasitas | Hari kerja yang sama | Pemimpin tim |
| S4 | Tanpa dampak layanan | Pertanyaan, perbaikan kecil | Terjadwal | Tidak perlu |

Prinsip jaga yang sehat:

- Rotasi terjadwal dengan pengganti yang jelas; tidak ada orang yang jaga terus-menerus.
- Serah terima tertulis di awal dan akhir giliran: insiden terbuka, perubahan hari ini, risiko malam ini.
- Petugas jaga berwenang menjalankan runbook tanpa menunggu persetujuan, dan wajib eskalasi bila di luar runbook.

- Setelah malam dengan insiden besar, petugas jaga berhak beristirahat; kelelahan adalah sumber insiden berikutnya.

## 173.1 Analisis: Menghitung Beban Jaga

Jadwal jaga yang tidak sehat adalah risiko operasional yang nyata. Petugas yang terlalu sering dibangunkan tengah malam membuat keputusan yang lebih buruk, dan tim yang kelelahan cenderung kehilangan anggota berpengalaman. Beban jaga bisa dihitung sederhana dari ukuran rotasi dan jumlah panggilan.

$$\text{Pekan jaga per orang per tahun} = \frac{52}{N_{\text{rotasi}}} \qquad \text{Panggilan per orang per tahun} = \frac{52}{N_{\text{rotasi}}} \times P_{\text{per pekan}}$$


*Gambar 173.1 — Beban jaga per orang menurut ukuran rotasi*

Dengan rotasi tiga orang, setiap orang berjaga lebih dari 17 pekan setahun, atau hampir satu dari setiap tiga pekan. Dengan rata-rata enam panggilan per pekan, itu lebih dari seratus panggilan per orang per tahun, sebagian di tengah malam. Rotasi lima atau enam orang menurunkan beban ke tingkat yang lebih berkelanjutan.

Ada dua tuas untuk memperbaiki beban jaga:

| Tuas | Cara | Catatan |
|---|---|---|
| Memperbesar rotasi | Melatih lebih banyak orang | Butuh investasi pelatihan dan |

| Tuas | Cara | Catatan |
|---|---|---|
| | untuk jaga, termasuk dari tim aplikasi dengan runbook yang jelas | dokumentasi |
| Mengurangi panggilan | Menghapus alert yang tidak butuh tindakan (Bab 47.4), memperbaiki akar masalah yang berulang (Bab 175.1), otomasi tindakan rutin (Bab 26.4) | Hasilnya dirasakan semua anggota rotasi |

Tuas kedua sering lebih efektif dan lebih murah. Menurunkan panggilan dari enam menjadi tiga per pekan setara dengan menggandakan ukuran rotasi, tanpa menambah satu orang pun.

Ukur beban jaga setiap bulan: jumlah panggilan, jumlah panggilan di luar jam kerja, dan waktu yang dihabiskan per panggilan. Laporkan angka ini kepada manajemen bersama rencana perbaikannya. Beban jaga yang terukur jauh lebih mudah diperjuangkan daripada keluhan umum bahwa “jaga terlalu berat”.

## Bab 174. Komunikasi Saat Insiden

Saat insiden S1, satu orang ditunjuk sebagai koordinator yang tidak ikut memperbaiki, agar tim teknis bisa fokus. Pembaruan status dikirim berkala dengan format tetap.

```
[S1] Core banking tidak dapat diakses — Pembaruan #3, 02.45
Dampak     : semua kanal tidak dapat bertransaksi sejak 02.10
Penyebab   : dugaan kegagalan structure CF, sedang diverifikasi
Tindakan   : rebuild structure ke CF02 sedang berjalan (RB-030)
Perkiraan  : pembaruan berikut 03.15 atau lebih cepat bila ada kemajuan
Koordinator: <nama> · Telepon jembatan: <nomor>
```

Irama pembaruan untuk S1 setiap 30 menit, untuk S2 setiap jam. Tanpa kabar lebih buruk daripada kabar “belum ada kemajuan”.

## 174.1 Analisis: Irama Komunikasi Saat Insiden

Selama insiden besar, keheningan dari tim teknis diisi dengan tebakan. Manajemen bertanya berulang-ulang, cabang menyebarkan kabar yang belum pasti, dan tim teknis terganggu oleh pertanyaan yang sebenarnya bisa dijawab dengan satu pembaruan terjadwal. Irama komunikasi yang disepakati sebelumnya mencegah semua itu.

| Tingkat insiden | Interval pembaruan | Penerima |
|---|---|---|
| Kritikal: layanan nasabah berhenti | Setiap 15–30 menit | Manajemen puncak, unit bisnis, layanan nasabah |
| Tinggi: layanan terganggu berat | Setiap 30–60 menit | Manajemen TI, unit bisnis terdampak |
| Sedang: dampak sebagian | Setiap beberapa jam atau saat ada kemajuan | Pemilik layanan |
| Rendah | Saat selesai | Tiket |

**Isi setiap pembaruan, dengan urutan tetap:**

```
[Waktu] Pembaruan ke-N — Insiden <nomor> — <status: investigasi / pemulihan / terpantau / selesai>
Dampak saat ini: <layanan apa, nasabah siapa, sejak kapan>
Yang sudah diketahui: <fakta, bukan dugaan>
Yang sedang dikerjakan: <langkah saat ini, siapa>
Perkiraan berikutnya: <kapan pembaruan berikutnya, bukan janji waktu pulih>
```

Baris terakhir penting. Menjanjikan waktu pulih yang belum pasti adalah kesalahan komunikasi yang paling sering, dan setiap janji yang meleset menurunkan kepercayaan. Yang dijanjikan adalah waktu pembaruan berikutnya, dan janji itu harus ditepati meskipun isinya “belum ada kemajuan”.

Pisahkan peran komunikator dari peran teknis. Satu orang ditugasi khusus menulis pembaruan dan menjawab pertanyaan, sehingga tim teknis dapat

fokus pada pemulihan (Bab 173). Setelah insiden selesai, kumpulan pembaruan ini menjadi linimasa awal untuk postmortem (Bab 175).

## Bab 175. Postmortem Tanpa Menyalahkan

Tujuan postmortem adalah memperbaiki sistem, bukan mencari orang yang salah. Orang yang takut disalahkan akan menyembunyikan informasi, dan informasi yang tersembunyi adalah bahan insiden berikutnya.

```
# Postmortem: <judul insiden>
- Tanggal dan durasi, tingkat, dampak ke nasabah dan
bisnis
## Kronologi
- Waktu, kejadian, keputusan, siapa (peran, bukan untuk
menyalahkan)
## Akar masalah
- Dianalisis dengan "lima kali mengapa" sampai ke kontrol
yang gagal
## Apa yang berjalan baik
## Apa yang harus diperbaiki
## Tindakan
- [ ] Tindakan, pemilik, tenggat, runbook yang dibuat atau
direvisi
```

Setiap postmortem S1 dan S2 harus menghasilkan minimal satu perubahan kontrol dan satu runbook baru atau revisi. Tinjau status tindakan postmortem dalam rapat bulanan sampai semuanya selesai.

### 175.1 Analisis: Mengukur Apakah Postmortem Benar-Benar Bekerja

Postmortem yang baik menghasilkan daftar tindakan perbaikan. Postmortem hanya bernilai jika tindakan itu benar-benar dikerjakan, dan jika insiden sejenis tidak terulang. Dua metrik sederhana mengukur keduanya.

$$\text{Tingkat penyelesaian} = \frac{\text{tindakan selesai tepat waktu}}{\text{total tindakan jatuh tempo}} \qquad \text{Tingkat pengulangan} = \frac{\text{insiden dengan akar masalah yang sudah pernah dibahas}}{\text{total insiden}}$$

| Kondisi | Tafsiran | Tindakan |
|---|---|---|
| Penyelesaian tinggi, | Proses bekerja | Pertahankan |

| Kondisi | Tafsiran | Tindakan |
|---|---|---|
| pengulangan rendah | | |
| Penyelesaian rendah | Tindakan terlalu banyak, terlalu besar, atau tanpa pemilik | Batasi tindakan per postmortem; setiap tindakan punya pemilik dan tenggat |
| Penyelesaian tinggi, pengulangan tetap tinggi | Tindakan tidak menyentuh akar masalah | Tinjau kualitas analisis akar masalah |
| Banyak tindakan ditunda berulang kali | Prioritas bersaing dengan pekerjaan proyek | Alokasikan kapasitas tim khusus untuk tindak lanjut insiden |

Pola yang sering terlihat adalah postmortem yang menghasilkan belasan tindakan, sebagian besar berupa “tingkatkan pemantauan” atau “perbarui dokumentasi”, lalu semuanya tertunda karena kalah prioritas dengan proyek. Lebih baik tiga tindakan yang jelas, berdampak, dan selesai, daripada lima belas yang tidak pernah disentuh.

Tinjau kedua metrik setiap bulan bersama daftar tindakan yang terbuka. Tindakan yang tertunda lebih dari dua kali harus diputuskan secara eksplisit: dikerjakan dengan prioritas baru, atau ditutup dengan penerimaan risiko yang tercatat. Menunda tanpa keputusan adalah cara paling pasti untuk bertemu insiden yang sama lagi.

## Bab 176. IBM Support dan Vendor

Kasus ke IBM dibuka lewat portal IBM Support dengan tingkat keparahan yang jujur. Sev 1 berarti dampak bisnis kritis di produksi; menyalahgunakan Sev 1 untuk pertanyaan biasa merusak kepercayaan saat benar-benar dibutuhkan.

| Data yang biasanya diminta IBM | Cara mendapatkannya |
|---|---|
| SVC dump atau stand-alone dump | Dataset dump sistem; kompres dengan `AMATERSE` sebelum dikirim |
| SYSLOG / OPERLOG di sekitar waktu kejadian | SDSF atau arsip log |
| LOGREC | Laporan EREP atau logstream LOGREC |
| Level software | Laporan SMP/E dan `D PROD,STATE` |

| Data yang biasanya diminta IBM | Cara mendapatkannya |
|---|---|
| Trace komponen | Diaktifkan sesuai instruksi IBM |

Data dikirim lewat layanan unggah resmi IBM (ECuRep) dengan nomor kasus. Pastikan dump tidak berisi data nasabah yang melanggar kebijakan, atau gunakan mekanisme yang disetujui unit keamanan.

Untuk vendor aplikasi core banking dan produk pihak ketiga, tuliskan di kontrak: waktu respons per tingkat keparahan, jalur eskalasi, akses jarak jauh yang diizinkan, dan kewajiban menjaga kerahasiaan data.

## 176.1 Analisis: Membuka Case yang Cepat Selesai

Waktu penyelesaian case ke IBM atau vendor sangat dipengaruhi kualitas informasi pertama yang dikirim. Case yang dibuka dengan “sistem lambat, tolong bantu” akan menghabiskan satu atau dua hari hanya untuk tanya jawab. Case dengan paket informasi lengkap bisa langsung dianalisis.

| Tingkat keparahan | Arti umum | Contoh |
|---|---|---|
| 1 | Dampak bisnis kritikal, layanan produksi berhenti | Sistem produksi tidak bisa di-IPL |
| 2 | Dampak signifikan, layanan terganggu berat | Abend berulang di subsistem produksi dengan jalan pintas yang mahal |
| 3 | Dampak sebagian, ada jalan pintas | Fungsi non-kritikal gagal |
| 4 | Dampak minimal atau pertanyaan | Pertanyaan konfigurasi atau dokumentasi |

Pilih tingkat keparahan secara jujur. Tingkat 1 untuk masalah non-kritikal menguras kepercayaan, dan membuat eskalasi yang sungguh-sungguh kurang didengar di kemudian hari.

**Paket informasi awal:**

- Deskripsi singkat: apa yang terjadi, kapan mulai, apa yang berubah sebelumnya.
- Pesan dan kode persis, disalin, bukan diketik ulang.

- Level software: rilis z/OS dan produk terkait, dan RSU atau PTF terakhir yang terpasang.
- Dump yang utuh (Bab 51.1), SYSLOG atau OPERLOG di sekitar waktu kejadian, dan LOGREC.
- Apa yang sudah dicoba dan hasilnya.
- Dampak bisnis dalam satu kalimat.

IBM menerbitkan panduan *MustGather* untuk banyak komponen, berisi daftar data yang dibutuhkan untuk jenis masalah tertentu. Ikuti panduan itu sebelum membuka case, bukan setelah diminta. Unggah data besar seperti dump lewat saluran unggah resmi IBM, dan catat nomor case di log insiden internal.

Setelah case selesai, simpan ringkasan masalah, APAR atau PTF solusinya, dan pelajaran yang didapat di basis pengetahuan internal. Masalah yang sama sering muncul lagi di sistem lain atau setelah RSU berikutnya. Catatan satu paragraf bisa menghemat satu hari kerja di kemudian hari.

## Bab 177. Lisensi dan Total Biaya Kepemilikan

| Model lisensi | Ciri | Contoh |
|---|---|---|
| MLC (Monthly License Charge) | Biaya bulanan berbasis MSU; sub-capacity dilaporkan lewat SCRT | z/OS, CICS, Db2, IMS, MQ pada model tradisional |
| IPLA (One-Time Charge + S&S) | Beli sekali, lalu biaya dukungan tahunan | Banyak alat dan produk tambahan |
| Tailored Fit Pricing | Berbasis konsumsi total atau kapasitas terpasang | Negosiasi dengan IBM |
| Lisensi vendor pihak ketiga | Berbasis MSU, kapasitas, atau jumlah pengguna | Scheduler, alat keamanan, monitoring |

TCO mainframe perlu dibandingkan secara utuh, bukan hanya harga hardware:

$$TCO_N = \sum_{t=1}^{N} (H_t + S_t + L_t + P_t + F_t)$$

H adalah hardware dan pemeliharaannya, S software IBM, L lisensi pihak ketiga, P personel, dan F fasilitas (ruang, listrik, pendinginan, jaringan) pada tahun ke-t. Perbandingan dengan platform lain harus memakai beban yang setara, termasuk ketersediaan, keamanan, dan biaya migrasi.

- Tinjau laporan SCRT setiap bulan sebelum dikirim; puncak 4HRA yang tidak wajar harus dijelaskan.
- Masukkan dampak MSU ke setiap keputusan teknis besar: rilis aplikasi, jadwal batch, dan pemindahan beban.
- Libatkan tim pengadaan sejak awal saat kontrak IBM dan vendor mendekati perpanjangan.

## 177.1 Analisis: Komponen TCO yang Bisa Dipengaruhi Tim Teknis

Total biaya kepemilikan mainframe sering dibahas sebagai angka tunggal yang besar. Memecahnya menjadi komponen memperlihatkan bagian mana yang bisa dipengaruhi keputusan teknis sehari-hari, dan bagian mana yang ditentukan kontrak.


*Gambar 177.1 — Komposisi TCO mainframe lima tahun*

Dalam ilustrasi, software dengan biaya bulanan (MLC) adalah komponen terbesar. Komponen ini juga yang paling bisa dipengaruhi tim teknis,

karena tagihannya mengikuti puncak 4HRA untuk produk yang memakai model sub-capacity. Setiap penurunan puncak yang berkelanjutan, misalnya dari offload ke zIIP, penggeseran batch, atau tuning SQL, dapat menurunkan tagihan bulan-bulan berikutnya (Bab 42.2).

| Komponen | Siapa yang memengaruhi | Tuas utama |
|---|---|---|
| Hardware dan pemeliharaan | Pengadaan, arsitektur | Ukuran konfigurasi, jumlah CPC, cadangan kapasitas |
| Software MLC | Tim teknis dan pengadaan | Puncak 4HRA, defined capacity, penempatan beban |
| Software IPLA dan S&S | Pengadaan | Jumlah produk dan kapasitas yang dilisensikan |
| Fasilitas dan energi | Pusat data | Konsolidasi, efisiensi energi generasi baru |
| Tim dan pelatihan | Manajemen TI | Otomasi, dokumentasi, pengembangan tim |

Hubungan antara penurunan MSU dan penurunan tagihan tidak selalu proporsional, karena tarif MLC umumnya bertingkat: harga per MSU berbeda di setiap rentang kapasitas. Karena itu perkiraan penghematan harus dihitung dengan struktur tarif kontrak yang berlaku, bersama tim pengadaan dan laporan SCRT.

Komponen tim sering dianggap tetap, padahal investasi pada otomasi dan dokumentasi menurunkan beban operasional dan risiko kehilangan pengetahuan saat anggota tim senior pensiun. Menghitung komponen ini dalam TCO membantu menjelaskan mengapa pengembangan tim adalah investasi, bukan hanya biaya.

# 🔗 Bagian XLII — Parallel Sysplex Mendalam

Bab 43 memperkenalkan sysplex. Bagian ini membahas mekanisme yang menentukan apakah sysplex benar-benar meningkatkan ketersediaan, atau justru menambah titik kegagalan.

## Bab 178. Signaling XCF

Sistem dalam sysplex saling mengirim sinyal lewat XCF untuk berkoordinasi dan saling memantau. Jalur sinyal bisa lewat list structure di Coupling Facility atau lewat koneksi CTC langsung.

| Konsep | Arti |
|---|---|
| PATHIN / PATHOUT | Jalur sinyal masuk dan keluar setiap sistem |
| Transport class | Pengelompokan sinyal berdasarkan ukuran pesan dan buffer |
| Failure detection interval | Waktu tanpa sinyal sebelum sistem dianggap bermasalah |
| Status update | Sinyal detak yang ditulis setiap sistem ke couple dataset |

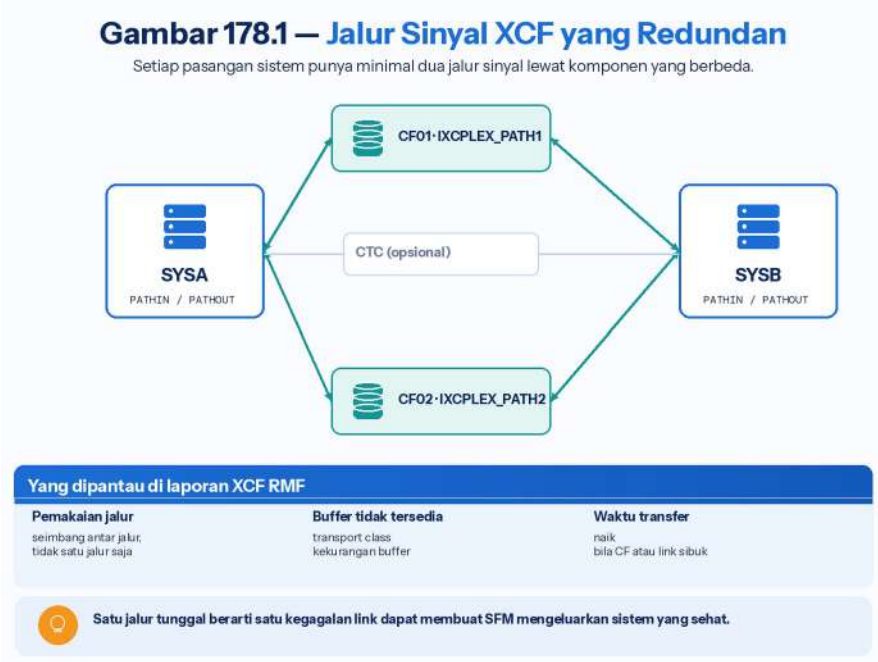
```
D XCF,PATHIN                Jalur sinyal masuk dan
statusnya
D XCF,PATHOUT               Jalur sinyal keluar
D XCF,CLASSDEF,CLASS=ALL    Definisi transport class
D XCF,COUPLE                Interval deteksi kegagalan
dan CDS
```

Setiap sistem sebaiknya punya minimal dua jalur sinyal ke setiap sistem lain, lewat CF yang berbeda. Satu jalur tunggal berarti satu kegagalan link bisa memicu SFM mengeluarkan sistem yang sebenarnya sehat.

### 178.1 Analisis: Kapasitas dan Redundansi Signaling XCF

Signaling XCF membawa pesan antar-sistem sysplex: koordinasi GRS, lock Db2 dan IRLM yang tidak lewat CF, komunikasi JES2, dan detak antar-

sistem. Jika jalur sinyal lambat atau hilang, dampaknya menyebar ke seluruh sysplex.




*Gambar 178.1 — Jalur sinyal XCF yang redundan*

**Redundansi.** Setiap pasangan sistem harus terhubung lewat minimal dua jalur yang tidak berbagi titik kegagalan, misalnya dua list structure di dua CF berbeda. Jika kedua jalur berada di CF yang sama, kegagalan satu CF memutus sinyal sepenuhnya. SFM lalu dapat mengeluarkan sistem yang sebenarnya sehat dari sysplex (Bab 180).

**Kapasitas.** Pesan XCF dikelompokkan ke transport class berdasarkan ukuran. Setiap class punya buffer. Bila buffer habis, pengirim harus menunggu, dan laporan XCF di RMF mencatat kejadian ini.

| Indikator di laporan XCF | Arti | Tindakan |
|---|---|---|
| Buffer tidak tersedia untuk sebuah class | Buffer class terlalu kecil untuk lalu lintas puncak | Perbesar batas buffer class tersebut |
| Pesan besar dikirim di class kecil | Pemborosan dan penundaan | Sesuaikan definisi class dengan ukuran pesan nyata |
| Pemakaian jalur tidak | Satu jalur memikul hampir | Periksa definisi jalur dan |

| Indikator di laporan XCF | Arti | Tindakan |
|---|---|---|
| seimbang | semua lalu lintas | kesehatan jalur lain |
| Waktu transfer naik | CF sibuk atau link bermasalah | Periksa utilisasi CF dan status link |

$$\text{Kebutuhan buffer class} \approx \lambda_{\text{pesan}} \times t_{\text{transfer}} \times \bar{s}_{\text{pesan}} \times (1 + h)$$

Rumus ini, versi lain dari hukum Little, menunjukkan bahwa kebutuhan buffer naik bila laju pesan naik atau waktu transfer memburuk. Karena itu buffer yang cukup di hari biasa bisa kurang saat CF sedang sibuk.

Tinjau laporan XCF setiap bulan dan setelah setiap perubahan besar, misalnya penambahan anggota data sharing atau migrasi CF. Kekurangan buffer jarang menyebabkan gangguan besar sekaligus. Yang terjadi adalah perlambatan halus yang menyebar ke banyak subsistem dan sulit ditelusuri bila laporan XCF tidak pernah dilihat.

## Bab 179. Structure dan Duplexing

| Tipe structure | Contoh pemakai | Perilaku saat CF gagal |
|---|---|---|
| Lock | Db2 IRLM, VSAM RLS, GRS star | Harus dibangun ulang atau di-duplex; kritikal |
| Cache | Group buffer pool Db2, RLS cache | Bisa dibangun ulang dari data di disk |
| List | Signaling XCF, shared queue MQ, logger | Tergantung pemakai; sebagian butuh duplexing |

Duplexing menyimpan dua salinan structure di dua CF. System-managed duplexing melindungi hampir semua jenis structure, dengan biaya latensi karena setiap pembaruan ditulis ke dua CF.

Contoh definisi structure dalam CFRM policy dengan utilitas `IXCMIAPU`:

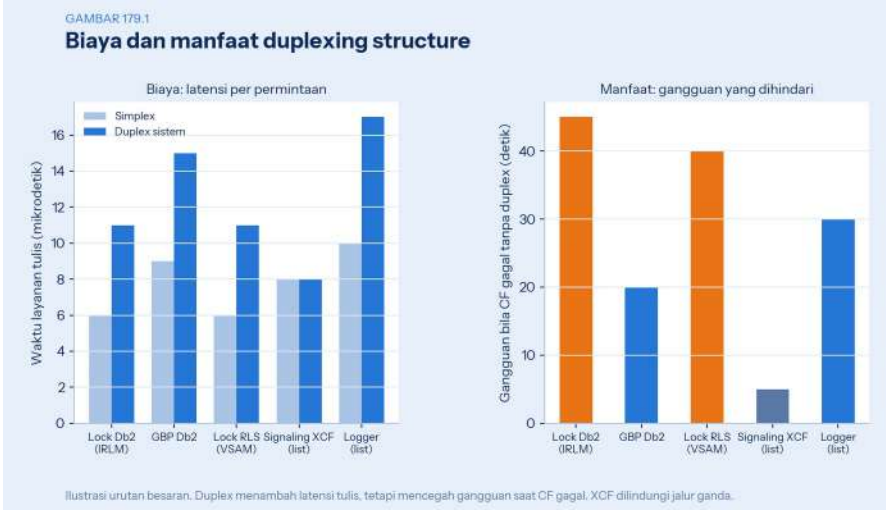
```
//DEFCFRM  EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM02) REPLACE(YES)
```

```
    /* Pernyataan CF NAME(CF01) dan CF NAME(CF02) berisi
identitas   */
    /* hardware setiap CF sesuai hasil D CF; dihilangkan
di contoh ini */
    STRUCTURE NAME(DSNDB0G_LOCK1)
              SIZE(512M) INITSIZE(256M)
              PREFLIST(CF01,CF02)
              DUPLEX(ENABLED)
    STRUCTURE NAME(DSNDB0G_GBP1)
              SIZE(2G) INITSIZE(1G)
              PREFLIST(CF02,CF01)
              DUPLEX(ENABLED)
/*

SETXCF START,POLICY,TYPE=CFRM,POLNAME=CFRM02   Aktifkan
policy baru
SETXCF START,REALLOCATE                        Tempatkan
ulang structure sesuai policy
D XCF,STR,STRNAME=DSNDB0G_LOCK1                Status dan
lokasi structure
```

Setelah mengaktifkan policy baru, jalankan REALLOCATE agar structure yang sudah ada benar-benar mengikuti preferensi baru. Tanpa itu, policy baru baru terasa saat terjadi rebuild berikutnya.

## 179.1 Analisis: Structure Mana yang Perlu Di-duplex?

Duplexing tidak gratis. Setiap permintaan tulis ke structure yang di-duplex harus diselesaikan di dua CF, sehingga waktu layanannya naik dan CPU CF yang terpakai bertambah. Karena itu keputusan duplexing dibuat per structure, dengan menimbang biaya itu terhadap gangguan yang dihindari.




*Gambar 179.1 — Biaya dan manfaat duplexing structure*

| Structure | Tanpa duplex, bila CF gagal | Rekomendasi umum |
|---|---|---|
| Lock Db2 (IRLM) | Rebuild dari informasi di anggota yang masih hidup; transaksi tertahan selama rebuild | Duplex bila tidak ada failure isolation, yaitu CF di mesin terpisah dari anggota data sharing |
| Group buffer pool Db2 | Halaman yang belum di-castout harus dipulihkan; bisa muncul objek GRECP | Duplex untuk GBP objek kritikal |
| Lock RLS (VSAM) | Mirip lock Db2 | Duplex bila RLS dipakai untuk data online kritikal |
| Signaling XCF | Pindah ke jalur lain bila tersedia | Tidak perlu duplex; cukup jalur ganda di CF berbeda (Bab 178.1) |
| Logger | Bergantung pada staging dataset | Duplex atau staging dataset sesuai kebutuhan pemulihan |

**Failure isolation** adalah konsep kunci. Jika CF berada di mesin yang sama dengan anggota data sharing, satu kegagalan mesin menghapus CF dan anggota sekaligus. Informasi untuk membangun ulang structure pun ikut hilang. Dalam kondisi ini duplexing hampir wajib untuk structure lock. Jika CF berada di mesin terpisah, rebuild dari anggota yang masih hidup menjadi pilihan yang lebih realistis.

$$\text{Overhead duplex} \approx N_{\text{tulis per detik}} \times (t_{\text{duplex}} - t_{\text{simplex}})$$

Ukur waktu layanan CF sebelum dan sesudah duplexing diaktifkan untuk setiap structure. Bila overhead-nya terlalu besar untuk sebuah structure, pertimbangkan alternatif: failure isolation yang lebih baik, pembagian beban ke structure lain, atau menerima rebuild dengan prosedur yang sudah diuji.

# Bab 180. SFM, ARM, dan GRS Star

| Kebijakan | Fungsi | Pengaturan penting |
|---|---|---|
| SFM (Sysplex Failure Management) | Mengisolasi sistem yang tidak merespons tanpa campur tangan operator | Isolasi otomatis, penanganan kegagalan koneksi, deteksi sistem yang tersendat |
| ARM (Automatic Restart Manager) | Menyalakan ulang subsistem yang gagal, di sistem yang sama atau lain | Elemen restart untuk Db2, CICS, MQ, IMS |
| GRS star | Serialisasi global lewat lock structure `ISGLOCK` | Jauh lebih efisien daripada mode ring |

```
D XCF,POLICY,TYPE=SFM          Kebijakan SFM aktif
D XCF,ARMSTATUS,DETAIL          Elemen ARM dan statusnya
D GRS                           Mode GRS (STAR) dan anggota
D GRS,C                         Contention ENQ saat ini
```

SFM yang dikonfigurasi baik adalah alasan sysplex bisa pulih dalam hitungan detik saat satu sistem hang. Tanpa SFM, sysplex menunggu operator menjawab WTOR, dan seluruh sistem lain ikut tertahan selama menunggu.

## 180.1 Analisis: Menyetel SFM antara Cepat dan Aman

SFM memutuskan kapan sebuah sistem yang tidak merespons dikeluarkan dari sysplex. Keputusan ini selalu merupakan pertukaran. Terlalu cepat, sistem yang hanya tersendat sesaat ikut dikeluarkan, dan layanan di sistem itu terhenti tanpa perlu. Terlalu lambat, seluruh sysplex menunggu sistem yang sebenarnya sudah mati, dan semua sistem lain ikut tertahan.

$$T_{\text{isolasi}} \approx T_{\text{deteksi kegagalan}} + T_{\text{ISOLATETIME}} + T_{\text{fencing}}$$

| Parameter | Fungsi | Pertimbangan |
|---|---|---|
| Interval deteksi kegagalan di `COUPLExx` | Berapa lama tanpa pembaruan status sebelum sistem dianggap bermasalah | Terlalu pendek memicu isolasi palsu saat sistem tersendat |
| `ISOLATETIME` | Jeda tambahan sebelum isolasi otomatis | Nol berarti segera; umumnya dipakai untuk isolasi otomatis cepat |
| `SSUMLIMIT` | Batas untuk sistem yang tetap hidup tetapi berhenti memperbarui status | Menangani sistem yang tidak mati tetapi juga tidak sehat |
| `CONNFAIL(YES)` | Menangani kegagalan konektivitas antar-sistem | Memilih sistem mana yang dipertahankan saat signaling terputus |
| `MEMSTALLTIME` | Menangani anggota XCF yang macet | Mencegah satu aplikasi macet menahan sinyal seluruh sysplex |

Tanpa SFM, keputusan isolasi menunggu jawaban operator di console. Pada pukul tiga pagi, menit-menit yang hilang untuk menunggu itu adalah menit ketika seluruh sysplex tertahan. Karena itu SFM dengan isolasi otomatis dianggap praktik standar untuk sysplex produksi.

Uji perilaku SFM di lingkungan uji dengan skenario yang terkendali: sistem yang di-reset dari HMC, sistem yang dihentikan sementara, dan jalur sinyal yang diputus. Catat berapa detik sampai sistem lain kembali berjalan normal. Angka ini menjadi bagian dari perhitungan RTO untuk kegagalan satu sistem, dan membuktikan bahwa pengaturan SFM sesuai dengan yang diharapkan.

## Bab 181. Algoritma Pemeriksaan Kesehatan Sysplex Harian

```
ALGORITMA KesehatanSysplex()
  untuk setiap sistem s: pastikan status ACTIVE di D
XCF,SYSPLEX,ALL
  untuk setiap CF: pastikan terhubung ke semua sistem (D
XCF,CF)
  untuk setiap structure kritikal:
```

```
      pastikan duplex aktif bila dikonfigurasi
DUPLEX(ENABLED)
      pastikan berada di CF sesuai PREFLIST
  untuk setiap tipe CDS: pastikan primer dan alternate ada
(D XCF,COUPLE,TYPE=ALL)
  pastikan minimal dua jalur PATHIN dan PATHOUT per
pasangan sistem
  pastikan STP sinkron (D ETR) dan kebijakan SFM aktif
  periksa exception Health Checker dengan awalan XCF_,
CFRM_, XCF_SFM_
  laporkan setiap penyimpangan sebagai WARN, kehilangan
redundansi sebagai CRIT
```

Pemeriksaan ini mudah diotomasi dengan REXX (Bab 86) dan menangkap penurunan redundansi sebelum menjadi pemadaman.

## 181.1 Analisis: Dari Algoritma ke Laporan Harian

Algoritma pemeriksaan kesehatan sysplex di bab ini paling berguna bila dijalankan otomatis setiap pagi, dan hasilnya disajikan dalam laporan yang dapat dibaca dalam satu menit. Laporan yang panjang dan penuh angka akan diabaikan. Laporan yang baik hanya menonjolkan penyimpangan.

```
LAPORAN KESEHATAN SYSPLEX PLEXPRD — 2026-09-21 06:00
Status keseluruhan: WARN (1 peringatan, 0 kritis)

OK    Anggota sysplex        SYSA, SYSB aktif
OK    Coupling Facility      CF01, CF02 terhubung ke semua
sistem
OK    Structure duplex       DSNDB0G_LOCK1 duplex, GBP1
duplex
WARN  Lokasi structure       IXCPLEX_PATH2 berada di CF01,
PREFLIST meminta CF02
OK    Couple dataset         Primer dan alternate tersedia
untuk semua tipe
OK    Jalur sinyal           2 PATHIN dan 2 PATHOUT per
pasangan sistem
OK    Sinkronisasi waktu     STP sinkron
OK    Health Checker         Tidak ada exception baru
terkait XCF/CFRM

Tindakan: jalankan SETXCF START,REALLOCATE pada jendela
berikutnya (Bab 179)
```

Format seperti ini memungkinkan petugas pagi mengambil keputusan cepat: jika status OK, tidak ada yang perlu dilakukan. Jika WARN atau CRIT, baris yang bermasalah langsung terlihat bersama tindakannya.

| Keputusan desain | Rekomendasi |
|---|---|
| Kapan dijalankan | Sebelum jam kerja, dan setelah setiap perubahan sysplex |
| Siapa penerimanya | Tim sysprog; peringatan kritis juga ke petugas jaga |
| Tingkat keparahan | Kehilangan redundansi = kritis; penempatan tidak ideal = peringatan |
| Riwayat | Simpan setiap laporan; tren peringatan berulang menunjukkan masalah konfigurasi |

Simpan riwayat laporan selama beberapa bulan. Peringatan yang muncul lagi dan lagi, misalnya structure yang terus kembali ke CF yang salah setelah setiap rebuild, menunjukkan masalah di policy yang perlu diperbaiki di sumbernya.

# 🧱 Bagian XLIII — Hardening dan Monitoring Terpadu

Keamanan dan keandalan bertemu di dua tempat: konfigurasi yang dikunci dengan benar, dan data pemantauan yang sampai ke orang yang tepat pada waktu yang tepat.

## Bab 182. Checklist Hardening z/OS

| Area | Kontrol | Cara memeriksa |
|---|---|---|
| Proteksi default | `SETROPTS PROTECTALL` aktif; UACC dataset produksi `NONE` | `SETROPTS LIST`, unload `IRRDBU00` |
| Atribut berprivilegi | `SPECIAL`, `OPERATIONS`, `AUDITOR` hanya untuk nama yang disetujui | `SEARCH CLASS(USER)` dengan filter atribut |
| Library APF | `UPDATE` hanya untuk sysprog; tidak ada library APF yang tidak ada di disk | `D PROG,APF`, laporan akses RACF |
| PARMLIB, PROCLIB, LPA | Akses tulis dibatasi dan diaudit | `LISTDSD` dengan `AUDIT` |
| Superuser z/OS UNIX | Tidak ada UID 0 permanen selain yang wajib; pakai `UNIXPRIV` | Laporan segmen OMVS |
| Started task | User `PROTECTED`, terdaftar di class `STARTED` | `RLIST STARTED *` |
| Kata sandi | Password phrase, MFA untuk user berprivilegi | `SETROPTS LIST`, konfigurasi MFA |
| Perintah console | Profil `OPERCMDS` untuk perintah berbahaya (`CANCEL`, `FORCE`, `VARY`, `SET`) | `RLIST OPERCMDS *` |
| Jaringan | TLS untuk semua protokol; FTP dan TN3270 polos dimatikan | Kebijakan AT-TLS, profile TCP/IP |

| Area | Kontrol | Cara memeriksa |
|---|---|---|
| Audit | SMF 80, 81, 83, 30, 119 aktif dan dikirim ke SIEM | D SMF, laporan SIEM |
| Kriptografi | Dataset sensitif terenkripsi; master key ICSF dikelola dengan dual control | Laporan ICSF, profil dataset |

Jalankan checklist ini setiap kuartal dan setelah setiap perubahan besar. Simpan hasilnya sebagai bukti untuk audit internal dan regulator.

## 182.1 Analisis: Memprioritaskan Temuan Hardening

Checklist hardening yang dijalankan pertama kali sering menghasilkan puluhan temuan sekaligus. Tidak semua bisa diperbaiki bersamaan, dan sebagian perbaikan berisiko mengganggu produksi bila dilakukan terburu-buru. Prioritas ditentukan dari risiko setiap temuan dan usaha untuk memperbaikinya.

$$\text{Prioritas} = \frac{\text{Kemungkinan dieksploitasi} \times \text{Dampak}}{\text{Usaha perbaikan}}$$

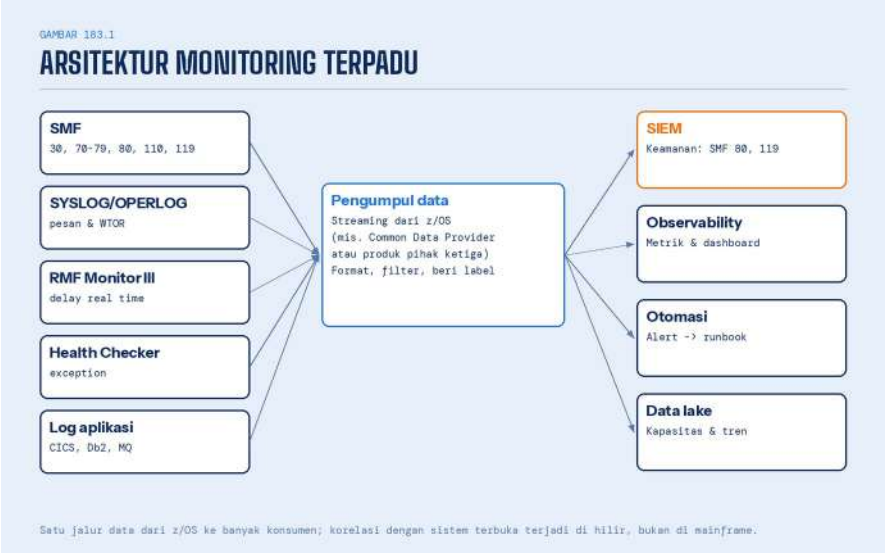
| Temuan contoh | Kemungkinan | Dampak | Usaha | Prioritas |
|---|---|---|---|---|
| Library APF menunjuk dataset yang tidak ada | Tinggi | Sangat tinggi | Rendah | Segera |
| Terlalu banyak user dengan atribut SPECIAL | Sedang | Sangat tinggi | Sedang | Tinggi |
| UACC READ pada data produksi | Tinggi | Tinggi | Sedang, perlu analisis siapa yang memakai | Tinggi |
| FTP tanpa TLS ke mitra lama | Sedang | Tinggi | Tinggi, perlu koordinasi mitra | Terencana |
| Profil OPERCMDS belum lengkap | Rendah | Sedang | Rendah | Kerjakan bersama perubahan berikutnya |

Beberapa perbaikan harus diuji dengan hati-hati. Menurunkan UACC dari READ ke NONE, misalnya, bisa membuat job produksi yang selama ini diam-diam mengandalkan akses umum tiba-tiba gagal S913. Mode WARNING membantu di sini: ubah profil ke mode WARNING dengan UACC baru selama beberapa pekan, kumpulkan pesan peringatan untuk mengetahui siapa yang terdampak, beri akses yang tepat, lalu keluarkan dari mode WARNING (Bab 137.1).

Ukur kemajuan dengan satu angka sederhana: persentase kontrol checklist yang terpenuhi, dilaporkan setiap kuartal. Angka ini mudah dipahami manajemen dan auditor, dan kenaikannya dari kuartal ke kuartal menunjukkan program hardening yang berjalan, bukan sekadar daftar temuan yang menumpuk.

## Bab 183. Arsitektur Monitoring Terpadu

Mainframe menghasilkan data pemantauan yang sangat kaya, tetapi sering terjebak di dalam z/OS. Monitoring terpadu mengalirkannya ke platform yang juga dipakai untuk sistem terbuka, sehingga satu insiden bisa dilihat dari ujung ke ujung.




*Gambar 183.1 — Arsitektur monitoring terpadu*

| Prinsip | Alasan |
|---|---|
| Satu jalur pengumpulan dari z/OS | Menghindari banyak agen yang masing-masing memakan CPU |
| Filter di sumber | Hanya data yang dipakai yang dikirim; SMF lengkap tetap diarsip di z/OS |
| Label konsisten | Nama sistem, aplikasi, dan kanal sama di semua platform agar bisa dikorelasikan |
| Waktu yang sinkron | STP di mainframe dan NTP di sistem terbuka merujuk sumber waktu yang sama |

## 183.1 Analisis: Biaya Pengumpulan Data Pemantauan

Pemantauan juga memakai sumber daya. Setiap record yang dikumpulkan, diformat, dan dikirim ke luar mainframe memakan CPU, dan pada skala besar biaya ini bisa terlihat di tagihan MSU. Arsitektur pemantauan yang baik mengumpulkan data yang dibutuhkan dengan biaya sekecil mungkin.

$$C_{\text{pemantauan}} \approx N_{\text{record per detik}} \times c_{\text{per record}} \qquad V_{\text{kirim}} = N_{\text{record}} \times \bar{s} \times f_{\text{lolos filter}}$$

f adalah porsi record yang lolos filter di sumber. Mengirim semua record SMF ke platform luar hampir tidak pernah perlu. Sebagian besar kebutuhan analitik dan keamanan terpenuhi oleh subset record tertentu, dan bahkan hanya field tertentu dari record itu.

| Keputusan | Pilihan hemat | Catatan |
|---|---|---|
| Record apa yang dikirim | Hanya tipe yang dipakai konsumen | Daftar diperbarui bersama konsumen data |
| Field apa yang dikirim | Field yang dibutuhkan, bukan record utuh | Mengurangi volume dan biaya di sisi penerima |
| Seberapa sering | Real-time untuk keamanan; interval untuk kinerja dan kapasitas | Interval RMF yang sama dengan analisis lain |
| Di mana diformat | Di platform luar bila memungkinkan | CPU sistem terbuka lebih murah untuk pekerjaan format |
| Beban di zIIP | Pilih komponen pengumpul yang eligible zIIP bila tersedia | Mengurangi dampak ke MSU (Bab 42.2) |

Ukur biaya pemantauan itu sendiri. Service class khusus untuk address space pengumpul data membuat CPU-nya terlihat terpisah di laporan RMF. Bila biaya pemantauan naik tanpa kenaikan kebutuhan, cari record atau field yang dikirim tanpa pernah dipakai.

Prinsip yang sama berlaku untuk latensi. Data keamanan harus sampai ke SIEM dalam hitungan detik atau menit agar alert berguna (Bab 184.1). Data kapasitas cukup dikirim per interval atau harian. Memperlakukan semua data seolah harus real-time adalah pemborosan yang sering terjadi.

## Bab 184. Integrasi dengan SIEM

SIEM menerima banyak sekali event. Pilih event mainframe yang benar-benar bermakna dan beri prioritas yang jelas agar tidak tenggelam.

| Event | Sumber | Prioritas |
|---|---|---|
| Pemakaian atribut `SPECIAL` atau `OPERATIONS` | SMF 80 | Tinggi |
| Perubahan APF, exit, atau PPT secara dinamis | SMF 90, SYSLOG | Tinggi |
| Lonjakan pelanggaran akses (`ICH408I`) | SMF 80 | Tinggi bila di atas ambang |
| IPL atau perubahan PARMLIB | SMF 0, 90, audit dataset | Sedang, wajib cocok dengan tiket |
| Login di luar jam atau dari alamat tak lazim | SMF 30, 119 | Sedang |
| Transfer file besar keluar | SMF 119 | Sedang |
| Revoke otomatis karena salah kata sandi | SMF 80 | Rendah, kecuali massal |

Setiap alert prioritas tinggi harus punya pemilik dan runbook. Alert tanpa tindakan yang jelas akan diabaikan, dan alert yang diabaikan lebih berbahaya daripada tidak ada alert.

## 184.1 Analisis: Aturan Korelasi untuk Event Mainframe

Satu event keamanan jarang cukup untuk menyimpulkan serangan. Pola yang mencurigakan biasanya terlihat dari urutan beberapa event yang masing-masing tampak wajar. SIEM bernilai justru karena kemampuannya mengorelasikan event-event itu.

| Aturan | Pola | Mengapa mencurigakan |
|---|---|---|
| Hak istimewa di luar jadwal | Pemakaian atribut SPECIAL atau OPERATIONS di luar jendela perubahan yang disetujui | Perubahan keamanan tanpa tiket |
| Tebak lalu berhasil | Banyak penolakan akses (`ICH408I`) dari satu user, lalu akses berhasil ke resource yang sama | Pencarian celah yang berhasil |
| Ubah lalu jalankan | Perubahan library APF atau PROGxx, diikuti job dari user yang sama dalam waktu singkat | Menanam program authorized |
| Keluar data malam | Transfer keluar berukuran besar di luar jam operasional, dari user yang biasanya tidak melakukannya | Eksfiltrasi data |
| Akun tidur bangun | Login dari user yang tidak aktif berbulan-bulan | Akun lama yang disalahgunakan |

```
ATURAN TebakLaluBerhasil
  JIKA  jumlah(SMF80 penolakan, user = u, resource = r) >=
5 dalam 10 menit
  DAN   ada SMF80 akses berhasil, user = u, resource = r,
dalam 30 menit berikutnya
  MAKA  alert prioritas tinggi: "kemungkinan akses setelah
percobaan berulang"
        lampirkan: urutan event, profil RACF r, perubahan
profil r dalam 24 jam terakhir
```

Lampiran pada alert sama pentingnya dengan aturannya. Alert yang langsung menyertakan konteks, seperti profil yang terlibat dan perubahan terakhir, menghemat banyak waktu analis. Analis tidak perlu masuk ke mainframe dan menjalankan perintah satu per satu.

Mulailah dengan tiga sampai lima aturan yang paling relevan, setel ambangnya dari data beberapa pekan, lalu tambah secara bertahap. Setiap aturan diuji dengan skenario simulasi sebelum diaktifkan, agar diketahui bahwa aturan itu benar-benar berbunyi saat pola terjadi.

## Bab 185. Metrik Observability untuk Mainframe

Empat sinyal emas yang dipakai di dunia observability juga berlaku untuk mainframe, dengan sumber data yang berbeda.

| Sinyal | Makna di IBM Z | Sumber |
|---|---|---|
| Latency | Waktu respons transaksi CICS dan API | SMF 110, z/OS Connect |
| Traffic | Transaksi per detik per kanal | SMF 110, statistik CICS |
| Errors | Abend transaksi, job gagal, SQLCODE negatif | SMF, log aplikasi |
| Saturation | CPU, PI WLM, spool, storage group, kedalaman antrean | RMF, SMF 70-79, perintah console |

Target layanan dirumuskan sebagai SLO, dan selisihnya dari 100% adalah anggaran kesalahan yang boleh dipakai untuk perubahan dan risiko:

$$\text{Anggaran kesalahan} = (1 - SLO) \times \text{jumlah permintaan dalam periode}$$

Contoh: SLO 99,9% untuk 30 juta transaksi sebulan berarti maksimal 30.000 transaksi boleh gagal atau melewati batas waktu. Jika anggaran hampir habis di pertengahan bulan, tunda perubahan berisiko sampai periode berikutnya.

### 185.1 Analisis: Alert Berbasis Laju Pemakaian Anggaran Kesalahan

Anggaran kesalahan (Bab 185) memberi batas berapa banyak kegagalan yang dapat diterima dalam sebulan. Pertanyaan berikutnya adalah kapan harus membangunkan petugas jaga. Alert pada setiap kegagalan terlalu bising. Alert saat anggaran sudah habis terlambat. Jalan tengahnya adalah memantau laju pemakaian anggaran, yang disebut *burn rate*.

$$\text{Burn rate} = \frac{\text{tingkat kesalahan saat ini}}{1 - SLO}$$

Burn rate 1 berarti anggaran habis tepat di akhir periode. Burn rate 10 berarti anggaran sebulan habis dalam sekitar tiga hari. Dengan SLO 99,9%, tingkat kesalahan 1,44% selama satu jam berarti burn rate 14,4, dan dalam satu jam itu sekitar 2% anggaran bulanan sudah terpakai.

| Jendela pengamatan | Ambang burn rate | Tindakan | Alasan |
|---|---|---|---|
| 1 jam, dikonfirmasi 5 menit | Sekitar 14 | Panggil petugas jaga segera | Masalah serius yang cepat menghabiskan anggaran |
| 6 jam, dikonfirmasi 30 menit | Sekitar 6 | Panggil petugas jaga | Masalah sedang yang bertahan lama |
| 3 hari | Sekitar 1 | Tiket untuk ditindaklanjuti di jam kerja | Penurunan kualitas yang lambat |

Jendela konfirmasi yang lebih pendek mencegah alert terus berbunyi setelah masalah selesai. Dengan pendekatan ini, gangguan singkat yang langsung pulih tidak membangunkan siapa pun, tetapi gangguan yang nyata tetap terdeteksi dalam hitungan menit.

Pola ini paling cocok untuk layanan dengan volume transaksi tinggi dan SLO yang jelas, seperti API saldo dan transfer. Untuk layanan bervolume rendah, satu kegagalan saja bisa menghasilkan burn rate besar, sehingga ambangnya perlu disesuaikan atau digabung dengan jumlah kegagalan minimum.

# 🧩 Bagian XLIV — Studi Kasus Troubleshooting

Dua puluh kasus berikut ringkas dan fiktif, disusun dari pola yang sering terjadi. Tutup kolom akar masalah, baca gejalanya, lalu coba tebak sebelum membaca jawabannya.

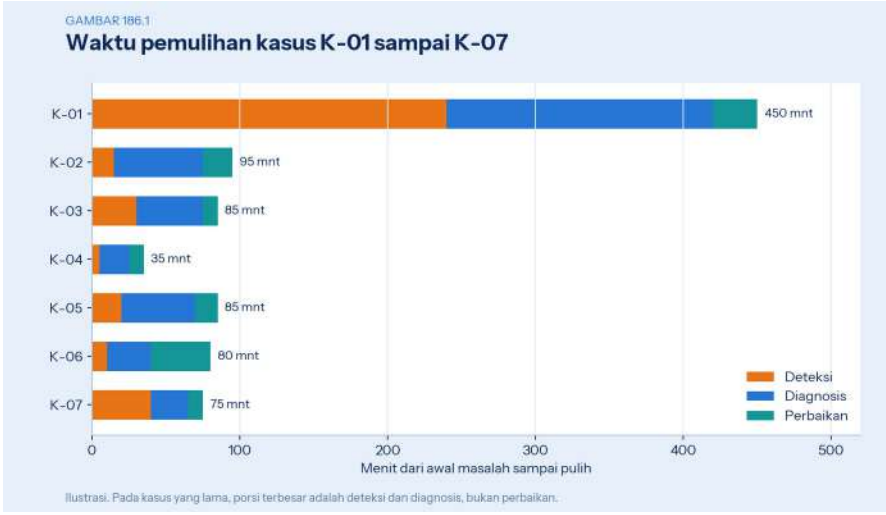
## Bab 186. Kasus Storage, Batch, dan Aplikasi

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-01 | Satu job EOD dua kali lebih lambat sejak pekan lalu | Volume baru ditambahkan ke storage group tanpa alias HyperPAV | Setiap volume baru harus melalui checklist konfigurasi lengkap |
| K-02 | CICS SOS setiap pukul 10.00 | Program memuat tabel referensi besar ke storage setiap transaksi | Tabel referensi dimuat sekali ke shared storage, bukan per task |
| K-03 | Job batch tidak pernah selesai, CPU tinggi | Kondisi akhir file diperiksa setelah READ yang salah, sehingga loop tanpa akhir | Parameter `TIME` realistis sebagai jaring pengaman |
| K-04 | IPL memakan waktu 20 menit lebih lama | CLPA dilakukan setiap IPL padahal LPA tidak berubah | Tentukan jenis IPL dari daftar perubahan, bukan kebiasaan |
| K-05 | Transaksi online timeout saat malam | REORG dengan `SHRLEVEL REFERENCE` dijadwalkan bersamaan dengan stand-in | Utilitas malam harus memperhitungkan beban stand-in |
| K-06 | Transfer file ke mitra gagal sejak pagi | Mitra mengganti sertifikat dengan CA perantara baru | Konfirmasi perubahan sertifikat mitra lewat prosedur perubahan bersama |

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-07 | Spool naik 20% dalam satu jam | `DISPLAY` debug tertinggal di program yang dipromosikan ke produksi | Code review memeriksa `DISPLAY` di loop |

## 186.1 Analisis: Ke Mana Waktu Pemulihan Pergi

Tujuh kasus pertama dapat dilihat dari sisi lain: bukan hanya apa penyebabnya, tetapi berapa lama waktu yang dibutuhkan untuk pulih, dan di tahap mana waktu itu habis (Bab 57.1).




*Gambar 186.1 — Waktu pemulihan kasus K-01 sampai K-07*

Kasus K-01, job EOD yang melambat karena volume tanpa alias HyperPAV, memakan waktu paling lama. Waktunya bukan habis untuk perbaikan, yang hanya setengah jam, melainkan untuk deteksi dan diagnosis. Perlambatan terjadi bertahap selama berhari-hari, tidak ada alert yang berbunyi, dan saat akhirnya diselidiki, penyebabnya harus dicari dari nol.

| Pola | Kasus | Perbaikan proses |
|---|---|---|
| Deteksi lama | K-01, K-07 | Alert berdasarkan tren dan baseline, bukan hanya ambang tetap |

| Pola | Kasus | Perbaikan proses |
|---|---|---|
| Diagnosis lama | K-01, K-02, K-05 | Algoritma diagnosis top-down (Bab 62.1) dan kamus pesan (Bab 50.1) |
| Perbaikan lama | K-06 | Koordinasi dengan pihak luar, seperti mitra atau vendor, yang sudah disiapkan sebelumnya |
| Cepat di semua tahap | K-04 | Masalah yang dikenal dengan runbook yang jelas |

Kasus K-04 memberi pelajaran sebaliknya. Masalahnya dikenal, runbook-nya jelas, dan pemulihan selesai dalam setengah jam lebih sedikit. Semakin banyak jenis masalah yang punya runbook teruji, semakin banyak insiden yang berakhir seperti K-04.

Mengumpulkan komponen waktu seperti ini dari setiap laporan pasca-insiden, lalu menggambarkannya bersama setiap kuartal, memberi tim gambaran yang jelas tentang di mana investasi berikutnya paling berguna: di pemantauan, di dokumentasi diagnosis, atau di koordinasi dengan pihak luar.

## Bab 187. Kasus Operasi dan Konfigurasi

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-08 | Aplikasi Liberty gagal menulis log | zFS penuh karena log tidak dirotasi | Rotasi log dan `aggrgrow` untuk semua zFS aplikasi |
| K-09 | Rantai job EOD tertahan 40 menit | WTOR mount tape menunggu tanpa ada yang menjawab | Otomasi alert WTOR yang berumur lebih dari 5 menit |
| K-10 | Banyak job lambat membuka dataset | Skrip ad hoc menjalankan `LISTCAT` massal saat jam sibuk | Laporan katalog massal dijadwalkan di luar jam sibuk |
| K-11 | CPU batch naik 15% setelah maintenance Db2 | REBIND tanpa `APREUSE` mengubah access path | Plan management dan pembandingan `PLAN_TABLE` |

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-12 | Transaksi teller satu wilayah lambat | Prefix transaksi baru tidak ada di aturan klasifikasi WLM | Perubahan WLM wajib di setiap rilis yang menambah transaksi |
| K-13 | Selisih GL valas setiap pagi | Revaluasi berjalan sebelum job kurs dari treasury selesai | Dependensi eksplisit di scheduler |
| K-14 | `S806` setelah deploy | Load library baru tidak ditambahkan ke `STEPLIB` di semua PROC | Deploy memakai daftar PROC yang dihasilkan otomatis |

## 187.1 Analisis: Insiden yang Berakar pada Perubahan

Kasus K-08 sampai K-14 memperlihatkan pola yang sama dengan analisis Pareto di Bab 189: sebagian besar berakar pada perubahan. Ada rebind tanpa perlindungan access path, transaksi baru tanpa aturan WLM, deploy tanpa library lengkap, dan jadwal baru tanpa dependensi. Setiap perubahan itu masuk akal sendiri-sendiri. Yang terlewat adalah dampaknya pada bagian lain sistem.

$$\text{Tingkat insiden perubahan} = \frac{\text{insiden yang disebabkan perubahan}}{\text{total perubahan}}$$

Angka ini, yang sama dengan tingkat kegagalan perubahan di Bab 55.1, sebaiknya dipantau per jenis perubahan. Jenis perubahan dengan tingkat insiden tinggi membutuhkan pemeriksaan tambahan yang spesifik, bukan birokrasi tambahan untuk semua perubahan.

| Jenis perubahan | Pertanyaan wajib di tiket | Kasus yang dicegah |
|---|---|---|
| Rilis aplikasi dengan transaksi baru | Apakah aturan klasifikasi WLM sudah diperbarui? | K-12 |
| Maintenance Db2 dengan rebind | Apakah access path dilindungi atau dibandingkan sebelum dan sesudah? | K-11 |
| Deploy load module | Apakah semua PROC dan library terkait diperbarui dan | K-14 |

| Jenis perubahan | Pertanyaan wajib di tiket | Kasus yang dicegah |
|---|---|---|
| | diverifikasi otomatis? | |
| Job atau jadwal baru | Apakah dependensi eksplisit sudah didefinisikan di scheduler? | K-13 |
| Skrip atau laporan ad hoc di produksi | Apakah dampaknya ke katalog dan I/O sudah dinilai, dan dijadwalkan di luar jam sibuk? | K-10 |

Pertanyaan-pertanyaan ini sebaiknya muncul otomatis di formulir tiket sesuai jenis perubahan, bukan diingat oleh peninjau. Setiap insiden baru yang berakar pada perubahan menambah satu pertanyaan ke daftar jenis perubahan yang terkait. Dengan begitu, proses perubahan belajar dari insiden secara sistematis, dan setiap pelajaran menjadi kontrol yang nyata (Bab 189.1).

## Bab 188. Kasus Kapasitas, Pemantauan, dan Keamanan

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-15 | HyperSwap tidak terjadi saat disk primer bermasalah | Pasangan volume sudah suspended berhari-hari tanpa alert | Alert status pasangan replikasi setara pentingnya dengan alert disk |
| K-16 | Operator tidak bisa login TN3270 | Sertifikat server TN3270 kedaluwarsa | Semua sertifikat masuk inventaris dengan alert (Bab 149) |
| K-17 | Kekurangan page dataset setiap Jumat | Address space monitoring bocor memori dan tidak pernah di-restart | Tren pemakaian memori per address space, bukan hanya total |
| K-18 | Rerun EOD gagal karena generasi GDG hilang | `LIMIT` GDG terlalu kecil untuk rerun beberapa hari | Batas GDG disesuaikan dengan kebutuhan rerun terburuk |

| Kasus | Gejala | Akar masalah | Pelajaran |
|---|---|---|---|
| K-19 | Stempel waktu transaksi di situs DR bergeser | Pengaturan zona waktu di `CLOCKxx` situs DR berbeda | Konfigurasi DR dibandingkan otomatis dengan situs utama |
| K-20 | Job batch malam gagal `S913` | Kata sandi user batch kedaluwarsa karena tidak ditandai `PROTECTED` | User batch dan started task selalu `PROTECTED` |

## 188.1 Analisis: Hal-Hal yang Kedaluwarsa atau Tumbuh Diam-Diam

Kasus K-15 sampai K-20 punya benang merah yang sama. Sesuatu kedaluwarsa atau tumbuh perlahan tanpa ada yang memperhatikan: pasangan replikasi yang suspended, sertifikat yang habis masa berlakunya, memori yang terus terpakai, batas GDG yang terlalu kecil, dan kata sandi user batch yang kedaluwarsa. Semua ini punya tanggal atau tren yang bisa diketahui jauh sebelumnya.

Pencegahannya adalah satu inventaris terpadu untuk semua hal yang punya batas waktu atau batas kapasitas, dipantau dari satu tempat.

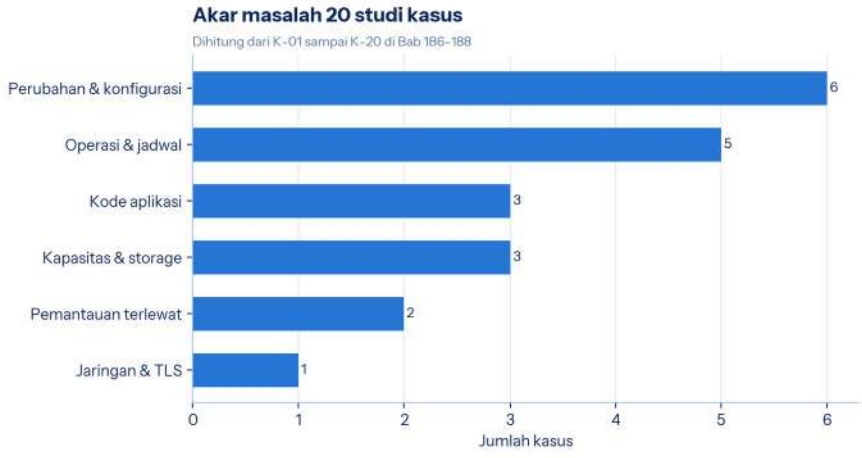
| Jenis | Contoh | Yang dipantau | Kasus |
|---|---|---|---|
| Batas waktu | Sertifikat, kata sandi user teknis, kontrak dukungan, kunci lisensi produk | Tanggal kedaluwarsa | K-16, K-20 |
| Batas jumlah | `LIMIT` GDG, `MAXDEPTH` antrean, jumlah extent | Jarak dari batas dibanding kebutuhan terburuk | K-18 |
| Tren pertumbuhan | Common storage, memori address space, storage group, log | Garis dasar dan laju pertumbuhan | K-17 |
| Status yang bisa berubah diam-diam | Pasangan replikasi, HyperSwap aktif, jalur sinyal XCF | Status saat ini dibanding status yang diharapkan | K-15 |

```
ALGORITMA InventarisBatas(daftar_item)
  untuk setiap item i:
      jika i punya tanggal kedaluwarsa: sisa <- tanggal -
hari ini
      jika i punya batas jumlah: sisa <- (batas -
pemakaian) / laju pertumbuhan
      jika i punya status yang diharapkan: sisa <- 0 bila
status tidak sesuai
      bandingkan sisa dengan waktu tenggang item i (Bab
149.2)
      urutkan dari sisa terkecil; kirim daftar kepada
pemilik masing-masing
```

Nilai inventaris ini ada pada penyatuannya. Sertifikat dipantau satu tim, GDG oleh tim lain, dan replikasi oleh tim storage. Masing-masing mungkin punya catatannya sendiri, tetapi tidak ada yang melihat gambaran utuhnya. Satu laporan mingguan yang memuat semua item dengan sisa waktu terkecil membuat hal-hal yang kedaluwarsa diam-diam menjadi terlihat bersama-sama.

## Bab 189. Pola dari Dua Puluh Kasus

Perubahan dan konfigurasi menyumbang porsi terbesar akar masalah, disusul operasi dan penjadwalan. Kode aplikasi dan kapasitas, yang sering dianggap penyebab utama, justru lebih jarang.




*Dihitung dari K-01 sampai K-20*

- Sebagian besar kasus bisa dicegah dengan checklist perubahan yang lengkap dan perbandingan konfigurasi otomatis.
- Hampir setiap kasus meninggalkan jejak yang sudah terlihat di data pemantauan sebelum menjadi insiden.
- Pelajaran tiap kasus harus berakhir sebagai kontrol baru: alert, langkah checklist, atau runbook.

## 189.1 Analisis: Sinyal Dini yang Terlewat

Pelajaran paling penting dari dua puluh studi kasus bukanlah akar masalahnya, tetapi kenyataan bahwa hampir setiap insiden memberi sinyal jauh sebelum terjadi. Sinyal itu ada di data yang sudah dikumpulkan, tetapi tidak dipantau atau tidak memicu tindakan.


*Gambar 189.1 — Sinyal dini sebelum insiden*

| Kasus | Sinyal yang sudah ada | Aturan yang akan menangkapnya |
|---|---|---|
| K-01 | IOSQ volume naik sejak volume baru ditambahkan | Waktu respons per volume dibanding baseline (Bab 6.3) |
| K-12 | PI service class satu wilayah di atas 1 | PI per dimensi bisnis (Bab 119.1) |
| K-15 | Status pasangan replikasi | Pemeriksaan harian status |

| Kasus | Sinyal yang sudah ada | Aturan yang akan menangkapnya |
|---|---|---|
| | suspended | replikasi (Bab 44.1) |
| K-16 | Tanggal kedaluwarsa sertifikat | Kalender kedaluwarsa dan ambang per jenis (Bab 149.2) |
| K-17 | Garis dasar memori harian address space terus naik | Tren garis dasar, bukan puncak (Bab 10.4) |
| K-18 | Batas GDG lebih kecil dari kebutuhan rerun | Tinjauan konfigurasi GDG EOD (Bab 133.1) |

Sebagian besar sinyal ini terlihat berhari-hari, bahkan berpekan-pekan, sebelum insiden. Artinya, yang kurang bukan data atau alat, melainkan aturan sederhana yang membaca data itu dan tindakan yang mengikutinya.

Setiap postmortem sebaiknya menjawab satu pertanyaan tambahan: kapan sinyal pertama terlihat, dan aturan apa yang akan menangkapnya? Jawaban itu menjadi tindakan perbaikan yang paling berharga (Bab 175.1). Setelah beberapa tahun, kumpulan aturan seperti ini membentuk sistem peringatan dini yang disesuaikan dengan kelemahan nyata lingkungan Anda sendiri.

## 189.2 Kasus Tambahan: Db2, CICS, dan MQ (K-21 s.d. K-27)

Dua puluh kasus berikut melanjutkan Bab 186–188 dalam format yang lebih lengkap: gejala, investigasi, akar masalah, perbaikan, dan pelajaran. Seperti sebelumnya, semua kasus fiktif tetapi disusun dari pola yang sering terjadi. Setiap kasus merujuk subbab analisis yang relevan, sehingga bisa dipakai sebagai latihan: baca gejalanya, tebak penyebabnya, lalu bandingkan.

### K-21 — Timeout teller saat posting bunga akhir bulan

*Gejala*. Pukul 21.30 pada akhir bulan, transaksi teller di cabang yang masih buka mulai timeout dengan `SQLCODE -911`. Pada hari biasa, tidak pernah terjadi.

*Investigasi.* Laporan Db2 menunjukkan lock escalation pada tablespace saldo, dipicu job posting bunga akhir bulan yang mulai pukul 21.00.

*Akar masalah.* Rilis terbaru job posting bunga mengubah frekuensi commit dari setiap 500 rekening menjadi setiap 50.000 rekening untuk "mempercepat" job. Jumlah lock per unit kerja melewati batas, dan Db2 menaikkan lock ke level tablespace.

*Perbaikan.* Frekuensi commit dikembalikan, kali ini sebagai parameter yang dapat diubah tanpa kompilasi ulang.

*Pelajaran.* Frekuensi commit adalah keputusan desain, bukan tuning bebas. Ukur dampaknya terhadap lock dan transaksi online yang berjalan bersamaan (Bab 70.3).

**K-22 — Kueri cepat menjadi lambat setelah RUNSTATS**

*Gejala.* Pagi hari setelah pemeliharaan statistik, satu transaksi inquiry memakai CPU puluhan kali lipat dari biasanya.

*Investigasi.* `EXPLAIN` menunjukkan tablespace scan pada tabel staging yang biasanya berisi ratusan ribu baris (Bab 33.4).

*Akar masalah.* RUNSTATS dijalankan tepat setelah tabel staging dikosongkan oleh job pembersihan. Statistik mencatat tabel kosong, sehingga optimizer memilih scan karena menganggap tabel itu kecil. Beberapa jam kemudian tabel kembali terisi penuh.

*Perbaikan.* Tabel yang isinya naik-turun tajam dalam sehari ditandai volatil agar optimizer lebih memilih index. Jadwal RUNSTATS disesuaikan dengan siklus isi tabel.

*Pelajaran.* Statistik yang akurat untuk saat yang salah sama menyesatkannya dengan statistik yang salah.

**K-23 — SOS di satu AOR setiap sore**

*Gejala.* Setelah rilis aplikasi pembiayaan, satu AOR mengalami short on storage setiap sore sekitar pukul 16.00, lalu pulih setelah di-restart.

*Investigasi.* Tren EDSA harian menunjukkan garis dasar yang naik terus sejak rilis (Bab 67.3). Statistik storage per program menunjukkan satu program yang memperoleh shared storage di setiap transaksi.

*Akar masalah.* Program baru meminta shared storage untuk cache sementara dan tidak pernah melepaskannya.

*Perbaikan.* Program diperbaiki. Untuk sementara, restart AOR terjadwal di malam hari sampai perbaikan terpasang.

*Pelajaran.* Aplikasi baru sebaiknya berjalan di AOR terpisah pada bulan-bulan pertama, agar kebocoran seperti ini tidak menjatuhkan transaksi lain (Bab 63.3).

**K-24 — Penghitung referensi yang meloncat**

*Gejala.* Sejak program pembuat nomor referensi dijadikan threadsafe, sesekali muncul dua transaksi dengan nomor referensi yang sama.

*Investigasi.* Program menaikkan penghitung di CWA tanpa kunci. Sebelumnya, di QR TCB, dua task tidak pernah berjalan bersamaan. Setelah threadsafe, dua task di open TCB bisa membaca nilai yang sama sebelum salah satunya menulis.

*Akar masalah.* Pemeriksaan storage bersama di daftar periksa threadsafe terlewat (Bab 65.4).

*Perbaikan.* Penghitung dipindahkan ke mekanisme yang aman untuk akses bersamaan.

*Pelajaran.* Kesalahan threadsafe jarang muncul di uji fungsi. Kesalahan ini hanya muncul di bawah beban bersamaan, jadi tinjauan kode tidak bisa dilewati.

**K-25 — Antrean transmisi menumpuk setelah mitra memperbarui sertifikat**

*Gejala.* Pesan ke payment hub menumpuk di transmission queue. Transfer keluar tertunda sejak pukul 08.00.

*Investigasi.* Channel sender ke payment hub berulang kali gagal membuka sesi. Log menunjukkan penolakan di pemetaan identitas.

*Akar masalah.* Payment hub memasang sertifikat baru dengan nama distinguished yang sedikit berbeda. Aturan SSLPEERMAP di queue manager bank masih mencocokkan nama lama (Bab 166.1).

*Perbaikan.* Aturan diperbarui, lalu channel di-restart. Antrean terkuras dalam sekitar 40 menit (Bab 34.1).

*Pelajaran.* Pembaruan sertifikat mitra harus melalui prosedur perubahan bersama yang mencakup semua aturan yang merujuk nama sertifikat, bukan hanya penggantian sertifikatnya.

**K-26 — Pesan masuk dead-letter queue setelah mitra berganti encoding**

*Gejala.* Sebagian pesan dari mitra switching berakhir di dead-letter queue dengan kode alasan konversi.

*Investigasi.* Header pesan menunjukkan CCSID yang berbeda dari sebelumnya. Mitra memindahkan aplikasinya ke platform baru yang mengirim UTF-8, tetapi field format pesan tidak diperbarui.

*Akar masalah.* Perubahan encoding di sisi mitra tidak dikomunikasikan, dan metadata pesan tidak konsisten dengan isinya (Bab 145.1).

*Perbaikan.* Mitra memperbaiki metadata pesan. Pesan di dead-letter queue diproses ulang setelah diverifikasi.

*Pelajaran.* Spesifikasi interface harus mencantumkan CCSID secara eksplisit, dan dead-letter queue perlu dipantau dengan alert, bukan diperiksa sesekali.

**K-27 — API saldo ditolak saat puncak gajian**

*Gejala.* Pada hari gajian, sebagian permintaan API saldo dari aplikasi mobile ditolak pada jam puncak. CPU mainframe tidak penuh.

*Investigasi.* Db2 mencatat batas thread DDF tercapai. Sebagian besar thread aktif ternyata diam, tanpa aktivitas.

*Akar masalah.* Connection pool di gateway API dikonfigurasi menahan koneksi tanpa melepasnya ke mode inaktif. Pada puncak, pool membuka koneksi baru, sementara koneksi lama tetap memegang thread (Bab 68.2).

*Perbaikan.* Konfigurasi pool di gateway diperbaiki, dan mode thread inaktif DDF dipastikan aktif.

*Pelajaran.* Saat batas thread tercapai sementara CPU masih longgar, masalahnya hampir selalu di pengelolaan koneksi, bukan kapasitas.

## 189.3 Kasus Tambahan: EOD, Interface, dan Keamanan (K-28 s.d. K-34)

**K-28 — Bunga tercatat dua kali setelah rerun**

*Gejala.* Rekonsiliasi pagi menemukan akrual bunga deposito untuk sekitar 40 ribu rekening tercatat dua kali.

*Investigasi.* Fase akrual abend karena ruang log hampir penuh, tepat setelah semua akrual ditulis tetapi sebelum status fase diperbarui menjadi `SELESAI`. Operator menjalankan ulang fase sesuai runbook.

*Akar masalah.* Fase tidak memeriksa apakah akrual untuk tanggal bisnis yang sama sudah ada sebelum menulis. Ini adalah skenario C di Bab 136.1, yang belum pernah diuji.

*Perbaikan.* Akrual ganda dibatalkan dengan jurnal koreksi yang ditinjau. Fase diubah agar melewati rekening yang sudah punya akrual untuk tanggal bisnis itu.

*Pelajaran.* Matriks uji restart (Bab 58.1) harus mencakup kegagalan tepat sebelum status ditandai, karena titik inilah yang paling mudah terlewat.

**K-29 — File mitra terpotong diproses sebagian**

*Gejala.* Nasabah melaporkan transfer masuk dari satu bank mitra tidak masuk ke rekening.

*Investigasi.* File dari mitra hanya berisi sekitar 60% dari jumlah record yang biasa. Transfer file terputus di tengah jalan, dan file yang terpotong tetap diproses.

*Akar masalah.* Validasi hanya memeriksa format setiap record, bukan keberadaan dan kecocokan trailer. File terpotong tidak punya trailer, tetapi tetap lolos (Bab 147.1).

*Perbaikan.* File diminta ulang dan diproses lengkap. Validasi struktural ditambahkan: tanpa trailer yang cocok, file ditolak seluruhnya.

*Pelajaran.* Kesalahan struktural harus selalu berarti tolak seluruhnya. Validasi per record tidak cukup untuk menangkap file yang tidak lengkap.

**K-30 — Ribuan selisih setelah rekonsiliasi dipindah**

*Gejala.* Setelah proses rekonsiliasi dipindahkan ke platform Linux, laporan pertamanya menunjukkan puluhan ribu selisih.

*Investigasi.* Selisih hampir seluruhnya berupa pasangan “tidak ditemukan” di kedua sisi, padahal datanya sama.

*Akar masalah.* File dari mainframe terurut berdasarkan EBCDIC, sedangkan program match-merge di Linux mengasumsikan urutan ASCII (Bab 146.1).

*Perbaikan.* File diurutkan ulang di sisi penerima sebelum dicocokkan.

*Pelajaran.* Setiap proses yang bergantung pada urutan harus mengurutkan ulang atau menyepakati urutan secara eksplisit saat berpindah platform.

**K-31 — Logstream SMF hampir penuh setelah audit diperluas**

*Gejala.* Peringatan logstream SMF mendekati penuh, dua kali dalam sepekan.

*Investigasi.* Volume SMF 80 naik sekitar sepuluh kali lipat sejak pekan lalu.

*Akar masalah.* Untuk menjawab permintaan audit, opsi audit baca diaktifkan pada profil generik yang melindungi hampir semua dataset produksi, bukan hanya dataset sensitif (Bab 30.1).

*Perbaikan.* Audit baca dipersempit ke profil data sensitif. Jadwal pembongkaran SMF dengan `IFASMFDL` dipercepat sementara (Bab 170.1).

*Pelajaran.* Perubahan opsi audit adalah perubahan kapasitas. Perkirakan volumenya sebelum diaktifkan.

**K-32 — Pengembang masih bisa mengubah library produksi**

*Gejala.* Audit tahunan menemukan dua pengembang dengan akses `UPDATE` ke load library produksi.

*Investigasi.* Keduanya dulu anggota tim dukungan produksi, lalu pindah ke tim pengembangan setahun sebelumnya.

*Akar masalah.* Proses pindah jabatan hanya menambahkan group baru, tanpa mencabut group lama (Bab 28.2).

*Perbaikan.* Akses dicabut. Log SMF 80 setahun terakhir diperiksa, dan tidak ditemukan pemakaian akses itu.

*Pelajaran.* Proses pindah jabatan harus mencabut akses lama secara eksplisit, dan recertification berkala menangkap yang lolos.

**K-33 — CPU naik setelah mitra mengubah klien**

*Gejala.* CPU address space TCP/IP dan aplikasi gateway naik tajam sejak satu mitra fintech merilis aplikasi versi baru.

*Investigasi.* Statistik AT-TLS menunjukkan jumlah handshake penuh hampir sama dengan jumlah transaksi dari mitra itu (Bab 148.1).

*Akar masalah.* Klien versi baru membuka koneksi TLS baru untuk setiap permintaan dan tidak memakai session resumption.

*Perbaikan.* Mitra diminta memakai koneksi persisten. Sementara itu, batas laju per mitra di gateway diperketat.

*Pelajaran.* Perilaku koneksi klien adalah bagian dari kontrak interface, sama pentingnya dengan format pesan.

**K-34 — Bunga cuti bersama terhitung kurang**

*Gejala.* Setelah cuti bersama tiga hari, nasabah deposito mengeluh bunga yang diterima lebih kecil dari seharusnya.

*Investigasi.* EOD setelah libur hanya menghitung bunga untuk satu hari kalender, bukan tiga.

*Akar masalah.* Cuti bersama diumumkan setelah kalender tahun itu dimuat, dan tabel kalender diperbarui dengan menandai hari libur tanpa memperbarui penanda jumlah hari kalender yang diproses (Bab 106.1).

*Perbaikan.* Bunga dikoreksi untuk semua rekening terdampak. Prosedur perubahan kalender darurat ditambah langkah verifikasi dengan simulasi EOD di UAT.

*Pelajaran.* Perubahan kalender adalah perubahan aplikasi, dan harus diuji seperti perubahan aplikasi.

## 189.4 Kasus Tambahan: Sysplex, DR, dan Operasi (K-35 s.d. K-40)

**K-35 — Structure tidak kembali ke CF pilihannya**

*Gejala.* Laporan kesehatan sysplex menandai peringatan penempatan structure selama tiga pekan berturut-turut (Bab 181.1).

*Investigasi.* Setelah pemeliharaan CF02, beberapa structure dipindah ke CF01 dan tidak pernah dikembalikan. Lock structure Db2 kini berjalan simplex di CF01, karena duplexing tidak dibangun ulang setelah CF02 kembali, sehingga kegagalan CF01 akan memaksa rebuild tanpa salinan.

*Akar masalah.* Langkah penempatan ulang structure tidak ada di runbook pemeliharaan CF.

*Perbaikan.* `SETXCF START,REALLOCATE` dijalankan di jendela berikutnya, dan langkah itu ditambahkan ke runbook.

*Pelajaran.* Peringatan yang muncul berulang harus punya pemilik dan tenggat. Peringatan tiga pekan yang tidak ditindaklanjuti bukan lagi peringatan, tetapi risiko yang diterima tanpa keputusan.

**K-36 — Sistem yang sehat dikeluarkan dari sysplex**

*Gejala.* SYSB dikeluarkan dari sysplex oleh SFM pada jam kerja. Layanan beralih ke SYSA, tetapi sempat melambat.

*Investigasi.* Link ke CF01 sempat terputus beberapa detik. SYSB tidak punya jalur sinyal lain, karena jalur lewat CF02 tidak diaktifkan kembali setelah pemeliharaan CF02 pekan sebelumnya (Bab 178.1).

*Akar masalah.* Redundansi jalur sinyal hilang tanpa terdeteksi.

*Perbaikan.* Jalur sinyal lewat CF02 diaktifkan kembali. Pemeriksaan jumlah jalur per pasangan sistem ditambahkan ke laporan kesehatan harian.

*Pelajaran.* Setelah setiap pemeliharaan, redundansi harus dibuktikan kembali, bukan diasumsikan.

**K-37 — Data di situs DR tidak bisa dibaca**

*Gejala.* Pada uji DR, sistem di situs DR menyala normal, tetapi aplikasi gagal membuka dataset terenkripsi.

*Investigasi.* ICSF di situs DR tidak dapat membuka CKDS, karena master key di kartu Crypto Express situs DR tidak sama dengan yang dipakai situs utama.

*Akar masalah.* Master key situs utama diganti enam bulan sebelumnya, tetapi prosedur pergantian tidak mencakup kartu di situs DR (Bab 31.1).

*Perbaikan.* Master key dimuat di situs DR dengan dual control. Prosedur pergantian master key diperbarui agar mencakup semua situs.

*Pelajaran.* Uji DR harus selalu mencakup pembacaan data terenkripsi. Menyalakan sistem saja tidak membuktikan data bisa dipakai.

**K-38 — Volume offline di situs DR**

*Gejala.* Pada uji DR, sebagian volume data tidak dapat diaktifkan di situs DR.

*Investigasi.* Control unit storage baru yang dipasang empat bulan sebelumnya tidak terdefinisi di IODF situs DR.

*Akar masalah.* Perubahan IODF di situs utama tidak disertai perubahan yang sama untuk situs DR (Bab 13.1).

*Perbaikan.* IODF situs DR diperbarui. Perbandingan IODF kedua situs ditambahkan ke deteksi drift (Bab 15.1).

*Pelajaran.* Setiap perubahan hardware di situs utama adalah dua perubahan: satu di situs utama, satu di situs DR.

**K-39 — Otomasi menjawab WTOR yang salah**

*Gejala.* Job batch yang membutuhkan tape tertentu malah memakai tape lain, dan hasilnya tertulis di atas data yang masih dibutuhkan.

*Investigasi.* Aturan otomasi menjawab WTOR mount berdasarkan ID pesan saja, tanpa memeriksa nama job. Aturan itu dibuat untuk satu job rutin, tetapi ikut menjawab WTOR job lain dengan ID yang sama.

*Akar masalah.* Pencocokan aturan otomasi terlalu longgar (Bab 85.1).

*Perbaikan.* Aturan dipersempit ke nama job tertentu. Data dipulihkan dari salinan cadangan.

*Pelajaran.* Setiap aturan otomasi harus menjawab pertanyaan “kapan aturan ini tidak boleh bertindak?”

**K-40 — Kueri analitik memicu capping di jam sibuk**

*Gejala.* Pukul 11.00, transaksi online melambat tiba-tiba, dan RMF menunjukkan LPAR sedang dicapping.

*Investigasi.* Seorang analis menjalankan kueri ad hoc besar lewat DDF di jam sibuk. Kueri itu mendorong 4HRA melewati defined capacity (Bab 42.2).

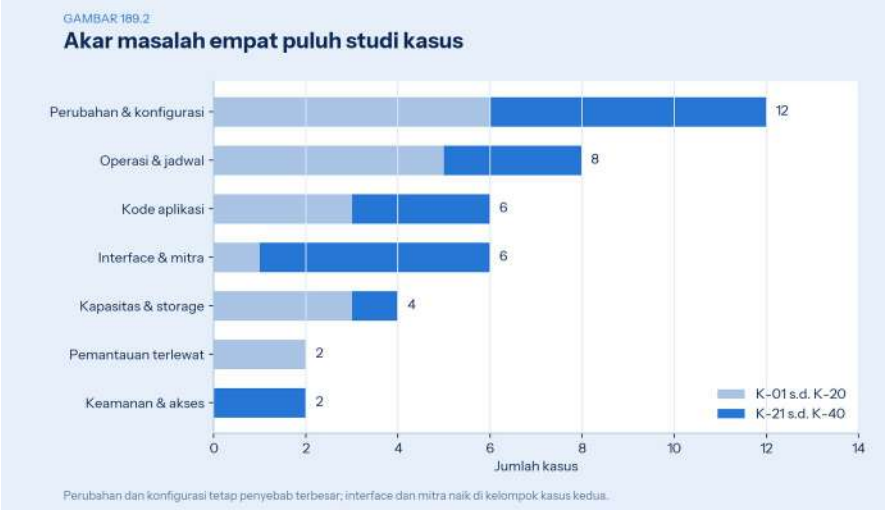
*Akar masalah.* Tidak ada batas ASUTIME untuk pengguna analitik, dan tidak ada resource group yang membatasi mereka (Bab 155.1 dan 169.1).

*Perbaikan.* Kueri dihentikan. Batas RLF dan resource group untuk pengguna analitik diterapkan.

*Pelajaran*. Satu kueri dari satu orang dapat memengaruhi tagihan dan kinerja seluruh bank. Perlindungan berlapis diperlukan sebelum masalah terjadi, bukan setelahnya.

## 189.5 Pola dari Empat Puluh Kasus

Dengan empat puluh kasus, polanya menjadi lebih jelas daripada dengan dua puluh.




*Gambar 189.2 — Akar masalah empat puluh studi kasus*

Perubahan dan konfigurasi tetap menjadi penyebab terbesar, dan sebagian besar kasus di kelompok ini berkaitan dengan situs DR atau langkah pemeliharaan yang tidak lengkap. Kategori interface dan mitra naik tajam di kelompok kedua: sertifikat, encoding, file terpotong, urutan data, dan perilaku koneksi. Semua itu berada di perbatasan antara bank dan pihak lain, tempat tanggung jawab paling mudah jatuh di celah.

| Pelajaran dari 40 kasus | Kasus pendukung |
|---|---|
| Setiap perubahan di situs utama harus diterapkan dan dibuktikan juga di situs DR | K-19, K-37, K-38 |
| Redundansi harus dibuktikan kembali setelah pemeliharaan | K-15, K-35, K-36 |
| Interface dengan mitra membutuhkan | K-06, K-25, K-26, K-30, K-33 |

| Pelajaran dari 40 kasus | Kasus pendukung |
|---|---|
| kontrak eksplisit: format, encoding, urutan, koneksi, dan sertifikat | |
| Restart dan rerun harus diuji di titik kegagalan yang paling sulit | K-18, K-28 |
| Otomasi dan perlindungan harus sempit dan eksplisit | K-39, K-40 |

Setiap baris tabel ini bisa diubah menjadi satu kontrol di lingkungan Anda: satu pemeriksaan otomatis, satu pertanyaan wajib di tiket perubahan, atau satu uji tambahan. Menerapkan kelima kontrol itu akan mencegah sebagian besar dari empat puluh kasus di bagian ini.

# 📑 Bagian XLV — Referensi Perintah Ringkas

Halaman-halaman ini dirancang untuk dicetak dan ditempel di meja jaga. Detail setiap perintah ada di bab yang dirujuk.

## Bab 190. Lembar Contekan per Area

| Area | Perintah | Fungsi | Bab |
|---|---|---|---|
| Sistem | `D IPLINFO` | Konfigurasi IPL aktif | 48 |
| Sistem | `D A,L` | Address space aktif | 48 |
| Sistem | `D R,L` | WTOR yang menunggu | 48 |
| Sistem | `D GRS,C` | Contention ENQ | 48 |
| Sistem | `D ASM` | Page dataset | 48 |
| Sysplex | `D XCF,SYSPLEX,ALL` | Anggota sysplex | 43 |
| Sysplex | `D XCF,STR,STRNAME=nama` | Status structure | 43, 179 |
| Sysplex | `D ETR` | Sinkronisasi waktu | 43 |
| I/O | `D M=DEV(xxxx)` | Jalur ke device | 6 |
| Storage | `D SMS,SG(nama),LISTVOL` | Pemakaian storage group | 18 |
| JES2 | `$D SPOOL` | Pemakaian spool | 21 |
| JES2 | `$D I` | Initiator | 21 |
| JES2 | `$DA` | Job aktif | 21 |
| Jaringan | `D TCPIP,,NETSTAT,CONN` | Koneksi aktif | 38 |

| Area | Perintah | Fungsi | Bab |
|---|---|---|---|
| Jaringan | `D TCPIP,,NETSTAT,TTLS` | Koneksi AT-TLS | 148 |
| WLM | `D WLM` | Policy aktif | 40 |
| SMF | `D SMF` | Status pencatatan SMF | 41 |
| Db2 | `-DIS THREAD(*)` | Thread aktif | 33 |
| Db2 | `-DIS DB(x) SPACENAM(*) RESTRICT` | Objek bermasalah | 33 |
| Db2 | `-DIS LOG` | Status log | 33 |
| CICS | `F region,CEMT I TASK` | Task aktif | 32 |
| CICS | `F region,CEMT I DSAS` | Storage CICS | 65 |
| MQ | `+CSQ1 DISPLAY USAGE PSID(*)` | Page set | 164 |
| MQ | `+CSQ1 DISPLAY CHSTATUS(*)` | Status channel | 164 |
| IMS | `/DISPLAY DB nama` | Status database IMS | 167 |
| RACF | `LU userid/LD DA('dsn') ALL` | Profil user dan dataset | 28 |
| HSM | `HSEND QUERY ACTIVE` | Aktivitas HSM | 45 |
| z/OS UNIX | `D OMVS,F` | Filesystem ter-mount | 25 |
| z/VM | `INDICATE LOAD` | Beban sistem z/VM | 160 |

## 190.1 Analisis: Menjaga Lembar Contekan Tetap Benar

Lembar contekan perintah paling berguna saat dicetak dan ditempel di meja jaga. Namun lembar yang dicetak juga cepat usang: nama sistem berubah, produk di-upgrade, dan perintah yang dulu benar kini memberi

hasil berbeda. Lembar contekan yang salah lebih berbahaya daripada tidak ada lembar sama sekali.

| Praktik | Penerapan |
|---|---|
| Sumber tunggal | Lembar dicetak dari satu file di Git, bukan disalin dan diedit per meja |
| Tanggal di setiap cetakan | Tanggal dan versi tercetak di sudut lembar |
| Tinjauan setelah upgrade | Setiap upgrade z/OS atau subsistem memicu pemeriksaan ulang semua perintah |
| Kelas risiko ditandai | Perintah baca saja, operasional, perubahan, dan destruktif dibedakan warna atau simbol (Bab 48.6) |
| Uji berkala | Setiap perintah di lembar dijalankan di sistem uji setiap kuartal |

Penandaan kelas risiko sangat membantu dalam tekanan. Pada pukul tiga pagi, petugas yang melihat perintah `FORCE` dengan tanda merah akan berhenti sejenak dan memeriksa runbook, dibandingkan bila perintah itu tercetak sama seperti `D A,L`.

Pertimbangkan juga versi digital yang dapat dicari, misalnya halaman di wiki internal atau artifact yang dapat diakses dari ponsel petugas jaga. Versi digital selalu mutakhir karena dibaca langsung dari sumbernya. Versi cetak tetap berguna saat akses ke sistem lain terganggu, yang justru sering terjadi saat insiden besar. Karena itu keduanya dipertahankan, dengan sumber yang sama.

## Bab 191. Triage: Gejala ke Perintah Pertama

| Gejala | Perintah pertama | Lanjutkan ke |
|---|---|---|
| Semua transaksi lambat | RMF Monitor III (delay) dan `D WLM` | Bab 62, 169 |
| Job tidak jalan | `$D J,$D I` | RB-026 |
| Job menunggu dataset | `D GRS,RES=(SYSDSN,nama)` | RB-003 |

| Gejala | Perintah pertama | Lanjutkan ke |
|---|---|---|
| Sistem terasa hang | `D GRS,ANALYZE,BLOCKER`, `D R,L` | RB-029 |
| Abend `x37` | `D SMS,SG(...)` | RB-001 |
| Abend `S0C7` | CEEDUMP atau `STATUS FAILDATA` | Bab 77 |
| CICS lambat | `CEMT I TASK`, `CEMT I DSAS` | Bab 67.1 |
| Db2 `-904` | `-DIS DB ... RESTRICT` | RB-019 |
| Mitra tidak bisa konek | `D TCPIP,,NETSTAT,CONN`, log PAGENT | RB-055 |
| Antrean MQ menumpuk | `DISPLAY QSTATUS ... IPPROCS` | RB-038 |
| GL tidak seimbang | Tabel kontrol EOD, balancing per cabang | RB-005, Bab 100 |
| Aktivitas user mencurigakan | SMF 80 dan 30 untuk user itu | RB-049 |

## 191.1 Analisis: Sepuluh Menit Pertama

Sepuluh menit pertama sebuah insiden menentukan banyak hal: apakah masalah dipahami dengan benar, apakah tindakan pertama memperbaiki atau memperburuk keadaan, dan apakah orang yang tepat dilibatkan tepat waktu. Urutan yang konsisten mencegah kepanikan dan tindakan yang tergesa-gesa.

| Menit | Langkah | Pertanyaan yang dijawab |
|---|---|---|
| 0–2 | Konfirmasi gejala dari dua sumber, misalnya alert dan transaksi uji | Apakah ini nyata? |
| 2–4 | Cakupan: semua layanan atau sebagian, satu sistem atau semua | Seberapa luas? |
| 4–6 | Perubahan terakhir: tiket, deploy, IPL, perubahan | Apa yang berubah? |

| Menit | Langkah | Pertanyaan yang dijawab |
|---|---|---|
| | konfigurasi dalam 24 jam | |
| 6–8 | Perintah pertama sesuai gejala (tabel Bab 191) | Di mana letak masalahnya? |
| 8–10 | Putuskan: tangani sendiri, eskalasi, atau nyatakan insiden besar | Siapa yang harus terlibat? |

Langkah “apa yang berubah” sering menjadi jalan pintas tercepat. Sebagian besar insiden dalam studi kasus berakar pada perubahan (Bab 189). Jika ada perubahan dalam 24 jam terakhir yang berkaitan dengan gejala, membatalkan perubahan itu sering lebih cepat daripada mendiagnosisnya.

**Hal yang tidak dilakukan dalam sepuluh menit pertama:**

- Me-restart komponen sebelum bukti diamankan, misalnya dump dan log, kecuali layanan harus segera dipulihkan.
- Mengubah beberapa hal sekaligus. Jika masalah hilang, tidak ada yang tahu perubahan mana yang memperbaikinya.
- Bekerja sendiri terlalu lama. Eskalasi lebih awal lebih murah daripada eskalasi yang terlambat.

Cetak tabel ini dan tempel di dekat console jaga bersama lembar contekan di Bab 190. Saat tekanan tinggi, urutan yang tertulis lebih bisa diandalkan daripada ingatan.

# 🌅 Bagian XLVI — Penutup

## Bab 192. Peta Jalan Pengembangan Diri

Lampiran D memetakan kompetensi per tahap. Bab ini melengkapinya dengan sumber belajar yang bisa dipakai di setiap tahap.

| Tahap | Sumber belajar | Praktik yang disarankan |
|---|---|---|
| Pemula | IBM Z Xplore, *ABCs of IBM z/OS System Programming* volume awal, Praktikum 1–2 | Jalankan setiap JCL di buku ini sendiri |
| Menengah | Dokumentasi resmi z/OS, Redbooks produk (CICS, Db2, MQ), Praktikum 3–5 | Ikut jaga bersama senior; tulis runbook pertama |
| Lanjut | Redbooks hardware seperti *IBM z17 Technical Guide*, komunitas pengguna mainframe | Pimpin uji DR dan proyek upgrade |
| Ahli | Konferensi dan komunitas internasional, proyek open source Zowe | Menjadi mentor; menulis dan berbagi pengetahuan |

Keahlian mainframe tumbuh dari insiden yang ditangani dengan tenang, dicatat dengan jujur, dan dibagikan kepada tim. Tidak ada jalan pintas, tetapi jalannya jelas.

### 192.1 Analisis: Rencana Belajar Dua Belas Bulan

Peta jalan pengembangan diri lebih mudah dijalankan bila dipecah menjadi rencana bulanan dengan hasil yang jelas. Rencana berikut ditujukan untuk engineer yang sudah memahami dasar z/OS dan ingin menjadi mandiri di sebagian besar area dalam satu tahun, dengan sekitar empat jam belajar per pekan di luar pekerjaan rutin.

| Bulan | Fokus | Hasil yang bisa dibuktikan |
|---|---|---|
| 1–2 | Batch, JES2, dataset, dan DFSORT | Praktikum 1 sampai 3 selesai; satu runbook batch ditulis |

| Bulan | Fokus | Hasil yang bisa dibuktikan |
|---|---|---|
| | | ulang dengan templat (Bab 52.1) |
| 3–4 | RACF dan keamanan | Praktikum 5 dan latihan lanjutannya; ikut satu tinjauan akses bulanan |
| 5–6 | Kinerja: RMF, WLM, dan SMF | Satu laporan kinerja mingguan disusun sendiri; satu analisis top-down seperti Bab 62.1 |
| 7–8 | Db2 dan CICS dari sisi sistem | Membaca statistik CICS dan akuntansi Db2; satu temuan SQL mahal dilaporkan |
| 9–10 | Sysplex, CF, dan DR | Ikut satu uji DR atau switchover; memimpin satu latihan tabletop |
| 11–12 | Otomasi dan modernisasi | Satu skrip REXX dengan pola standar (Bab 84.1) dipakai tim; satu langkah pipeline dengan Zowe CLI |

Setiap hasil di kolom ketiga adalah bukti nyata, bukan sertifikat kursus. Bukti seperti ini lebih berguna untuk penilaian kinerja, dan juga untuk menaikkan faktor bus tim (Bab 3.4), karena setiap hasil berarti satu area lagi yang bisa ditangani lebih dari satu orang.

Pasangkan rencana ini dengan mentor dari tim yang sama. Pertemuan singkat dua pekan sekali untuk membahas apa yang dipelajari dan apa yang membingungkan jauh lebih efektif daripada belajar sendiri dari dokumentasi. Mentor juga mendapat manfaat: menjelaskan sesuatu kepada orang lain adalah cara terbaik menemukan celah dalam pemahaman sendiri.

## 192.2 Jalur Membaca Menurut Peran

Buku ini tidak dirancang untuk dibaca dari halaman pertama sampai terakhir. Pembaca dengan peran berbeda akan mendapat manfaat terbesar dari jalur yang berbeda.

| Peran | Mulai dari | Lanjutkan ke | Simpan sebagai rujukan |
|---|---|---|---|
| Operator dan petugas jaga | Bagian XII (operasi harian), Bab 190–191 | Bagian XIII (troubleshooting), Bab 47.4, 90.1, 191.1 | Katalog runbook (Bab 88–95), lembar contekan |
| System engineer baru | Bagian I sampai V | Praktikum (Bagian XXXI), Bab 192.1 | Glosarium, Bab 190 |
| System engineer berpengalaman | Subbab analisis di bab-bab yang relevan dengan pekerjaan saat ini | Bagian XLII–XLIV, studi kasus | Rumus dan tabel keputusan di setiap bab |
| Arsitek dan perencana kapasitas | Bab 11.1, 42.2, 101.1, 129.1, 172.1 | Bagian XXX (proyek), Bab 177.1 | Bab 59.1, 170.1 |
| Tim keamanan | Bagian VII, Bab 29.1, 30.1, 31.1 | Bagian XXVI, Bab 182.1, 184.1 | Bab 112.1, 92.1 |
| Manajer TI | Bab 1.3, 3.4, 46.1, 177.1 | Bab 55.1, 173.1, 175.1 | Bab 126.1, 130.1 |

Subbab analisis yang ditambahkan di hampir setiap bab dapat dibaca terpisah. Setiap subbab dimulai dari masalah nyata, memberi cara mengukurnya, lalu menawarkan tindakan. Pola ini juga bisa dipakai untuk menulis analisis sendiri tentang lingkungan kerja Anda: ukur, bandingkan dengan batas, putuskan, lalu ukur lagi.

Angka-angka dalam contoh di buku ini sebagian besar adalah ilustrasi. Nilainya bukan pada angkanya, tetapi pada cara berpikirnya. Ganti setiap angka dengan data dari sistem Anda sendiri, dan buku ini akan menjadi alat kerja, bukan sekadar bacaan.

## Bab 193. Kata Penutup

Buku ini dimulai dengan pertanyaan sederhana: mengapa bank masih mempercayakan transaksinya pada platform yang lahir tahun 1964? Jawabannya ada di setiap bab: integritas yang dibangun ke dalam, ketersediaan yang dirancang sejak awal, dan ekosistem alat yang matang untuk setiap masalah yang pernah terjadi.

Namun platform hanya separuh cerita. Separuh lainnya adalah orang-orang yang menjaganya pada pukul tiga pagi, membaca dump dengan sabar, dan menulis runbook agar rekan berikutnya tidak perlu mengulang malam yang sama.

Kepada para system engineer mainframe di Indonesia, senior yang akan pensiun dan junior yang baru mulai, buku ini ditulis untuk Anda. Semoga bermanfaat, dan semoga setiap EOD selesai sebelum batas window.

**Romi Nur Ismanto**

rominur.com · hello@rominur.com

September 2026

# Lampiran

## Lampiran A. Peta Konsep IBM i ke IBM Z

Tabel ini membantu engineer IBM i yang pindah atau merangkap ke mainframe. Padanannya tidak selalu satu banding satu, tetapi cukup untuk orientasi awal.

| Konsep | IBM i | IBM Z (z/OS) |
|---|---|---|
| Hypervisor | PowerVM, VIOS | PR/SM, z/VM |
| Konsol hardware | HMC Power | HMC/SE IBM Z |
| Boot | IPL (A/B side, SLIC) | IPL dari SYSRES dengan LOADPARM |
| Konfigurasi sistem | System value | Member PARMLIB |
| Penyimpanan | Single-level storage, ASP | Dataset, volume 3390, DFSMS storage group |
| Wadah objek | Library | HLQ dataset dan katalog |
| Filesystem Unix | IFS | z/OS UNIX dengan zFS |
| Bahasa otomasi | CL | REXX, JCL |
| Batch | Job queue, subsystem | JES2, initiator, job class |
| Prioritas kerja | Subsystem, memory pool, run priority | WLM service class |
| Basis data | Db2 for i (terintegrasi) | Db2 for z/OS (subsistem terpisah) |
| Transaksi online | Program interaktif 5250 | CICS, IMS TM |
| Keamanan | User profile, object authority | RACF (atau ACF2/Top Secret) |
| Audit | Audit journal QAUDJRN | SMF tipe 80 |
| Journaling | Journal dan receiver | Log Db2, logstream, log CICS |
| Pemeliharaan | PTF, `LODPTF/APYPTF` | SMP/E `RECEIVE/APPLY/ACCEPT` |
| Backup | SAVxxx, BRMS | DFSMSdss, DFSMShsm, FlashCopy |

| Konsep | IBM i | IBM Z (z/OS) |
|---|---|---|
| HA/DR | PowerHA, replikasi logis | Parallel Sysplex, GDPS, HyperSwap |
| Pemantauan kinerja | Collection Services, PDI | RMF, SMF |
| Antarmuka web | Navigator for i | z/OSMF |

# Lampiran B. Glosarium (A–Z)

| Istilah | Arti | Bab |
|---|---|---|
| 4HRA | Rata-rata bergulir empat jam pemakaian MSU; dasar tagihan sub-capacity | 42 |
| Abend | Penghentian abnormal program, misalnya `S0C7` | 49 |
| AOR / TOR | Application-Owning Region / Terminal-Owning Region di CICS | 63 |
| APF | Authorized Program Facility; library yang programnya boleh berjalan authorized | 29 |
| APU-PPT | Anti Pencucian Uang dan Pencegahan Pendanaan Terorisme | 96 |
| ARM | Automatic Restart Manager, restart otomatis subsistem | 180 |
| ARO | Automatic Roll Over pada deposito | 98 |
| AT-TLS | Application Transparent TLS, TLS berbasis kebijakan di stack TCP/IP | 148 |
| BI-FAST | Infrastruktur transfer ritel real-time Bank Indonesia | 101 |
| BMP | Batch Message Processing region di IMS | 167 |
| BSDS | Bootstrap Data Set, | 68 |

| Istilah | Arti | Bab |
|---|---|---|
| | inventaris log Db2 dan MQ | |
| CBU | Capacity Backup, kapasitas cadangan untuk DR | 4 |
| CCSID | Coded Character Set Identifier, penanda pengodean data | 144 |
| CEEDUMP | Laporan dump dari Language Environment | 76 |
| CF | Coupling Facility, LPAR penyimpan structure bersama sysplex | 43 |
| CFRM | Coupling Facility Resource Management policy | 179 |
| CHPID | Channel Path ID, nomor logis jalur I/O | 6 |
| CICS | Monitor transaksi online utama di z/OS | 32 |
| CIF | Customer Information File | 96 |
| CKPN | Cadangan Kerugian Penurunan Nilai | 99 |
| COMP-3 | Packed decimal di COBOL | 74 |
| CPC | Central Processor Complex, satu mesin fisik IBM Z | 4 |
| CSD | Dataset definisi resource CICS | 64 |
| Cyber Vault | Lingkungan pemulihan terisolasi berbasis snapshot tak dapat diubah | 113 |
| DASD | Direct Access Storage Device, istilah untuk disk | 7 |
| DBRC | Database Recovery Control di IMS | 167 |
| DDF | Distributed Data Facility, akses jarak jauh ke Db2 | 33 |

| Istilah | Arti | Bab |
|---|---|---|
| DFSMS | Subsistem manajemen dataset dan storage | 18 |
| DLQ | Dead-letter queue MQ untuk pesan yang tidak terkirim | 165 |
| DSA | Dynamic Storage Area di CICS | 65 |
| DVIPA | Dynamic Virtual IP Address | 37 |
| EBCDIC | Pengodean karakter mainframe | 144 |
| EOD / EOM / EOY | End of Day / Month / Year | 104 |
| GBP | Group Buffer Pool di Db2 data sharing | 71 |
| GDG | Generation Data Group, dataset berversi | 16 |
| GDPS | Solusi otomasi ketersediaan dan DR IBM | 44 |
| GRS | Global Resource Serialization, pengelola ENQ | 180 |
| HCD / IODF | Alat dan file definisi konfigurasi I/O | 13 |
| HiperSockets | Jaringan memori antar-LPAR dalam satu CPC | 36 |
| HLQ | High-Level Qualifier, qualifier pertama nama dataset | 16 |
| HMC / SE | Hardware Management Console / Support Element | 5 |
| HyperPAV | Alias device untuk I/O paralel ke satu volume | 6 |
| HyperSwap | Pengalihan I/O ke disk sekunder tanpa menghentikan aplikasi | 44 |
| Hypercare | Periode pemantauan intensif setelah cutover | 130 |
| ICF | Integrated Coupling Facility, | 4 |

| Istilah | Arti | Bab |
|---|---|---|
| | PU untuk menjalankan CF | |
| ICSF | Layanan kriptografi z/OS | 31 |
| Idempoten | Sifat proses yang aman dijalankan lebih dari sekali | 105 |
| IFL | Integrated Facility for Linux | 4 |
| ILC | Instruction Length Code | 79 |
| IMS | Database hierarkis dan transaction manager IBM | 167 |
| IPCS | Alat analisis dump | 51 |
| IPL | Initial Program Load, proses boot z/OS | 12 |
| JCL / JES2 | Bahasa kendali job / subsistem entri job | 19, 21 |
| LBUT | Laporan Bank Umum Terintegrasi | 109 |
| LE | Language Environment, runtime bersama bahasa pemrograman | 76 |
| LGR | Live Guest Relocation di z/VM | 161 |
| LOADPARM | Parameter delapan karakter untuk IPL | 12 |
| LPA | Link Pack Area, area modul bersama | 12 |
| LPAR | Logical Partition | 5 |
| LPL / GRECP | Status objek Db2 yang perlu dipulihkan | 33 |
| MLC | Monthly License Charge | 177 |
| MQT | Materialized Query Table | 153 |
| MSU | Millions of Service Units per hour, satuan kapasitas lisensi | 4 |
| MTBF / MTTR | Rata-rata waktu antar- | 158 |

| Istilah | Arti | Bab |
|---|---|---|
| | kegagalan / rata-rata waktu pemulihan | |
| MXT | Batas task bersamaan di CICS | 65 |
| OSA | Open Systems Adapter, kartu jaringan IBM Z | 36 |
| PAGENT | Policy Agent untuk AT-TLS dan kebijakan jaringan | 148 |
| PARMLIB | Library parameter sistem z/OS | 12 |
| PBR / PBG | Partition-by-range / Partition-by-growth table space | 152 |
| PDS / PDSE | Partitioned dataset (extended) | 16 |
| PI | Performance Index WLM; di atas 1 berarti tujuan meleset | 168 |
| PR/SM | Hypervisor firmware yang membagi CPC menjadi LPAR | 5 |
| PSW | Program Status Word | 78 |
| QSG | Queue sharing group MQ | 164 |
| RACF | Pengelola keamanan bawaan z/OS | 27 |
| RAK | Rekening Antar Kantor | 100 |
| RLS | Record Level Sharing untuk VSAM | 66 |
| RMF | Resource Measurement Facility, pengukur kinerja | 41 |
| RSU | Recommended Service Upgrade | 14 |
| RTS | Real-Time Statistics Db2 | 156 |
| Safeguarded Copy | Snapshot DS8000 yang tidak dapat diubah oleh host | 113 |
| SAP | System Assist Processor, PU | 4 |

| Istilah | Arti | Bab |
|---|---|---|
| | pengelola I/O | |
| SCRT | Sub-Capacity Reporting Tool | 42 |
| SCV | Single Customer View untuk penjaminan LPS | 109 |
| SDSF | Antarmuka melihat job, output, dan log | 24 |
| SFM | Sysplex Failure Management | 180 |
| SLIK | Sistem Layanan Informasi Keuangan OJK | 109 |
| SMF | System Management Facilities, catatan kejadian sistem | 41 |
| SMP/E | Alat instalasi dan pemeliharaan software z/OS | 14 |
| SOS | Short on Storage di CICS | 65 |
| SSI | Single System Image, cluster z/VM | 161 |
| Stand-in | Mode layanan terbatas saat core menjalankan batch | 35 |
| STP | Server Time Protocol, sinkronisasi waktu antar-CPC | 43 |
| Suspense | Akun penampung sementara transaksi yang belum final | 103 |
| Sysplex | Kumpulan sistem z/OS yang bekerja sebagai satu kesatuan | 43 |
| SYSRES | Volume berisi target library sistem operasi | 12 |
| TCB / SRB | Unit kerja tugas biasa / rutinitas sistem berprioritas tinggi | 10 |
| TSO / ISPF | Lingkungan interaktif dan antarmuka menu z/OS | 23 |

| Istilah | Arti | Bab |
|---|---|---|
| UACC | Akses default RACF bagi semua pengguna | 27 |
| VIPA | Virtual IP Address | 37 |
| VSAM | Virtual Storage Access Method | 16 |
| WLM | Workload Manager | 40, 168 |
| WTOR | Write To Operator with Reply, pesan yang menunggu jawaban | 48 |
| XCF | Cross-system Coupling Facility, komunikasi antarsistem sysplex | 178 |
| z/OSMF | Antarmuka web dan REST untuk administrasi z/OS | 128 |
| z/VM | Hypervisor IBM Z untuk banyak guest | 160 |
| zFS | Filesystem z/OS UNIX | 25 |
| zIIP | Prosesor khusus untuk beban eligible, tidak menambah biaya MSU | 4 |
| Zowe | Kerangka open source untuk mengakses z/OS dari alat modern | 87 |

## Lampiran C. Latihan Soal

1. Jelaskan perbedaan IPL dengan CLPA dan tanpa CLPA, serta kapan masing-masing diperlukan.
2. Sebuah job EOD abend SB37 di step SORT. Sebutkan tiga kemungkinan penyebab dan tindakan untuk masing-masing.
3. Mengapa start `COLD` pada JES2 berbahaya di lingkungan produksi?
4. Rancang service class WLM untuk transaksi teller, API mobile banking, dan batch EOD. Tetapkan goal dan importance, lalu jelaskan alasannya.

5. Transaksi CICS mulai abend AEY9 setelah jam 02.00. Susun urutan diagnosis Anda.
6. Jelaskan bagaimana memindahkan beban ke zIIP dapat menurunkan biaya bulanan.
7. Rancang skenario uji DR yang mengukur RTO dan RPO secara jujur.

**Proyek latihan:** tulis skrip REXX yang setiap pagi memeriksa spool, storage group, WTOR yang tertunda, dan status CF, lalu mengirim ringkasannya lewat email atau ke sistem monitoring.

## Lampiran D. Peta Jalan Pengembangan Diri

| Tahap | Fokus belajar | Bukti kompetensi |
|---|---|---|
| 0–6 bulan | ISPF, JCL, SDSF, perintah console, dataset | Mampu menangani RB-001 sampai RB-003 sendiri |
| 6–18 bulan | SMP/E, PARMLIB, RACF dasar, REXX | Memasang RSU di sandbox dan memimpin IPL terencana |
| 18–36 bulan | WLM, RMF, sysplex, Db2/CICS dari sisi sistem | Memimpin analisis kinerja dan uji DR |
| Di atas 3 tahun | Arsitektur, GDPS, modernisasi, perencanaan biaya | Menyusun roadmap platform dan menjadi mentor |

## Lampiran E. Daftar Pustaka

Angka spesifikasi dan fitur dapat berubah per versi. Rujuk edisi terbaru dari setiap sumber. Sumber daring diakses September 2026.

### Buku dan IBM Redbooks


- Ismanto, R. N. (2026). *Handbook of System Engineering for IBM i (AS/400): Arsitektur, Operasi, dan Core Banking* (Edisi Pertama). rominur.com.
- Parziale, L., Fadel, L., & Jon, S. (2017). *ABCs of IBM z/OS System Programming, Volume 1* (5th ed., SG24-6981-04). IBM Redbooks. redbooks.ibm.com

- Palacio, E., Achouri, H., Bride, M., Ertl, M., Kuehner, L., Lascu, O., et al. (2025). *IBM z17 (9175) Technical Guide* (SG24-8579-00). IBM Redbooks. redbooks.ibm.com
- White, B., & Palacio, E. (2026). *IBM z17 Technical Introduction* (2nd ed., SG24-8580-01). IBM Redbooks. redbooks.ibm.com
- Palacio, E., Ashida, S., Bride, M., Ertl, M., Kuehner, L., Monteiro, P., et al. (t.t.). *IBM z17 ME2 and IBM z17 MER Technical Guide* (SG24-8608-00). IBM Redbooks. redbooks.ibm.com
- White, B., Palacio, E., Monteiro, P., Oughton, P., & Soellig, M. (t.t.). *IBM Z Connectivity Handbook* (SG24-5444-24). IBM Redbooks. redbooks.ibm.com
- White, B. (2025). *IBM Z Time Synchronization Implementation Guide* (3rd ed., SG24-8480-02). IBM Redbooks. redbooks.ibm.com


## Dokumentasi Produk IBM

Seluruh judul berikut tersedia di IBM Documentation untuk versi produk yang berlaku.


- IBM Corporation. *z/OS MVS System Commands*.
- IBM Corporation. *z/OS MVS Initialization and Tuning Reference*.
- IBM Corporation. *z/OS MVS JCL Reference* dan *z/OS JES2 Commands*.
- IBM Corporation. *z/OS MVS System Codes* dan *z/OS MVS System Messages*.
- IBM Corporation. *z/OS MVS System Management Facilities (SMF)*.
- IBM Corporation. *z/OS Security Server RACF Command Language Reference*.
- IBM Corporation. *SMP/E for z/OS User's Guide*.
- IBM Corporation. *z/OS Communications Server: IP Configuration Guide*.
- IBM Corporation. *z/OS DFSMS* (koleksi dokumentasi DFSMSdfp, DFSMSdss, DFSMShsm).
- IBM Corporation. *z/OS Language Environment Programming Guide*.

- IBM Corporation. Dokumentasi *CICS Transaction Server for z/OS*, *Db2 for z/OS*, *IBM MQ for z/OS*, *IMS*, dan *z/VM*.

### Regulasi dan Publikasi Indonesia

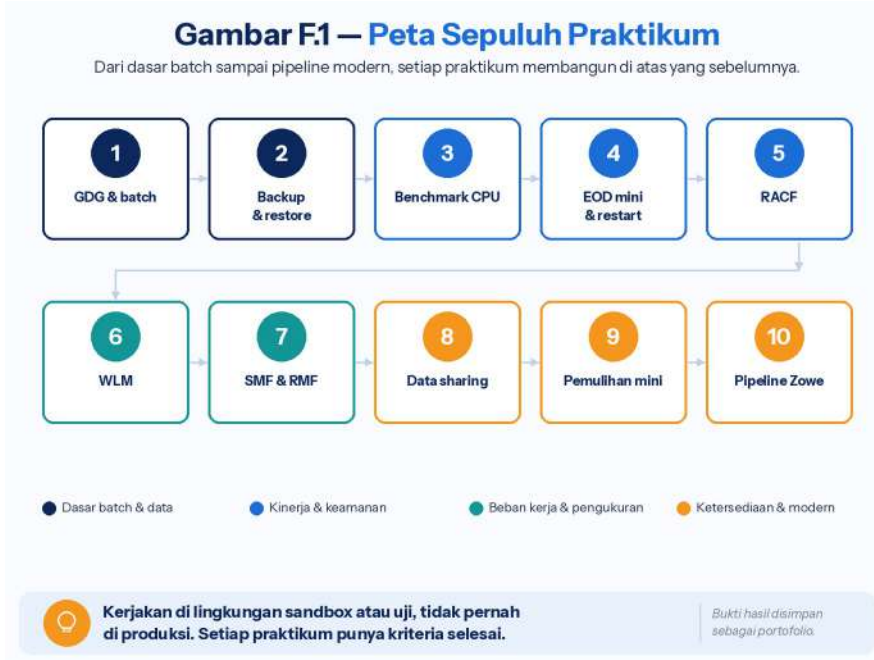
- Bank Indonesia. (2019). *Peraturan Bank Indonesia Nomor 21/9/PBI/2019 tentang Laporan Bank Umum Terintegrasi*. peraturan.go.id
- Bank Indonesia. (2021). *FAQ BI-FAST*. bi.go.id
- Bank Indonesia. (2021). *FAQ Skema Harga dan Batas Nominal Transaksi BI-FAST*. bi.go.id
- Otoritas Jasa Keuangan. (2019). *POJK Nomor 40/POJK.03/2019 tentang Penilaian Kualitas Aset Bank Umum*.
- Otoritas Jasa Keuangan. (2021). *BI-ANTASENA Siap Implementasi Juli 2021*. ojk.go.id
- Otoritas Jasa Keuangan. (2022). *POJK Nomor 11/POJK.03/2022 tentang Penyelenggaraan Teknologi Informasi oleh Bank Umum*.
- Republik Indonesia. (2022). *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*.

### Program Pembelajaran

- IBM. *IBM Z Xplore* — program belajar mandiri z/OS.
- IBM. *IBM Z Academic Initiative* — materi untuk perguruan tinggi.
- Open Mainframe Project. *Zowe* — proyek open source untuk akses z/OS dari alat modern.

## Lampiran F. Praktikum Lanjutan

Lampiran ini berisi lima praktikum lanjutan (Praktikum 6–10) yang melanjutkan Praktikum 1–5 di Bagian XXXI. Kelima praktikum ini membawa pembaca dari pekerjaan batch dan keamanan ke pengelolaan beban kerja, pengukuran, ketersediaan, dan otomasi modern.




*Gambar F.1 — Peta sepuluh praktikum*

**Aturan main untuk semua praktikum:**

- Kerjakan hanya di sandbox atau sistem uji yang memang disediakan untuk latihan, dengan izin pemilik sistem.
- Pakai HLQ latihan sendiri, misalnya `HLQ.LAB`, dan jangan menyentuh dataset lain.
- Simpan bukti setiap praktikum: job log, output perintah, dan jawaban pertanyaan refleksi. Kumpulan bukti ini adalah portofolio belajar (Bab 192.1).
- Bila ragu tentang dampak suatu langkah, tanyakan dulu kepada pembimbing. Di mainframe, “coba saja” bukan metode belajar yang aman.

## F.1 Praktikum 6: Mengamati WLM Bekerja

**Tujuan.** Melihat langsung bagaimana klasifikasi dan tujuan WLM memengaruhi kinerja sebuah job.

**Prasyarat.** Sandbox dengan hak mengubah policy WLM. Job pemakan CPU dari Praktikum 3 yang dapat dijalankan beberapa salinan sekaligus.

**Langkah.**

1. Simpan salinan policy WLM yang aktif sebelum mengubah apa pun, lewat aplikasi WLM di ISPF. Salinan ini adalah jalan kembali.
2. Buat service class `LABLOW` dengan tujuan velocity 10 dan importance 5, serta service class `LABHIGH` dengan tujuan velocity 60 dan importance 2.
3. Tambahkan aturan klasifikasi di subsistem JES: nama job `LABLO*` ke `LABLOW`, dan `LABHI*` ke `LABHIGH`.
4. Install definisi, lalu aktifkan policy dengan `V WLM,POLICY=nama`. Verifikasi dengan `D WLM`.
5. Jalankan tiga salinan job CPU bernama `LABLO1`–`LABLO3` dan satu salinan bernama `LABHI1` secara bersamaan, di saat sandbox cukup sibuk agar ada kontensi.
6. Amati di RMF Monitor III (laporan ringkasan sistem dan delay) selama beberapa interval.

**Yang diamati.**

| Pengamatan | Yang diharapkan |
|---|---|
| Velocity yang dicapai `LABHIGH` | Mendekati atau melebihi 60 |
| Delay CPU di `LABLOW` | Jauh lebih tinggi dari `LABHIGH` |
| Waktu selesai `LABHI1` dibanding `LABLO1` | `LABHI1` selesai lebih cepat walaupun kodenya sama |
| PI kedua service class | `LABHIGH` di bawah 1; `LABLOW` bisa di atas 1 tanpa banyak dibantu WLM karena importance rendah |

**Pertanyaan refleksi.** Apa yang terjadi bila tujuan `LABHIGH` dinaikkan menjadi velocity 95? Mengapa tujuan yang terlalu tinggi bisa merugikan pekerjaan lain (Bab 168.1)?

**Kriteria selesai.** Tabel pengamatan terisi dengan angka dari RMF, policy dikembalikan ke salinan awal, dan `D WLM` membuktikan policy asli aktif kembali.

## F.2 Praktikum 7: Menyusun Laporan Kinerja dari SMF

**Tujuan.** Membongkar record SMF dari logstream dan menyusun laporan kinerja harian dengan RMF Postprocessor.

**Prasyarat.** Sandbox dengan perekaman SMF ke logstream dan RMF aktif. Akses baca ke logstream SMF.

**Langkah 1: bongkar record 70–79 untuk satu hari.**

```
//SMFDL    EXEC PGM=IFASMFDL
//SYSPRINT DD SYSOUT=*
//OUTDD    DD DSN=HLQ.LAB.SMF7079,DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,
(50,50),RLSE),DCB=(RECFM=VBS,LRECL=32760)
//SYSIN    DD *
  LSNAME(IFASMF.DEFAULT,OPTIONS(DUMP))
  OUTDD(OUTDD,TYPE(70:79))
  DATE(2026265,2026265)
/*
```

Sesuaikan nama logstream dan tanggal (tahun dan hari ke-berapa) dengan sandbox Anda.

**Langkah 2: jalankan RMF Postprocessor.**

```
//RMFPP    EXEC PGM=ERBRMFPP
//MFPINPUT DD DISP=SHR,DSN=HLQ.LAB.SMF7079
//MFPMSGDS DD SYSOUT=*
//SYSIN    DD *
  REPORTS(CPU)
  SYSRPTS(WLMGL(SCPER))
  SUMMARY(INT)
/*
```

**Langkah 3.** Susun ringkasan satu halaman berisi pemakaian CPU per jam, tiga service class dengan PI tertinggi, dan satu kalimat kesimpulan untuk setiap temuan.

**Yang diamati.** Jam dengan pemakaian CPU tertinggi, service class yang pernah meleset dari tujuannya, dan apakah pola harian sesuai dengan yang diharapkan dari beban sandbox.

**Pertanyaan refleksi.** Bagaimana Anda memvalidasi bahwa angka di laporan ini benar (Bab 171.1)? Record SMF apa lagi yang dibutuhkan untuk menjelaskan mengapa sebuah service class meleset?

**Kriteria selesai.** Ringkasan satu halaman ditulis dengan bahasa yang bisa dipahami atasan non-teknis, lengkap dengan angka pendukung dari laporan RMF.

## F.3 Praktikum 8: Mengamati Data Sharing

**Tujuan.** Memahami apa yang terjadi di Coupling Facility saat dua anggota Db2 berbagi data, termasuk saat structure dibangun ulang.

**Prasyarat.** Grup data sharing Db2 uji dengan minimal dua anggota di sistem berbeda. Hak menjalankan perintah Db2 dan `SETXCF` di lingkungan uji.

**Langkah.**

1. Tampilkan kondisi grup dengan `-DIS GROUP DETAIL` dari salah satu anggota. Catat anggota, statusnya, dan nama structure grup.
2. Tampilkan lock structure dan group buffer pool dengan `D XCF,STR,STRNAME=nama` untuk masing-masing. Catat CF tempatnya dan status duplex.
3. Tampilkan statistik group buffer pool dengan `-DIS GBPOOL(GBP1) GDETAIL`.
4. Jalankan dua job batch dari dua anggota berbeda yang memperbarui tabel yang sama, dengan frekuensi commit yang sama.
5. Ulangi langkah 3 dan bandingkan angka baca, tulis, dan castout.
6. Dengan persetujuan pengelola lingkungan uji, jalankan rebuild terencana untuk structure group buffer pool dengan `SETXCF START,REBUILD,STRNAME=nama`. Amati pesan di console dan durasi rebuild.

**Yang diamati.**

| Pengamatan | Pertanyaan |
|---|---|
| Statistik GBP sebelum dan sesudah job | Seberapa besar lalu lintas ke CF yang dihasilkan dua job itu? |
| Pesan saat rebuild | Berapa lama rebuild, dan apakah job yang berjalan terganggu? |
| Lokasi structure setelah rebuild | Apakah structure kembali ke CF sesuai `PREFLIST`? |

**Pertanyaan refleksi.** Mengapa kegagalan CF tanpa duplexing lebih berbahaya bagi lock structure daripada bagi group buffer pool (Bab 179.1)? Apa yang akan Anda periksa bila rebuild membutuhkan waktu jauh lebih lama dari biasanya?

**Kriteria selesai.** Tabel pengamatan terisi, lokasi structure dikembalikan sesuai `PREFLIST` bila berpindah, dan kondisi grup sama seperti sebelum praktikum.

## F.4 Praktikum 9: Latihan Pemulihan Mini

**Tujuan.** Mengukur RTO dan RPO nyata untuk pemulihan satu aplikasi kecil, dari keputusan pemulihan sampai data terbukti benar.

**Prasyarat.** Dataset EOD mini dari Praktikum 4, backup dari Praktikum 2, dan stopwatch.

**Langkah.**

1. Catat waktu T0 sebagai “saat kerusakan ditemukan”. Anggap dataset master EOD mini rusak.
2. Tentukan backup terakhir yang bersih dan catat waktu pembuatannya. Selisih T0 dengan waktu backup adalah RPO yang akan dicapai.
3. Pulihkan dataset dari backup ke nama lain dengan `RENAMEU`, tanpa menimpa data asli (Bab 134.1).
4. Jalankan EOD mini terhadap salinan yang dipulihkan untuk mengejar transaksi sejak backup.

5. Bandingkan hasilnya dengan run acuan menggunakan ICETOOL: jumlah record dan total nominal.
6. Catat waktu T1 saat total kontrol cocok. Selisih T1 dengan T0 adalah RTO yang dicapai.

**Yang diamati.**

| Ukuran | Hasil | Target contoh |
|---|---|---|
| RPO (T0 dikurangi waktu backup) | ... menit | Kurang dari 24 jam |
| RTO (T1 dikurangi T0) | ... menit | Kurang dari 60 menit |
| Langkah terlama | ... | |
| Langkah yang tidak ada di runbook | ... | |

**Pertanyaan refleksi.** Langkah mana yang paling lama, dan apakah bisa dipercepat dengan otomasi (Bab 116.1)? Bila data produksi seratus kali lebih besar, apakah RTO-nya masih memenuhi janji layanan?

**Kriteria selesai.** Tabel terisi dengan angka nyata, dan satu perbaikan runbook ditulis berdasarkan langkah yang terbukti hilang atau lambat.

## F.5 Praktikum 10: Pipeline Build dengan Zowe CLI

**Tujuan.** Menjalankan compile, uji, dan pemeriksaan hasil program COBOL dari laptop atau server pipeline, tanpa membuka ISPF.

**Prasyarat.** Zowe CLI terpasang dan terhubung ke sandbox. Program COBOL sederhana dan JCL compile yang sudah berjalan dari ISPF.

**Langkah.**

1. Buat konfigurasi koneksi dengan `zowe config init`, lalu uji koneksi dengan menampilkan daftar dataset di HLQ latihan.
2. Unggah source: `zowe zos-files upload file-to-data-set hitung.cbl "HLQ.LAB.COBOL(HITUNG)"`.

3. Jalankan compile dan tunggu hasilnya dalam format JSON: `zowe zos-jobs submit data-set "HLQ.LAB.JCL(COMPILE)" --wait-for-output --rfj > hasil.json`.
4. Periksa return code dari JSON, dan hentikan skrip bila bukan `CC 0000`.
5. Jalankan job uji dengan cara yang sama, lalu unduh seluruh output job ke folder lokal dengan `zowe zos-jobs download output <id-job>`.
6. Satukan langkah 2–5 dalam satu skrip shell yang berhenti di langkah pertama yang gagal.

**Yang diamati.** Waktu dari perubahan kode sampai hasil uji tersedia, dan apakah skrip benar-benar berhenti saat compile gagal. Uji dengan sengaja memasukkan kesalahan sintaks ke program.

**Pertanyaan refleksi.** Di mana kredensial Zowe disimpan, dan bagaimana mengamankannya di server pipeline (Bab 87.1)? Langkah apa yang harus ditambahkan sebelum hasil build boleh dipromosikan ke produksi?

**Kriteria selesai.** Skrip berjalan dari awal sampai akhir untuk kode yang benar, berhenti dengan jelas untuk kode yang salah, dan output job tersimpan sebagai artefak di folder lokal.

# Lampiran G. Analisis Lanjutan

Lampiran ini berisi analisis tingkat kedua untuk topik yang paling sering dihadapi system engineer core banking: CICS, Db2, EOD, keamanan, dan DR. Setiap analisis melengkapi subbab analisis pertama di bab terkait, dengan satu langkah lebih dalam ke pengukuran dan keputusan.

## G.1 CICS: Menguraikan Waktu Respons Transaksi

Waktu respons transaksi CICS adalah jumlah dari beberapa komponen, dan data monitoring CICS (SMF 110 kelas performance) mencatat masing-masing komponen secara terpisah untuk setiap transaksi.

$$T_{\text{respons}} = T_{\text{tunggu dispatch}} + T_{\text{CPU}} + T_{\text{tunggu Db2}} + T_{\text{tunggu file}} + T_{\text{lain}}$$

| Komponen | Contoh transaksi transfer, total 120 ms | Bila komponen ini dominan |
|---|---|---|
| Tunggu dispatch | 8 ms | QR TCB jenuh atau MXT hampir tercapai (Bab 32.3) |
| CPU di CICS | 12 ms | Logika program atau jumlah perpindahan TCB (Bab 65.4) |
| Tunggu Db2 | 70 ms | SQL mahal, lock, atau I/O Db2 (Bab 33.4) |
| Tunggu file VSAM | 20 ms | I/O volume atau kontensi record (Bab 6.3) |
| Lainnya | 10 ms | MQ, koneksi ke region lain, atau sistem luar |

Dalam contoh ini, lebih dari separuh waktu dihabiskan menunggu Db2. Menambah AOR atau menaikkan MXT tidak akan menolong. Yang perlu diperiksa adalah SQL dan lock di transaksi itu.

Analisis seperti ini paling berguna bila dilakukan per jenis transaksi dan dibandingkan dengan baseline minggu sebelumnya. Komponen yang naik tajam menunjukkan di mana perubahan terjadi, bahkan sebelum penyebabnya diketahui. Susun laporan mingguan berisi sepuluh transaksi terpenting dengan komponen waktunya, agar pergeseran seperti ini langsung terlihat.

## G.2 Db2: Berapa Lama Log Aktif Bertahan

Db2 menulis perubahan ke log aktif, lalu memindahkan (offload) log aktif yang penuh ke archive log. Jika log aktif terisi lebih cepat daripada offload, dan semua log aktif penuh, Db2 tidak dapat melanjutkan pekerjaan yang mengubah data. Karena itu kapasitas log aktif harus dihitung terhadap laju tulis log di jam paling sibuk.

$$T_{\text{isi}} = \frac{N_{\text{log aktif}} \times V_{\text{per log}}}{\text{laju tulis log}} \qquad \text{syarat: laju offload} > \text{laju tulis log}$$

Contoh: enam log aktif masing-masing 4 GB (total 24 GB), dan laju tulis log di puncak EOD sekitar 20 MB per detik. Seluruh log aktif terisi dalam

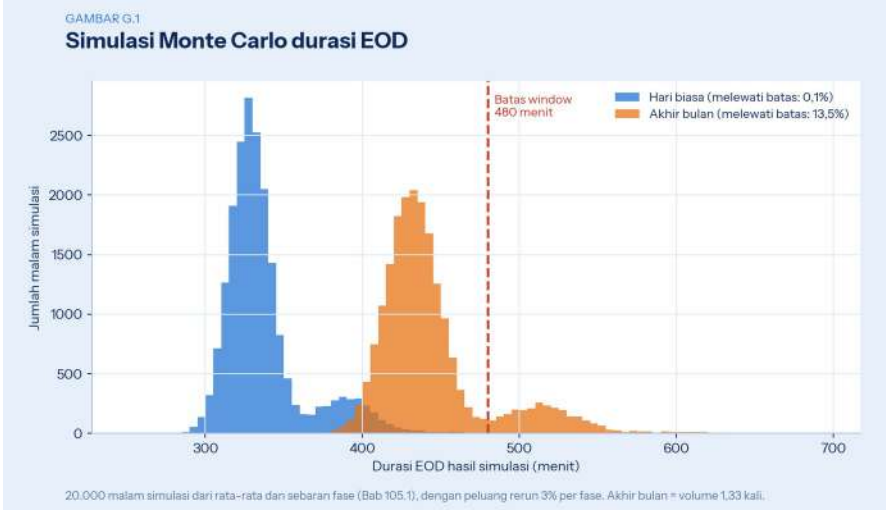
sekitar 20 menit. Selama offload berjalan lebih cepat daripada 20 MB per detik, tidak ada masalah. Tetapi jika offload tertunda, misalnya karena perangkat archive lambat atau ruang archive habis, Db2 hanya punya waktu sekitar 20 menit sebelum berhenti menerima perubahan.

| Keputusan | Rekomendasi |
|---|---|
| Kapasitas total log aktif | Cukup untuk beberapa jam laju tulis puncak, agar ada waktu menangani masalah offload |
| Dual logging | Aktif untuk log aktif dan archive; satu salinan rusak tidak menghentikan Db2 |
| Pemantauan | Alert saat jumlah log aktif yang belum di-offload melewati ambang |
| Ruang archive | Diproyeksikan seperti storage group (Bab 59.1), dengan cadangan untuk akhir bulan |

Ukur laju tulis log dari statistik Db2 untuk hari biasa dan akhir bulan secara terpisah. Program batch dengan commit yang sangat jarang, atau yang memperbarui banyak kolom yang tidak perlu, dapat menaikkan laju tulis log secara besar (Bab 70.3).

## G.3 EOD: Peluang Selesai Tepat Waktu dengan Simulasi

Analisis utilisasi window (Bab 104.1) memakai rata-rata. Simulasi Monte Carlo melangkah lebih jauh: dengan mengetahui rata-rata dan sebaran durasi setiap fase, peluang EOD melewati batas window dapat diperkirakan.




*Gambar G.1 — Simulasi Monte Carlo durasi EOD*

```
import numpy as np
rng = np.random.default_rng(42)
N = 20000                                          # jumlah
malam simulasi
rata = np.array([65, 70, 8, 60, 50, 60])           # menit
per fase, dari tabel kontrol
sd   = np.array([ 4,  3, 1.5, 9, 2.5, 6])
d = rng.normal(rata, sd, (N, 6)).clip(0)
rerun = (rng.random((N, 6)) < 0.03) * d            #
peluang 3% sebuah fase diulang
total = d.sum(axis=1) + rerun.sum(axis=1) + 15     # 15
menit langkah tetap
print('peluang melewati 480 menit:', (total > 480).mean())
```

Dengan data ilustrasi dari Bab 105.2, EOD hari biasa rata-rata sekitar 337 menit, dan peluangnya melewati batas 480 menit hanya sekitar 0,1%. Untuk akhir bulan dengan volume sekitar 1,33 kali, rata-ratanya naik menjadi sekitar 444 menit, dan peluang melewati batas melonjak menjadi sekitar 13,5%. Artinya, kira-kira satu dari tujuh akhir bulan akan terlambat bila tidak ada perubahan.

Angka seperti ini lebih meyakinkan bagi manajemen daripada “window akhir bulan sudah ketat”. Simulasi juga bisa menguji pilihan perbaikan: berapa peluangnya bila laporan dipindahkan setelah BOD, bila fase aging

dipercepat 20%, atau bila window diperpanjang 30 menit. Pilihan dengan penurunan peluang terbesar per satuan usaha adalah prioritas pertama.

## G.4 EOD: Keputusan Saat EOD Terlambat

Saat EOD berjalan lebih lambat dari biasanya dan batas window mendekat, keputusan harus diambil sebelum terlambat. Keputusan yang diambil pukul 05.45 untuk layanan yang harus buka pukul 06.00 biasanya bukan keputusan, melainkan kepanikan.

| Fase | Boleh ditunda setelah BOD? | Alasan |
|---|---|---|
| Posting transaksi hari ini | Tidak | Saldo esok hari bergantung padanya |
| Akrual bunga | Tidak, kecuali dengan prosedur khusus yang disetujui | Bunga salah dilihat nasabah |
| Aging kredit | Kadang, bila tidak mengubah layanan pagi | Bergantung pada kebijakan bank |
| GL dan balancing | Tidak | Buku harus seimbang sebelum hari baru |
| Ganti tanggal bisnis | Tidak | Syarat membuka layanan hari berikutnya |
| Laporan internal dan ekstraksi | Ya | Tidak memblokir layanan nasabah |
| Interface ke regulator dan mitra | Ya, sampai tenggat masing-masing | Tenggat dicatat dan dipantau |

```
ALGORITMA KeputusanEODTerlambat(sekarang,
batas_buka_layanan)
  sisa_kritis <- jumlah perkiraan durasi fase kritis yang
belum selesai (median riwayat)
  cadangan <- batas_buka_layanan - sekarang - sisa_kritis
  jika cadangan < 60 menit:
      tunda semua fase yang boleh ditunda; jalankan
setelah BOD
      beri tahu manajer jaga dan unit bisnis tentang
kemungkinan keterlambatan
  jika cadangan < 0:
      aktifkan prosedur pembukaan layanan terlambat;
siapkan komunikasi ke kanal
```

```
  catat keputusan dan waktunya untuk tinjauan pasca-
insiden
```

Tabel fase yang boleh ditunda harus disepakati dengan unit bisnis di jam kerja, jauh sebelum dibutuhkan. Pada malam kejadian, petugas jaga cukup menjalankan keputusan yang sudah disepakati, bukan membuat kebijakan baru.

## G.5 Keamanan: Model Ancaman Sederhana untuk Mainframe

Kontrol keamanan paling efektif bila dirancang dari ancaman yang nyata, bukan dari daftar periksa umum. Model ancaman sederhana memetakan siapa yang mungkin menyerang, lewat jalur apa, dan kontrol mana yang mencegah atau mendeteksinya.

| Ancaman | Jalur yang mungkin | Kontrol pencegahan | Deteksi |
|---|---|---|---|
| Orang dalam dengan hak berlebih | Atribut SPECIAL atau akses UPDATE yang tidak diperlukan | Hak minimum per peran; recertification (Bab 28.2) | Pemakaian hak istimewa di luar jadwal (Bab 184.1) |
| Kredensial yang dicuri | Login TSO, FTP, atau API dengan kata sandi curian | MFA untuk user berprivilegi; revoke otomatis akun tidur | Login dari lokasi atau jam tidak lazim |
| Kode jahat di library authorized | Program ditanam di library APF | Akses tulis APF sangat terbatas; review APF (Bab 29.1) | Perubahan APF dan job dari user yang sama |
| Pencurian data | Transfer keluar lewat FTP atau API | TLS, batas akses dataset, kontrol egress jaringan | Transfer besar di luar jam (SMF 119) |
| Ransomware atau perusakan data | Admin storage yang disusupi | Snapshot tak dapat diubah (Bab 113), pemisahan tugas | Pola penghapusan atau enkripsi massal |
| Mitra yang disusupi | Channel MQ atau API mitra | `MCAUSER` terbatas, `CHLAUTH`, batas laju (Bab 166.1) | Lonjakan pesan atau pola pesan tidak lazim |

Setiap baris sebaiknya punya pemilik kontrol dan cara menguji bahwa kontrolnya bekerja. Misalnya, alert pemakaian SPECIAL di luar jadwal diuji

dengan simulasi terencana setiap kuartal. Kontrol yang tidak pernah diuji sering ternyata tidak bekerja saat dibutuhkan.

Model ancaman ditinjau setiap tahun dan setiap kali ada perubahan besar, seperti kanal baru, mitra baru, atau integrasi dengan cloud. Setiap jalur masuk baru adalah ancaman baru yang perlu dipetakan.

## G.6 Keamanan dan Operasi: Mengukur Kematangan Kontrol

Pertanyaan “apakah kontrol ini sudah ada?” jarang dijawab dengan ya atau tidak saja. Sebagian besar kontrol berada di antara keduanya: ada, tetapi tidak konsisten; konsisten, tetapi tidak diukur. Skala kematangan membuat posisi setiap kontrol jelas, dan menunjukkan langkah berikutnya.

| Tingkat | Nama | Ciri |
|---|---|---|
| 1 | Ad hoc | Dilakukan bila ada yang ingat, bergantung pada orang tertentu |
| 2 | Terdokumentasi | Prosedur tertulis ada, tetapi pelaksanaannya belum konsisten |
| 3 | Konsisten | Dijalankan rutin sesuai prosedur, dengan bukti |
| 4 | Terukur | Ada metrik dan target; penyimpangan terlihat |
| 5 | Terus diperbaiki | Metrik dipakai untuk perbaikan berkala; pelajaran insiden masuk ke kontrol |

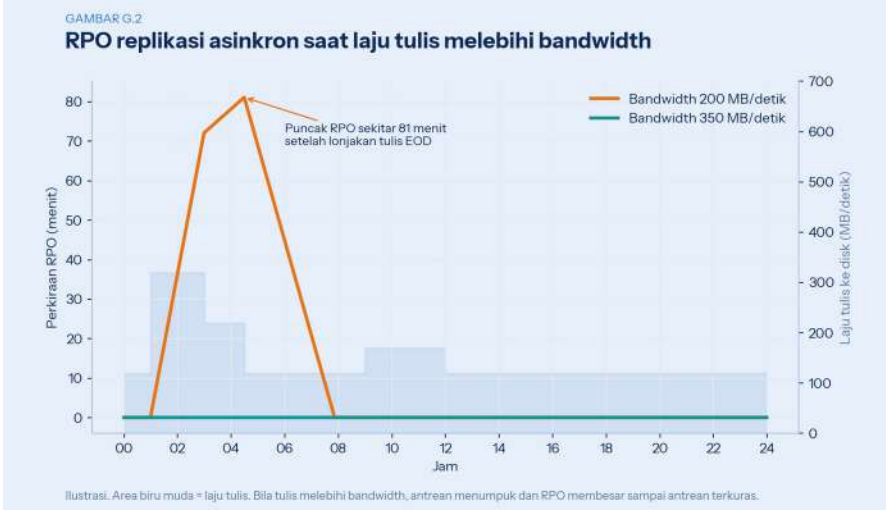
| Area contoh | Tingkat saat ini | Langkah ke tingkat berikutnya |
|---|---|---|
| Review library APF | 2 | Jadwalkan review triwulan dengan laporan otomatis (Bab 29.1) |
| Uji DR | 3 | Ukur cakupan uji dan RTO nyata setiap uji (Bab 121.1) |
| Pengelolaan sertifikat | 2 | Inventaris terpadu dengan |

| Area contoh | Tingkat saat ini | Langkah ke tingkat berikutnya |
|---|---|---|
| | | ambang per jenis (Bab 149.2) |
| Deteksi drift konfigurasi | 1 | Baseline di Git dan perbandingan harian (Bab 15.1) |
| Tinjauan pasca-insiden | 4 | Tinjau tingkat pengulangan untuk memperbaiki kualitas analisis (Bab 175.1) |

Penilaian ini paling jujur bila dilakukan bersama, dengan bukti untuk setiap tingkat yang diklaim. Tingkat 3 tanpa bukti pelaksanaan rutin sebenarnya tingkat 2. Laporkan hasilnya setahun sekali sebagai peta jalan perbaikan. Target yang wajar adalah menaikkan beberapa area satu tingkat per tahun, bukan semua area sekaligus.

## G.7 DR: Bandwidth Replikasi dan RPO Asinkron

Replikasi asinkron jarak jauh, seperti Global Mirror, mengirim perubahan data ke situs DR dengan sedikit jeda. Jeda itu menentukan RPO. Selama bandwidth antar-situs lebih besar dari laju tulis ke disk, jeda tetap kecil, biasanya dalam hitungan detik. Tetapi saat laju tulis melebihi bandwidth, perubahan menumpuk dan RPO membesar.




*Gambar G.2 — RPO replikasi asinkron saat laju tulis melebihi bandwidth*

$$Q(t) = \max(0,\ Q(t-\Delta t) + (W(t) - B)\,\Delta t) \qquad RPO(t) \approx \frac{Q(t)}{B}$$

Q adalah antrean perubahan yang belum terkirim, W laju tulis, dan B bandwidth efektif. Dalam ilustrasi, laju tulis di puncak EOD mencapai 320 MB per detik selama dua jam, sementara bandwidth hanya 200 MB per detik. Antrean tumbuh sekitar 120 MB setiap detik. Di puncaknya, RPO mencapai sekitar 81 menit, dan butuh beberapa jam lagi sampai antrean terkuras. Dengan bandwidth 350 MB per detik, antrean tidak pernah terbentuk.

Ini berarti RPO yang dijanjikan, misalnya “kurang dari 5 menit”, hanya berlaku di luar puncak tulis bila bandwidth tidak cukup. Bencana yang terjadi pukul 03.00, di tengah EOD, justru akan kehilangan data paling banyak.

| Pilihan | Dampak |
|---|---|
| Bandwidth di atas puncak tulis, dengan cadangan | RPO kecil sepanjang hari |
| Kompresi di jalur replikasi | Bandwidth efektif naik tanpa menambah jalur |
| Mengurangi tulis yang tidak perlu saat EOD | Misalnya dataset sementara di storage group yang tidak direplikasi |

| Pilihan | Dampak |
|---|---|
| Menerima RPO lebih besar di jam EOD | Harus ditulis sebagai keputusan dan disetujui pemilik risiko |

Ukur laju tulis per jam dari laporan RMF DASD selama beberapa pekan, termasuk akhir bulan, lalu bandingkan puncaknya dengan bandwidth replikasi yang tersedia.

## G.8 DR: Menangani Kegagalan Sebagian Situs

Rencana DR sering hanya membahas dua keadaan: situs utama normal, atau situs utama hilang seluruhnya. Dalam praktik, kegagalan yang lebih sering adalah kegagalan sebagian: satu array storage bermasalah, satu CPC mati, atau jaringan antar-situs terganggu. Menyatakan bencana penuh untuk kegagalan sebagian bisa menimbulkan gangguan yang lebih besar daripada kegagalannya sendiri.

| Kegagalan | Respons pertama | Perlu deklarasi DR penuh? |
|---|---|---|
| Satu array storage di situs utama | HyperSwap ke salinan sekunder (Bab 44.1) | Tidak, bila HyperSwap berhasil |
| Satu CPC dari dua di situs utama | Beban berpindah ke sistem lain di sysplex; kapasitas cadangan (Bab 11.1) | Tidak, bila kapasitas tersisa cukup |
| Coupling Facility | Rebuild atau duplex mengambil alih (Bab 179.1) | Tidak |
| Jaringan antar-situs | Replikasi tertunda; RPO membesar | Tidak, tetapi risiko naik: pulihkan jaringan segera |
| Jaringan kanal ke situs utama | Arahkan kanal lewat jalur cadangan | Tergantung ketersediaan jalur cadangan |
| Beberapa komponen sekaligus, atau penyebab belum diketahui | Kumpulkan informasi cepat; siapkan situs DR | Diputuskan pemimpin insiden berdasarkan kriteria tertulis |

Kriteria deklarasi DR penuh harus ditulis sebelumnya dan disetujui manajemen. Contohnya: layanan inti tidak dapat dipulihkan di situs utama dalam waktu tertentu, atau situs utama tidak dapat diakses secara fisik.

Tanpa kriteria tertulis, keputusan ini cenderung diambil terlalu cepat karena panik, atau terlalu lambat karena ragu.

Setiap kegagalan sebagian juga mengurangi perlindungan yang tersisa. Setelah HyperSwap, situs utama berjalan tanpa salinan sinkron sampai array diperbaiki. Setelah satu CPC mati, sysplex berjalan tanpa cadangan N+1. Periode ini harus diperlakukan sebagai risiko tinggi, dengan perubahan ditunda dan pemantauan diperketat sampai redundansi pulih (Bab 94.1).

# Lampiran H. Bank Soal dan Studi Latihan

Lampiran ini dapat dipakai untuk ujian internal tim, penilaian sebelum seseorang masuk rotasi jaga, atau belajar mandiri. Soal pilihan ganda menguji pemahaman konsep, soal hitungan menguji kemampuan memakai rumus di buku ini, dan studi latihan menguji cara berpikir dalam situasi yang mirip kenyataan. Kunci dan rujukan bab ada di H.5.

## H.1 Soal Pilihan Ganda

**Platform dan sistem**

1. Karakterisasi prosesor yang menjalankan beban Java yang eligible tanpa menambah MSU software z/OS: (a) CP (b) zIIP (c) SAP (d) ICF
2. Saat CP bersama diperebutkan, porsi kapasitas setiap LPAR ditentukan oleh: (a) jumlah logical CP (b) bobot PR/SM (c) ukuran memori (d) urutan aktivasi
3. Langkah terpenting sebelum mengaktifkan IODF baru secara dinamis: (a) IPL (b) `ACTIVATE ... TEST` (c) CLPA (d) `SETXCF START,REALLOCATE`
4. PTF HIPER yang relevan dengan konfigurasi Anda sebaiknya: (a) menunggu RSU tahun depan (b) dipasang di luar siklus setelah uji singkat (c) tidak dipasang (d) langsung dipasang di produksi tanpa uji
5. Setelah job pack area, urutan pencarian program saat `PGM=` adalah: (a) LNKLST, LPA, STEPLIB (b) STEPLIB/JOBLIB, LPA, LNKLST (c) LPA, STEPLIB, LNKLST (d) hanya STEPLIB

6. Syarat utama agar sebuah program boleh ditandai threadsafe: (a) ditulis dalam Java (b) tidak memperbarui storage bersama tanpa serialisasi (c) tidak memakai Db2 (d) dikompilasi dengan `OPT(0)`

**Data dan batch**

7. Kapasitas maksimum dataset non-extended di satu volume dengan `SPACE=(CYL,(P,S))`: (a) P + S (b) P + 15 × S (c) 16 × P (d) 15 × P + S
8. `DISP` output yang paling aman untuk rerun: (a) `(NEW,CATLG,CATLG)` (b) `MOD` (c) `(NEW,CATLG,DELETE)` (d) `OLD`
9. CI split dan CA split sebuah KSDS terlihat di: (a) SMF 30 (b) `LISTCAT` (c) SYSLOG (d) RMF Monitor III
10. Job dengan slack nol berarti: (a) di luar jalur kritis (b) di jalur kritis (c) tidak penting (d) sudah selesai
11. Operator ICETOOL untuk mengambil semua record dengan kunci ganda: (a) `COUNT` (b) `STATS` (c) `SELECT ... ALLDUPS` (d) `SORT`
12. File interface yang trailer-nya tidak cocok dengan isinya harus: (a) diproses sebagian (b) dikarantina per record (c) ditolak seluruhnya (d) diabaikan

**Kinerja**

13. Komponen waktu respons I/O yang naik karena alias PAV kurang: (a) IOSQ (b) PEND (c) DISC (d) CONN
14. Tujuan WLM untuk transaksi online sebaiknya berbasis: (a) rata-rata (b) persentil (c) velocity (d) discretionary
15. PI sebuah service class di atas 1 berarti: (a) tujuan tercapai (b) tujuan meleset (c) CPU penuh (d) WLM tidak aktif
16. Buffer pool dengan 200.000 getpage per detik dan hit ratio 98% melakukan sekitar: (a) 400 (b) 4.000 (c) 40.000 (d) 196.000 I/O sinkron per detik
17. `ACCESSTYPE = R` pada transaksi online berarti: (a) akses index unik (b) tablespace scan (c) akses lewat MQT (d) akses baca saja
18. Kategori delay RMF Monitor III untuk pekerjaan yang menunggu mount tape: (a) CPU (b) Device (c) Operator (d) Storage

**Ketersediaan dan DR**

19. Dengan tiga sistem dan batas utilisasi 85% saat satu sistem gagal, utilisasi normal maksimum per sistem sekitar: (a) 42,5% (b) 56,7% (c) 63,7% (d) 85%
20. Penyebab HyperSwap gagal yang paling sering: (a) CPU penuh (b) pasangan replikasi suspended tanpa alert (c) spool penuh (d) terlalu banyak LPAR
21. Structure yang cukup dilindungi dengan jalur ganda di CF berbeda, tanpa duplexing: (a) lock Db2 (b) GBP Db2 (c) signaling XCF (d) lock RLS
22. Retained lock di data sharing dilepas setelah: (a) CF di-restart (b) anggota yang gagal di-restart dan menyelesaikan unit kerjanya (c) semua anggota di-IPL (d) operator menjalankan `FORCE`
23. Titik tanpa kembali yang sebenarnya dalam switchover terencana: (a) menghentikan kanal (b) swap storage (c) IPL di situs DR (d) membuka kanal di situs DR
24. Uji DR yang lengkap wajib membuktikan: (a) sistem bisa di-IPL (b) subsistem bisa start (c) data terenkripsi dapat dibaca dan transaksi berjalan (d) jaringan tersambung

**Keamanan**

25. Bila ada profil `BANK.PROD.**` dan `BANK.PROD.CIF.**`, akses ke `BANK.PROD.CIF.MASTER` ditentukan oleh: (a) gabungan keduanya (b) profil yang lebih spesifik saja (c) profil yang lebih umum saja (d) yang dibuat lebih dulu
26. Perubahan profil di class yang di-RACLIST berlaku setelah: (a) IPL (b) `SETROPTS RACLIST(class) REFRESH` (c) user logoff (d) 24 jam
27. Temuan review APF yang paling berbahaya: (a) library jarang dipakai (b) APF menunjuk dataset yang tidak ada (c) audit tidak aktif (d) nama library panjang
28. Master key kriptografi dimuat dengan prinsip: (a) satu administrator (b) dual control (c) otomatis dari skrip (d) dari file konfigurasi

29. Bagian TLS yang paling memakan CPU: (a) enkripsi data simetris (b) handshake (c) penutupan koneksi (d) kompresi
30. Total kontrol yang mendeteksi record tertukar rekening dengan nominal sama: (a) jumlah record (b) total nominal (c) hash total (d) tidak ada

## H.2 Soal Hitungan

1. File berisi 3 juta record FB dengan LRECL 80 dan BLKSIZE optimal setengah track. Berapa silinder yang dibutuhkan? (Bab 18.3)
2. Sebuah sistem melayani 1.200 transaksi per detik dengan waktu rata-rata 0,08 detik di CICS, dibagi rata ke enam AOR. Berapa rata-rata task aktif per AOR? (Bab 35.2)
3. Berapa window TCP yang dibutuhkan untuk mengisi link 1 Gbps dengan RTT 30 ms? (Bab 39.1)
4. Garis dasar ECSA sekarang 70%, naik 1,5% per hari, dan ambang kritis 85%. Berapa hari tersisa? (Bab 10.4)
5. Peluang insiden 2% per perubahan dan direncanakan 25 perubahan. Berapa peluang minimal satu insiden? (Bab 126.1)
6. SLO 99,95%, dan tingkat kesalahan saat ini 0,7%. Berapa burn rate-nya? (Bab 185.1)
7. Checkpoint memakan 30 detik, dan job rata-rata gagal sekali per 12 jam berjalan. Berapa interval checkpoint optimal? (Bab 60.1)
8. Antrean MQ dengan `MAXDEPTH` 300.000 kini berisi 60.000 pesan, konsumen berhenti, dan pesan masuk 150 per detik. Berapa menit sampai penuh? (Bab 34.1)
9. Pemakaian puncak sekarang 60% dari kapasitas, batas aman 85%, dan pertumbuhan 15% per tahun. Berapa bulan sampai kapasitas habis? (Bab 172.1)
10. Sistem melakukan 150.000 permintaan CF sinkron per detik dengan waktu layanan 8 mikrodetik. Berapa CPU yang terpakai untuk menunggu CF? (Bab 43.1)

## H.3 Penyelesaian Soal Hitungan

| No | Penyelesaian | Jawaban |
|---|---|---|
| 1 | 698 record per track; 3.000.000 / 698 dibulatkan ke atas = 4.298 track; 4.298 / 15 dibulatkan ke atas | 287 silinder |
| 2 | 1.200 × 0,08 = 96 task aktif; 96 / 6 | 16 task per AOR |
| 3 | 125 MB/detik × 0,030 detik | sekitar 3,75 MB |
| 4 | (85 − 70) / 1,5 | 10 hari |
| 5 | 1 − 0,98^25 = 1 − 0,603 | sekitar 40% |
| 6 | 0,7% / (100% − 99,95%) = 0,7 / 0,05 | 14 |
| 7 | √(2 × 0,5 menit × 720 menit) = √720 | sekitar 27 menit |
| 8 | (300.000 − 60.000) / 150 = 1.600 detik | sekitar 27 menit |
| 9 | ln(85/60) / ln(1,15) = 0,348 / 0,140 = 2,49 tahun | sekitar 30 bulan |
| 10 | 150.000 × 8 µs = 1,2 detik CPU per detik | setara sekitar 1,2 prosesor |

## H.4 Studi Latihan

Setiap studi latihan dijawab dalam satu sampai dua halaman: langkah investigasi, dugaan akar masalah beserta alasannya, tindakan jangka pendek dan jangka panjang, serta cara membuktikan bahwa masalah sudah selesai. Rubrik di akhir berlaku untuk semua studi.

**Studi 1 — EOD melambat 40% sejak Senin.** Tidak ada rilis aplikasi pekan ini. Satu-satunya perubahan tercatat adalah penambahan volume disk ke storage group produksi pada hari Jumat. Susun langkah investigasi dari data yang tersedia (Bab 22.1, 6.3, 62.1).

**Studi 2 — Mitra melaporkan transfer ganda.** Payment hub menyatakan sekitar 300 transfer tercatat dua kali di core banking setelah gangguan

jaringan 5 menit pukul 14.10. Jelaskan kemungkinan penyebabnya, data yang harus dicocokkan, dan perbaikan desainnya (Bab 37.1, 102.3, 54.1).

**Studi 3 — Uji DR: aplikasi tidak bisa membuka file.** Sistem di situs DR menyala dan Db2 serta CICS aktif, tetapi aplikasi pembiayaan gagal membuka tiga file VSAM. Susun daftar kemungkinan penyebab dan urutan pemeriksaannya (Bab 121.1, 15.1, 31.1, 13.1).

**Studi 4 — Audit menemukan 15 user dengan atribut SPECIAL.** Rancang rencana 90 hari untuk menurunkan jumlah itu tanpa mengganggu operasi, termasuk cara mengetahui siapa yang sebenarnya membutuhkan atribut itu (Bab 182.1, 28.2, 184.1).

**Studi 5 — Rencana upgrade z/OS di sysplex dua sistem.** Susun rencana tingkat tinggi, termasuk koeksistensi, urutan sistem, kriteria lanjut antar-tahap, dan rencana fallback (Bab 128.2, 131.1, 130.1).

| Kriteria | Bobot | Jawaban sangat baik |
|---|---|---|
| Investigasi berbasis data | 30% | Menyebut data spesifik (laporan, perintah, record SMF) dan urutan pemeriksaan yang logis |
| Ketepatan analisis | 25% | Dugaan akar masalah masuk akal dan dibedakan dari gejala |
| Tindakan | 20% | Membedakan tindakan jangka pendek dan jangka panjang, dengan jalan kembali |
| Bukti penyelesaian | 15% | Menyebut cara membuktikan masalah selesai: angka, total kontrol, atau uji |
| Komunikasi | 10% | Ringkas, bisa dipahami manajer jaga dan unit bisnis |

### H.5 Kunci Soal Pilihan Ganda

| No | Kunci | Bab | No | Kunci | Bab | No | Kunci | Bab |
|---|---|---|---|---|---|---|---|---|
| 1 | b | 4.3 | 11 | c | 20.1 | 21 | c | 179.1 |
| 2 | b | 4.3 | 12 | c | 147.1 | 22 | b | 71.1 |
| 3 | b | 13.1 | 13 | a | 6.3 | 23 | d | 115.1 |
| 4 | b | 14.4 | 14 | b | 40.2 | 24 | c | 121.1 |
| 5 | b | 19.6 | 15 | b | 168 | 25 | b | 137.1 |
| 6 | b | 65.4 | 16 | b | 69.1 | 26 | b | 27.3 |
| 7 | b | 18.3 | 17 | b | 33.4 | 27 | b | 29.1 |
| 8 | c | 19.5 | 18 | c | 41.2 | 28 | b | 31.1 |
| 9 | b | 18.3 | 19 | b | 11.1 | 29 | b | 148.1 |
| 10 | b | 22.1 | 20 | b | 44.1 | 30 | c | 61.1 |

Nilai lulus yang disarankan untuk masuk rotasi jaga: 24 dari 30 soal pilihan ganda, 8 dari 10 soal hitungan, dan skor minimal 70% pada satu studi latihan yang dinilai oleh dua penilai.

## Lampiran I. Kumpulan Rumus

Lampiran ini mengumpulkan rumus-rumus utama di buku dalam satu tempat, ditulis dalam bentuk teks sederhana agar mudah disalin ke spreadsheet atau skrip. Arti variabel dan contoh pemakaiannya ada di bab yang dirujuk.

### I.1 Kapasitas dan Kinerja

| Rumus | Arti variabel | Bab |
|---|---|---|
| `C_i = w_i / Σw × C_shared` | Porsi kapasitas LPAR i dari bobot PR/SM saat kontensi | 4.3 |
| `RT_IO = IOSQ + PEND + DISC + CONN` | Komponen waktu respons I/O DASD | 6.3 |
| `U_normal ≤ U_maks × (n − 1) / n` | Utilisasi normal maksimum per sistem agar selamat saat | 11.1 |

| Rumus | Arti variabel | Bab |
|---|---|---|
| | satu dari n sistem gagal | |
| `L = λ × W` | Rata-rata pekerjaan aktif = laju kedatangan × waktu di sistem (hukum Little) | 21.3, 35.2, 68.2 |
| `ρ = λ × S / N` | Kesibukan N initiator dengan waktu eksekusi S | 21.3 |
| `TPS_maks ≈ U_target / t_CPU_QR` | Batas throughput satu AOR yang dibatasi QR TCB | 32.3 |
| `Throughput ≈ Window / RTT` | Batas satu koneksi TCP | 39.1 |
| `MSU_sesudah = MSU_awal × (1 − e × z)` | Efek offload zIIP; e = porsi eligible, z = porsi yang benar-benar di zIIP | 42.2 |
| `Overhead_CF ≈ N_req × t_sync` | CPU yang dipakai menunggu CF sinkron | 43.1 |
| `Sync_IO = getpage × (1 − HR)` | I/O sinkron dari hit ratio buffer pool | 69.1 |
| `T_resp = tunggu dispatch + CPU + tunggu Db2 + tunggu file + lain` | Dekomposisi waktu respons CICS | G.1 |
| `Throughput_MQ ≈ BATCHSZ / (RTT + T_commit + T_transfer)` | Batas channel MQ jarak jauh | 165.1 |
| `D_1 ≈ P85(service unit per pekerjaan)` | Durasi period pertama service class DDF | 169.1 |

## I.2 Ruang, Pertumbuhan, dan Proyeksi

| Rumus | Arti variabel | Bab |
|---|---|---|
| `rec_per_blok = floor(27998 / LRECL)` | Record per blok setengah track | 18.3 |
| `track = ceil(N / (2 × rec_per_blok)), silinder =` | Kebutuhan ruang dataset FB | 18.3 |

| Rumus | Arti variabel | Bab |
|---|---|---|
| `ceil(track / 15)` | | |
| `K_maks = P + 15 × S` | Kapasitas dataset non-extended per volume | 18.3 |
| `T_sisa = (U_ambang − U_sekarang) / ΔU_dasar_harian` | Hari tersisa sebelum batas, dari garis dasar | 10.4 |
| `T_ambang = (U_ambang − U_sekarang) / b` | Hari tersisa dari laju regresi b | 59.1 |
| `T_habis = ln(K_maks / K_sekarang) / ln(1 + g)` | Tahun sampai kapasitas habis, pertumbuhan majemuk g | 172.1 |
| `K_desain = TPS_puncak × (1 + g)^t × (1 + h)` | Kapasitas desain dari puncak, pertumbuhan, dan cadangan | 101.1 |
| `C_SGC ≈ r_ubah × C_produksi × D_retensi` | Kapasitas Safeguarded Copy kasar | 7.1 |
| `V_partisi = N_baris × s_baris × (1 + f)` | Ukuran satu partisi Db2 | 152.1 |
| `V_log_MQ ≈ N_persisten × (s + o_put + o_get)` | Volume log MQ per hari | 164.1 |
| `T_isi_log = N_log × V_log / laju_tulis` | Waktu sampai log aktif Db2 penuh | G.2 |
| `T_penuh = (MAXDEPTH − D) / λ_masuk`, `T_kuras = D / (μ − λ)` | Waktu antrean MQ penuh dan terkuras | 34.1 |
| `V_SMF = Σ N_t × s_t × D_retensi_t` | Kebutuhan ruang SMF per tipe record | 170.1 |

## I.3 Batch, EOD, dan Data

| Rumus | Arti variabel | Bab |
|---|---|---|
| `T_EOD = max_jalur Σ` | Durasi EOD dari jalur kritis | 22.1 |

| Rumus | Arti variabel | Bab |
|---|---|---|
| `d_j, slack = LS − ES` | | |
| `U_window = T_EOD / T_window, T_EOD + max T_f ≤ T_window` | Utilisasi window dan syarat ruang rerun | 104.1 |
| `Ambang_fase = rata-rata + 2 × σ` | Ambang peringatan durasi fase | 105.2 |
| `T_opt ≈ √(2 × C × M)` | Interval checkpoint optimal; C = biaya checkpoint, M = rata-rata waktu sampai gagal | 60.1 |
| `T_overhead = N / n × t_commit, T_lock = n × t_record` | Pertukaran frekuensi commit | 70.3 |
| `T_RECOVER ≈ T_restore + L × t_apply` | Waktu pemulihan tablespace | 72.5 |
| `H = (Σ nomor rekening) mod 10^k` | Hash total | 61.1 |
| `b_harian = S × r / N_hari` | Bunga harian | 97.3 |
| `p_(t+1) = p_t × M` | Proyeksi distribusi kolektibilitas dengan matriks migrasi | 99.3 |
| `Σ debit = Σ kredit, Σ saldo RAK = 0` | Keseimbangan GL | 100.3 |
| `R_kunci ≥ jendela retry + margin` | Retensi kunci idempotensi | 102.3 |
| `R_jalur ≈ 1 − Π(1 − r_k)` | Risiko kerusakan dari banyak titik konversi | 145.1 |
| `N_uji = N_fase × N_titik + N_uji_ganda` | Jumlah skenario uji restart EOD | 58.1 |

## I.4 Ketersediaan, DR, dan Keandalan

| Rumus | Arti variabel | Bab |
|---|---|---|
| `A = MTBF / (MTBF +` | Ketersediaan | 158 |

| Rumus | Arti variabel | Bab |
|---|---|---|
| MTTR) | | |
| MTTR = T_deteksi + T_diagnosis + T_perbaikan + T_verifikasi | Komponen waktu pemulihan | 57.1 |
| ALE = Σ p_s × (RTO_s × K_jam + RPO_s × K_data) | Perkiraan kerugian tahunan per skenario | 46.1 |
| R_retensi ≥ T_dwell + T_deteksi | Retensi snapshot minimum | 111.1 |
| C_DR = fungsi terbukti di DR / total fungsi kritikal | Cakupan uji DR | 121.1 |
| Q(t) = max(0, Q + (W − B) × Δt), RPO ≈ Q / B | RPO replikasi asinkron saat tulis melebihi bandwidth | G.7 |
| T_isolasi ≈ T_deteksi + ISOLATETIME + T_fencing | Waktu SFM mengisolasi sistem | 180.1 |

## I.5 Risiko, Operasi, dan Tim

| Rumus | Arti variabel | Bab |
|---|---|---|
| P(≥1 insiden) = 1 − (1 − p)^n | Peluang insiden dari n perubahan | 126.1 |
| R_total = Σ p_i × d_i | Skor risiko proyek | 131.1 |
| Prioritas = kemungkinan × dampak / usaha | Prioritas temuan hardening | 182.1 |
| Anggaran kesalahan = (1 − SLO) × N_permintaan | Kegagalan yang dapat diterima per periode | 185 |
| Burn rate = tingkat kesalahan / (1 − | Laju pemakaian anggaran kesalahan | 185.1 |

| Rumus | Arti variabel | Bab |
|---|---|---|
| SLO) | | |
| `Presisi alert = alert benar / total alert` | Kualitas alert | 30.1 |
| `Beban alert = alert butuh tindakan / jam shift` | Kelelahan alert | 47.4 |
| `Pekan jaga = 52 / N_rotasi` | Beban jaga per orang | 173.1 |
| `Faktor bus = jumlah orang dengan skor ≥ 3 di area` | Konsentrasi pengetahuan | 3.4 |
| `Rata-rata ± t × s / √n` | Selang kepercayaan benchmark | 135.1 |
| `C_per_transaksi = TCO_tahunan / N_transaksi` | Biaya per transaksi | 1.3 |
| `Hemat_MQT = Q × Δc − R × c_refresh` | Manfaat bersih MQT | 153.1 |
| `ASUTIME = k × P99,9` | Batas RLF dari distribusi | 155.1 |

# Lampiran J. Templat Kerja

Templat berikut siap disalin ke sistem tiket, wiki, atau repositori tim. Setiap templat merangkum praktik yang dibahas di bab-bab terkait, sehingga pengisinya tidak perlu mengingat semua hal yang harus diperiksa.

## J.1 Tiket Perubahan

```
# CHG-<nomor> — <ringkasan perubahan>
Pemohon: <nama/tim>   Pelaksana: <nama>   Penyetuju:
<nama>
Jadwal: <tanggal, jam mulai–selesai>   Sistem: <SYSA,
SYSB, DR>

## Apa dan mengapa
```

```
<perubahan dan alasan bisnis/teknisnya>

## Dampak dan risiko
Layanan terdampak: <...>   Tingkat risiko: rendah / sedang
/ tinggi
Apakah menyentuh jalur kritis EOD? ya / tidak (Bab 22.1)

## Pertanyaan wajib sesuai jenis (Bab 187.1)
- Transaksi baru: aturan klasifikasi WLM diperbarui?
- Rebind Db2: access path dibandingkan sebelum/sesudah?
- Deploy: semua PROC dan library diverifikasi otomatis?
- Job baru: dependensi scheduler didefinisikan?
- Perubahan di situs utama: perubahan yang sama untuk
situs DR? (Bab 189.5)

## Langkah, verifikasi, dan kembali
1. <langkah>   Verifikasi: <bukti berhasil>
Rencana kembali: <langkah dan perkiraan durasi>   Sudah
diuji: ya / tidak

## Setelah selesai
Baseline di Git diperbarui (Bab 15.1): ya / tidak
```

## J.2 Laporan Pasca-Insiden

```
# INC-<nomor> — <judul berdasarkan dampak>
Tingkat: <1–4>   Durasi dampak: <mulai – selesai>
Pemimpin insiden: <nama>

## Ringkasan (3 kalimat)
<apa yang terjadi, dampak ke nasabah, bagaimana
dipulihkan>

## Linimasa
<waktu> <kejadian>   (dari pembaruan komunikasi, Bab
174.1)

## Komponen waktu (Bab 57.1)
Deteksi: <menit>  Diagnosis: <menit>  Perbaikan: <menit>
Verifikasi: <menit>

## Akar masalah
<penyebab yang dibuktikan, bukan dugaan>
Sinyal pertama terlihat: <kapan, di data apa> (Bab 189.1)

## Tindakan (maksimal 3–5, masing-masing dengan pemilik
```

```
dan tenggat)
| Tindakan | Pemilik | Tenggat | Jenis: cegah / deteksi /
pulihkan |

## Pelajaran untuk tim
<apa yang berjalan baik, apa yang perlu diubah>
```

### J.3 Laporan Kapasitas Bulanan

```
# Laporan Kapasitas — <bulan tahun>

## Ringkasan untuk manajemen
Status: aman / perlu perhatian / perlu keputusan
<satu paragraf: angka terpenting dan keputusan yang
dibutuhkan>

## CPU dan MSU
Puncak 4HRA: <MSU>, <% dari defined capacity>   Tren 3
bulan: <naik/turun>
Pemicu keputusan tercapai? <ya/tidak> (Bab 172.1)

## Storage
Storage group mendekati ambang: <daftar dengan sisa hari>
(Bab 59.1)
Array dan salinan (FlashCopy, Safeguarded): <pemakaian>
(Bab 7.1)

## Proyeksi skenario
| Skenario | Asumsi | Kapasitas habis | Keputusan |

## Hal yang kedaluwarsa atau tumbuh diam-diam (Bab 188.1)
<daftar item dengan sisa waktu terkecil>
```

### J.4 Checklist Uji DR

```
# Uji DR — <tanggal>   Jenis: komponen / aplikasi / proses
bisnis / switchover penuh

## Sebelum
[ ] Tujuan dan kriteria berhasil disepakati
[ ] Perbandingan konfigurasi utama vs DR bersih (IODF,
CFRM, CSD, TCP/IP, master key)
[ ] Rencana kembali dan titik keputusan tertulis (Bab
115.1)
```

```
## Selama (catat waktu setiap langkah, Bab 116.1)
[ ] Storage dialihkan        waktu: ____
[ ] IPL di situs DR          waktu: ____
[ ] Subsistem aktif          waktu: ____
[ ] Data terenkripsi terbaca waktu: ____ (Bab 31.1)
[ ] Transaksi uji berhasil   waktu: ____
[ ] Total kontrol cocok      waktu: ____

## Sesudah
RTO tercapai: ____ menit (target ____)   RPO tercapai:
____ (target ____)
Temuan dan pemiliknya: <...>
Cakupan uji DR diperbarui (Bab 121.1)
```

## J.5 Serah Terima Shift

```
# Serah terima — <tanggal> <jam>   Dari: <nama>   Ke:
<nama>
- Insiden terbuka: <nomor, status, langkah berikutnya,
pemilik>
- Job ditahan atau dijalankan ulang: <nama, alasan,
instruksi>
- Perubahan hari ini: <tiket, sistem, risiko untuk malam
ini>
- Kapasitas: spool __%, storage group terpadat __%, log
Db2 __ belum di-offload
- Redundansi yang sedang berkurang: <komponen, sejak
kapan> (Bab 94.1)
- Hal yang harus dipantau malam ini: <EOM, jatuh tempo
massal, dll.>
- Kontak eskalasi: <nama dan nomor>
```

## J.6 Rencana Gladi Rollback

```
# Gladi rollback — <proyek>   Lingkungan: <uji>   Tanggal:
<...>
Tim: <nama-nama yang sama dengan malam sebenarnya>

| Langkah | Perkiraan | Aktual | Catatan: apa yang tidak
tertulis? |

Total waktu rollback: rencana ____ menit, aktual ____
menit
Total kontrol sebelum dan sesudah rollback sama: ya /
```

```
tidak
Titik tanpa kembali pada malam cutover (dihitung mundur):
____ (Bab 130.1)
Perbaikan runbook dari gladi ini: <...>
```

## Lampiran K. Glosarium Tambahan

Istilah berikut banyak dipakai di subbab analisis dan lampiran lanjutan. Istilah dasar ada di Lampiran B.

| Istilah | Arti | Bab |
|---|---|---|
| Anggaran kesalahan | Jumlah kegagalan yang dapat diterima dalam satu periode sesuai SLO | 185 |
| Baseline | Konfigurasi atau ukuran acuan yang dipakai sebagai pembanding | 15.1 |
| Blocking | Teknik membatasi perbandingan hanya pada catatan dengan kunci sederhana yang sama | 96.3 |
| Burn rate | Laju pemakaian anggaran kesalahan dibanding laju normal | 185.1 |
| Cakupan uji DR | Porsi fungsi bisnis kritikal yang terbukti berjalan di situs DR | 121.1 |
| Change freeze | Periode pembatasan perubahan non-darurat | 126.1 |
| Drift konfigurasi | Perbedaan yang tidak disengaja antara sistem yang seharusnya identik | 15.1 |
| Dwell time | Lama penyerang berada di dalam sistem sebelum terdeteksi atau bertindak | 111.1 |
| Failure isolation | Penempatan komponen agar satu kegagalan tidak menghapus komponen dan cadangannya sekaligus | 179.1 |

| Istilah | Arti | Bab |
|---|---|---|
| Faktor bus | Jumlah orang yang mampu bekerja mandiri di satu area teknis | 3.4 |
| Gravitasi data | Kecenderungan logika paling efisien dijalankan dekat dengan datanya | 56.2 |
| Hash total | Jumlah field yang tidak bermakna bisnis, dipakai sebagai sidik jari data | 61.1 |
| Hukum Little | Hubungan rata-rata pekerjaan aktif, laju kedatangan, dan waktu di sistem | 21.3 |
| Jalur kritis | Rangkaian job terpanjang yang menentukan durasi EOD | 22.1 |
| Karantina | Pemisahan record atau file yang gagal validasi agar sisanya tetap diproses | 147.1 |
| Kelelahan alert | Menurunnya respons petugas karena terlalu banyak alert tanpa tindakan | 47.4 |
| Kematangan kontrol | Tingkat kontrol dari ad hoc sampai terus diperbaiki | G.6 |
| Kontensi palsu | Kontensi lock global karena dua resource berbeda jatuh ke entri tabel lock yang sama | 43.1 |
| Matriks migrasi | Tabel peluang perpindahan kolektibilitas antar-periode | 99.3 |
| Monte Carlo | Simulasi berulang dengan nilai acak untuk memperkirakan peluang | G.3 |
| N+1 | Kapasitas yang tetap cukup saat satu komponen gagal | 11.1 |
| Pareto | Analisis yang mengurutkan penyebab dari yang paling sering | 49.1 |

| Istilah | Arti | Bab |
|---|---|---|
| Persentil | Nilai di bawah mana sejumlah persen pengamatan berada, misalnya persentil 90 | 40.2 |
| Presisi alert | Porsi alert yang benar-benar membutuhkan tindakan | 30.1 |
| Rantai bukti | Pencatatan dan penjagaan bukti insiden agar dapat dipertanggungjawabkan | 112.1 |
| Retained lock | Lock milik anggota Db2 yang gagal, dijaga sampai anggota itu pulih | 71.1 |
| Session resumption | Pemakaian ulang sesi TLS untuk menghindari handshake penuh | 148.1 |
| Short engine | Logical CP yang hanya mendapat sebagian waktu CP fisik | 4.3 |
| Sinyal dini | Data yang sudah menunjukkan masalah sebelum menjadi insiden | 189.1 |
| Slack | Waktu sebuah job boleh terlambat tanpa menunda akhir EOD | 22.1 |
| Strangler | Pola modernisasi yang membungkus fungsi lama di balik API lalu menggantinya bertahap | 53.1 |
| Threadsafe | Program yang aman dijalankan bersamaan di open TCB | 65.4 |
| Titik tanpa kembali | Saat setelahnya kembali ke keadaan semula tidak lagi sederhana | 115.1, 130.1 |
| Uji pembaca baru | Menilai runbook dengan meminta orang yang belum pernah memakainya menjalankannya | 95.1 |

# Lampiran L. Studi Kasus K-41 sampai K-60

Dua puluh kasus terakhir melengkapi Bab 186–189 untuk area yang belum banyak dibahas: z/VM dan Linux on Z, IMS, zFS dan z/OS UNIX, z/OS Connect dan API, jaringan antarbank, serta hybrid cloud. Formatnya sama dengan kasus K-21 sampai K-40, dan semua kasus fiktif tetapi disusun dari pola yang sering terjadi.

## L.1 z/VM dan Linux on Z (K-41 s.d. K-44)

**K-41 — Semua guest Linux berhenti bersamaan**

*Gejala.* Pukul 14.20, semua guest Linux di satu anggota z/VM berhenti merespons, termasuk gateway API.

*Investigasi.* Console z/VM menunjukkan page space penuh.

*Akar masalah.* Dua pekan sebelumnya, memori belasan guest dinaikkan dua kali lipat untuk migrasi aplikasi Java “agar aman”. Page space tidak ditambah, dan pemakaiannya naik perlahan sampai habis (Bab 160.1, 162.1).

*Perbaikan.* Page space ditambah darurat. Ukuran memori guest dikembalikan sesuai kebutuhan terukur.

*Pelajaran.* Pemakaian page space z/VM adalah metrik kritis yang harus punya alert, sama pentingnya dengan storage group di z/OS.

**K-42 — Pemeliharaan z/VM tertunda karena relokasi gagal**

*Gejala.* Di tengah jendela pemeliharaan, beberapa guest menolak direlokasi ke anggota lain, dan pemeliharaan dibatalkan.

*Investigasi.* Guest tersebut memakai device khusus yang tidak didefinisikan di anggota tujuan.

*Akar masalah.* Uji kelayakan relokasi tidak dijalankan sebelum hari pemeliharaan (Bab 161.1).

*Perbaikan.* Definisi device diseragamkan di semua anggota. Uji kelayakan dijadwalkan sepekan sebelum setiap pemeliharaan.

*Pelajaran.* Relokasi yang "biasanya berhasil" harus dibuktikan sebelum jendela pemeliharaan, bukan di dalamnya.

**K-43 — Linux on Z melambat tanpa beban tambahan**

*Gejala.* Waktu respons beberapa aplikasi Linux naik bertahap selama sebulan, padahal jumlah pengguna tetap.

*Investigasi.* Laju paging z/VM naik tajam. Guest yang lambat justru memiliki memori paling besar, dan sebagian besar memorinya dipakai sebagai cache file Linux.

*Akar masalah.* Guest diberi memori jauh di atas kebutuhan kerja. Cache file Linux mengisi memori itu, dan z/VM terpaksa melakukan paging untuk semua guest (Bab 162.1).

*Perbaikan.* Memori guest diturunkan ke ukuran kerja plus cadangan kecil. Paging z/VM kembali normal.

*Pelajaran.* Di z/VM, memori berlebih pada satu guest adalah beban bagi semua guest lain.

**K-44 — Gateway di Linux on Z tetap lambat setelah migrasi**

*Gejala.* Gateway API dipindahkan dari server x86 ke Linux on Z di CPC yang sama dengan z/OS, dengan harapan latensi turun. Latensinya tidak berubah.

*Investigasi.* Lalu lintas gateway ke z/OS Connect masih lewat adapter OSA dan switch jaringan pusat data.

*Akar masalah.* Rute ke z/OS masih mengarah ke alamat jaringan lama. HiperSockets sudah dikonfigurasi, tetapi tidak pernah dipakai (Bab 36.1).

*Perbaikan.* Rute diubah ke alamat di jaringan HiperSockets. Latensi turun sesuai harapan.

*Pelajaran.* Setelah migrasi, verifikasi jalur jaringan yang benar-benar dipakai, bukan hanya jalur yang dikonfigurasi.

## L.2 IMS (K-45 s.d. K-47)

**K-45 — Pemulihan database IMS memakan enam jam**

*Gejala.* Satu database IMS rusak dan harus dipulihkan. Perkiraan tim adalah 45 menit, tetapi pemulihan memakan enam jam.

*Investigasi.* Change accumulation terakhir berumur tiga pekan, sehingga pemulihan harus membaca ulang log tiga pekan.

*Akar masalah.* Job change accumulation tidak ikut dipindahkan saat migrasi scheduler dua bulan sebelumnya, dan tidak ada yang menyadari (Bab 167.1).

*Perbaikan.* Job dijadwalkan kembali. Inventaris job pemeliharaan database dibandingkan antara scheduler lama dan baru.

*Pelajaran.* Job pemeliharaan yang tidak menghasilkan output bisnis paling mudah hilang saat migrasi, dan paling mahal saat dibutuhkan.

**K-46 — Transaksi IMS antre setelah produk baru diluncurkan**

*Gejala.* Setelah peluncuran produk tabungan baru, transaksi IMS antre pada jam sibuk. CPU LPAR masih tersisa.

*Investigasi.* Region pemroses pesan untuk kelas transaksi itu selalu penuh di jam sibuk.

*Akar masalah.* Jumlah region pemroses pesan tidak ditambah sesuai kenaikan volume transaksi produk baru.

*Perbaikan.* Region ditambah dan pembagian kelas transaksi disesuaikan.

*Pelajaran.* Peluncuran produk baru membutuhkan tinjauan kapasitas di semua lapisan, termasuk jumlah region subsistem, bukan hanya CPU.

**K-47 — Peringatan RECON diabaikan berbulan-bulan**

*Gejala.* Suatu pagi, operasi DBRC melambat dan muncul peringatan ruang dataset RECON hampir penuh.

*Investigasi.* Peringatan serupa sudah muncul di log selama berbulan-bulan, tetapi disembunyikan dari console sebagai pesan rutin.

*Akar masalah.* Catatan lama di RECON tidak pernah dibersihkan, dan pesan peringatannya dimasukkan ke daftar penekanan pesan tanpa ditinjau (Bab 50.1).

*Perbaikan.* RECON dibersihkan dan ukurannya ditinjau. Pesan peringatan dikeluarkan dari daftar penekanan dan diteruskan sebagai alert.

*Pelajaran.* Menyembunyikan pesan dari console adalah keputusan yang harus ditinjau berkala, bukan diterapkan sekali lalu dilupakan.

## L.3 zFS dan z/OS UNIX (K-48 s.d. K-50)

**K-48 — Filesystem aplikasi penuh oleh heap dump**

*Gejala.* Aplikasi Liberty gagal menulis file sesi, dan beberapa endpoint API mengembalikan error.

*Investigasi.* Filesystem aplikasi berisi puluhan heap dump Java berukuran besar.

*Akar masalah.* Aplikasi mengalami kebocoran memori setelah rilis terakhir dan kehabisan heap berulang kali. Setiap kejadian menulis heap dump ke direktori aplikasi, di filesystem yang sama dengan data aplikasi (Bab 25.2).

*Perbaikan.* Heap dump dipindahkan ke filesystem khusus diagnosis. Kebocoran memori diperbaiki di rilis berikutnya.

*Pelajaran.* Output diagnosis harus punya tempat sendiri, agar masalah pertama tidak memicu masalah kedua.

**K-49 — Aplikasi gagal start setelah IPL**

*Gejala.* Setelah IPL terencana, satu aplikasi Java gagal start karena direktori konfigurasinya tidak ada.

*Investigasi.* Filesystem konfigurasi aplikasi tidak ter-mount.

*Akar masalah.* Filesystem itu di-mount manual sebulan sebelumnya saat aplikasi dipasang, dan tidak pernah dimasukkan ke konfigurasi mount permanen.

*Perbaikan.* Mount ditambahkan ke konfigurasi permanen, dan perbandingan filesystem yang ter-mount sebelum dan sesudah IPL ditambahkan ke checklist IPL.

*Pelajaran.* Setiap perubahan dinamis, baik mount, perintah SETPR0G, maupun OBEYFILE, harus dicatat ke konfigurasi permanen (Bab 38.1).

**K-50 — Skrip pembersihan menghapus direktori yang salah**

*Gejala.* Direktori konfigurasi sebuah aplikasi hilang, dan aplikasi gagal saat restart berikutnya.

*Investigasi.* Skrip pembersihan log berjalan dengan UID 0. Variabel direktori kosong karena kesalahan di file parameter, sehingga perintah hapus berjalan di direktori induk.

*Akar masalah.* Skrip berjalan dengan hak superuser permanen yang tidak diperlukan, tanpa pemeriksaan variabel kosong (Bab 182, 84.1).

*Perbaikan.* Direktori dipulihkan dari backup. Skrip dijalankan dengan user terbatas dan pemeriksaan variabel yang ketat.

*Pelajaran.* Hak superuser membuat satu kesalahan kecil menjadi kerusakan besar. Hak minimum adalah kontrol keselamatan, bukan hanya kontrol keamanan.

## L.4 z/OS Connect dan API (K-51 s.d. K-54)

**K-51 — Beban API naik lima kali lipat setelah rilis**

*Gejala.* Setelah rilis API pembayaran tagihan, jumlah panggilan API naik sekitar lima kali lipat, padahal jumlah transaksi berhasil tidak berubah.

*Investigasi.* Sebagian besar panggilan adalah retry dari aplikasi klien untuk transaksi yang ditolak karena saldo tidak cukup.

*Akar masalah.* API mengembalikan kode kesalahan server untuk penolakan bisnis. Klien menafsirkannya sebagai gangguan sementara dan mencoba ulang berkali-kali (Bab 54.1).

*Perbaikan.* Penolakan bisnis dipetakan ke kode kesalahan klien dengan alasan yang jelas, sehingga klien berhenti mencoba ulang.

*Pelajaran.* Pemetaan kode kesalahan adalah bagian dari desain kapasitas: kode yang salah bisa melipatgandakan beban.

**K-52 — Transfer berhasil di core, gagal di aplikasi nasabah**

*Gejala.* Nasabah melaporkan transfer "gagal" di aplikasi, tetapi saldonya berkurang.

*Investigasi.* Gateway menyerah setelah 30 detik, sedangkan z/OS Connect dan CICS tetap memproses sampai selesai di detik ke-35 pada jam sibuk.

*Akar masalah.* Rantai timeout terbalik: lapisan atas menyerah lebih cepat daripada lapisan bawah (Bab 54.1).

*Perbaikan.* Timeout disusun ulang dari bawah ke atas dengan margin. Status inquiry ditambahkan agar aplikasi dapat menanyakan hasil transaksi yang terputus.

*Pelajaran.* Setiap lapisan harus menyerah sedikit lebih cepat daripada lapisan di atasnya.

**K-53 — Aplikasi mitra rusak setelah perubahan kecil pada API**

*Gejala.* Setelah rilis kecil, aplikasi mitra e-commerce gagal memproses respons API saldo.

*Investigasi.* Satu field di respons diganti nama agar lebih konsisten.

*Akar masalah.* Perubahan yang tidak kompatibel dirilis di versi API yang sama, tanpa masa transisi (Bab 54.1).

*Perbaikan.* Nama field lama dikembalikan. Perubahan dipindahkan ke versi API baru dengan masa transisi yang diumumkan.

*Pelajaran.* Untuk API yang dipakai pihak luar, tidak ada perubahan yang terlalu kecil untuk membutuhkan versi.

**K-54 — Promo mitra memicu puncak MSU**

*Gejala.* Tagihan software bulan berikutnya naik tajam. Laporan SCRT menunjukkan puncak 4HRA baru pada satu hari tertentu.

*Investigasi.* Pada hari itu, satu mitra e-commerce menjalankan promo besar. Panggilan API cek saldo dari mitra itu naik puluhan kali lipat selama beberapa jam.

*Akar masalah.* Tidak ada batas laju per mitra di gateway API (Bab 54.1, 42.2).

*Perbaikan.* Batas laju per mitra diterapkan, dan perjanjian dengan mitra diperbarui agar promo besar diberitahukan lebih dulu.

*Pelajaran.* Perilaku satu mitra dapat menentukan tagihan software seluruh bank. Batas laju adalah kontrol biaya, bukan hanya kontrol keamanan.

## L.5 Jaringan Antarbank (K-55 s.d. K-57)

**K-55 — Layanan transfer antarbank berhenti karena jalur cadangan sudah lama mati**

*Gejala.* Jalur jaringan utama ke penyelenggara sistem pembayaran putus, dan layanan transfer antarbank berhenti sekitar 40 menit.

*Investigasi.* Jalur cadangan tidak mengambil alih, karena ternyata sudah tidak berfungsi sejak tiga bulan sebelumnya.

*Akar masalah.* Status jalur cadangan tidak dipantau, karena tidak membawa lalu lintas selama jalur utama sehat.

*Perbaikan.* Jalur cadangan diperbaiki. Pemantauan aktif ditambahkan, dan pengalihan ke jalur cadangan diuji terjadwal.

*Pelajaran.* Komponen cadangan yang tidak pernah dipakai harus diuji secara aktif. Diam bukan berarti sehat (Bab 188.1).

**K-56 — Transfer file ke mitra gagal setelah perubahan firewall mitra**

*Gejala.* Transfer file harian ke satu bank mitra gagal di tengah sesi: login berhasil, tetapi pengiriman data macet.

*Investigasi.* Koneksi kendali terbentuk, tetapi koneksi data tidak pernah tersambung.

*Akar masalah.* Mitra memperbarui firewall dan menutup rentang port yang dipakai koneksi data, tanpa pemberitahuan (Bab 39.1).

*Perbaikan.* Rentang port dibuka kembali setelah koordinasi. Spesifikasi interface diperbarui dengan rentang port yang disepakati.

*Pelajaran.* Diagnosis berlapis dari bawah ke atas dan bukti dari kedua sisi mempercepat penyelesaian masalah lintas organisasi (Bab 93.1).

**K-57 — Pesan ditolak karena stempel waktu**

*Gejala.* Sebagian pesan ke penyelenggara pembayaran ditolak dengan alasan pesan kedaluwarsa, padahal dikirim segera.

*Investigasi.* Stempel waktu di pesan dari satu server gateway berselisih lebih dari dua menit dengan waktu standar.

*Akar masalah.* Server gateway itu kehilangan sinkronisasi waktu setelah perubahan jaringan, sementara penerima memeriksa kesegaran pesan berdasarkan stempel waktu.

*Perbaikan.* Sinkronisasi waktu dipulihkan, dan selisih waktu setiap server dipantau terhadap sumber waktu yang sama dengan STP di mainframe.

*Pelajaran.* Sinkronisasi waktu adalah persyaratan fungsional, bukan hanya kenyamanan untuk membaca log (Bab 112.1).

## L.6 Hybrid Cloud (K-58 s.d. K-60)

**K-58 — Aplikasi cloud lambat walaupun mainframe cepat**

*Gejala.* Aplikasi penawaran produk di cloud membutuhkan hampir dua detik untuk menampilkan halaman profil nasabah. Semua transaksi di mainframe selesai di bawah 50 ms.

*Investigasi.* Halaman itu memanggil delapan API mainframe secara berurutan, masing-masing melintasi jaringan dengan RTT sekitar 20 ms, ditambah overhead TLS dan gateway.

*Akar masalah.* Desain API yang terlalu halus untuk pemakaian lintas lokasi (Bab 56.2).

*Perbaikan.* Dibuat satu API agregat untuk data profil yang dibutuhkan halaman itu.

*Pelajaran.* Untuk pemakaian lintas lokasi, jumlah perjalanan jaringan lebih menentukan daripada kecepatan masing-masing sistem.

**K-59 — Dashboard cloud menampilkan data yang tertinggal**

*Gejala.* Manajemen melihat saldo simpanan di dashboard pukul 07.00 jauh lebih rendah dari laporan resmi.

*Investigasi.* Replikasi data dari Db2 ke platform analitik di cloud tertinggal hampir empat jam setelah EOD.

*Akar masalah.* Laju perubahan saat EOD jauh melebihi kapasitas replikasi, dan keterlambatannya tidak dipantau (Bab G.7).

*Perbaikan.* Pemantauan jeda replikasi ditambahkan, dan dashboard menampilkan waktu data terakhir diperbarui.

*Pelajaran.* Setiap data turunan harus menunjukkan seberapa segar datanya. Data yang terlambat tanpa keterangan lebih berbahaya daripada data yang tidak ada.

**K-60 — Biaya transfer data cloud melonjak**

*Gejala.* Tagihan cloud bulan itu naik tajam pada komponen transfer data.

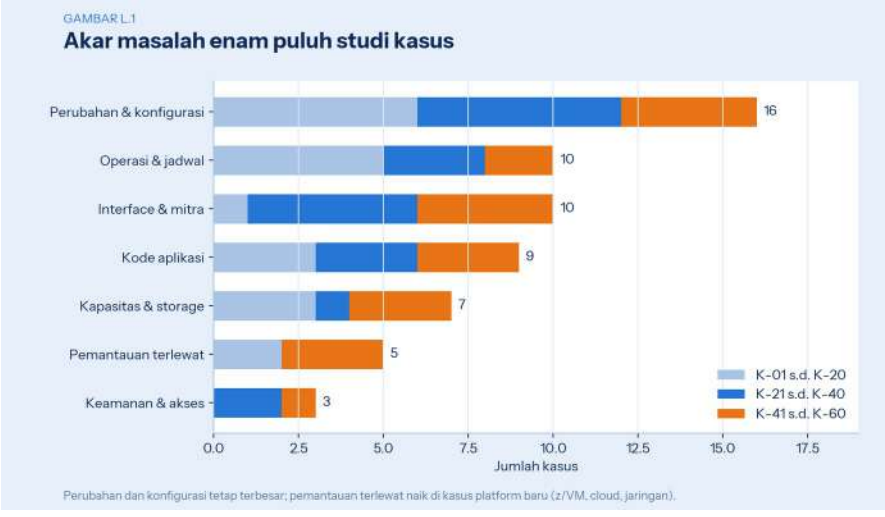
*Investigasi.* Laporan harian di cloud mengambil seluruh tabel transaksi dari mainframe setiap malam, bukan hanya perubahan sejak kemarin.

*Akar masalah.* Desain ekstraksi dibuat untuk uji coba dengan data kecil, lalu dipakai apa adanya di produksi (Bab 183.1).

*Perbaikan.* Ekstraksi diubah menjadi berbasis perubahan.

*Pelajaran.* Desain uji coba harus ditinjau ulang sebelum masuk produksi, terutama untuk volume dan biaya yang tumbuh seiring data.

## L.7 Pola dari Enam Puluh Kasus




*Gambar L.1 — Akar masalah enam puluh studi kasus*

Dengan enam puluh kasus, perubahan dan konfigurasi tetap menjadi penyebab terbesar. Kelompok kasus terakhir menambah satu pola yang jelas: pemantauan terlewat pada komponen yang jarang dilihat, seperti page space z/VM, jalur jaringan cadangan, jeda replikasi ke cloud, dan pesan yang disembunyikan dari console.

| Pelajaran tambahan dari K-41 s.d. K-60 | Kasus |
|---|---|
| Komponen cadangan dan komponen “diam” harus diuji dan dipantau aktif | K-47, K-55, K-59 |
| Perubahan dinamis wajib dicatat ke konfigurasi permanen | K-44, K-49 |
| Desain API menentukan beban, biaya, dan pengalaman nasabah | K-51, K-52, K-53, K-54, K-58 |

| Pelajaran tambahan dari K-41 s.d. K-60 | Kasus |
|---|---|
| Memori berlebih di z/VM adalah beban bagi semua guest | K-41, K-43 |
| Job pemeliharaan harus ikut diinventaris saat migrasi alat | K-45 |

## Lampiran M. Peta Silang Gejala

Peta ini menghubungkan gejala yang sering dilaporkan dengan perintah pertama, runbook, subbab analisis, studi kasus, dan praktikum yang relevan. Pakai saat insiden untuk menemukan rujukan dengan cepat, dan saat belajar untuk menelusuri satu topik dari teori sampai praktik.

| Gejala | Perintah pertama | Runbook | Analisis | Studi kasus | Praktikum |
|---|---|---|---|---|---|
| Semua transaksi online lambat | RMF Monitor III, `D WLM` | — | 41.2, 62.1, G.1 | K-01, K-12 | 6, 7 |
| Satu wilayah atau kanal lambat | PI per service class | — | 119.1 | K-12 | 6 |
| Transaksi CICS timeout | `CEMT I TASK`, `CEMT I DSAS` | RB-013, RB-014 | 32.3, 67.3, G.1 | K-02, K-23 | — |
| `SQLCODE -911` atau `-913` | `-DIS DB ... LOCKS` | RB-018 | 70.3 | K-05, K-21 | 8 |
| `SQLCODE -904` | `-DIS DB ... RESTRICT` | RB-019 | 72.5 | — | — |
| Permintaan DDF ditolak | `-DIS DDF DETAIL` | — | 68.2, 169.1 | K-27 | — |
| Job tidak jalan | `$D J`, `$D I` | RB-026 | 21.3 | K-09, K-13 | 1 |
| Job menunggu | `D GRS,RES=(SYSDSN,na` | RB-003 | 17.2 | K-10 | — |

| Gejala | Perintah pertama | Runbook | Analisis | Studi kasus | Praktikum |
|---|---|---|---|---|---|
| dataset | `ma)` | | | | |
| Abend `x37` | `D SMS,SG(...)` | RB-001 | 18.3, 89.1 | K-01 | 1 |
| Abend `S0C7` | CEEDUMP, `STATUS FAILDATA` | — | 77.2, 81.3 | K-29 | 4 |
| Abend `S806` | Job log, `D PROG,LNKLST` | RB-016 | 19.6, 79.3 | K-14 | 10 |
| Abend `S913` | `ICH408I` di SYSLOG | — | 27.3, 137.1 | K-20, K-32 | 5 |
| EOD terlambat | Tabel kontrol fase | — | 22.1, 104.1, 105.2, G.3, G.4 | K-01, K-28 | 4 |
| Bunga atau saldo tidak cocok | Total kontrol, rekonsiliasi | RB-005 | 61.1, 97.3, 100.3 | K-28, K-34 | 4 |
| File interface ditolak | Validasi header dan trailer | — | 147.1, 145.1, 146.1 | K-29, K-30 | — |
| Antrean MQ menumpuk | `DISPLAY QSTATUS`, `CHSTATUS` | RB-038 | 34.1, 165.1 | K-25, K-26 | — |
| Mitra tidak bisa terhubung | `NETSTAT CONN`, log PAGENT | RB-055 | 39.1, 93.1, 149.2 | K-06, K-56 | — |
| CPU TCP/IP tinggi | Statistik AT-TLS | — | 148.1 | K-33 | — |
| Common storage terus naik | Laporan common storage RMF | — | 10.4 | K-17 | — |
| CF atau sysplex bermasalah | `D XCF,STR`, `D XCF,PATHIN` | — | 178.1, 179.1, 180.1 | K-35, K-36 | 8 |

| Gejala | Perintah pertama | Runbook | Analisis | Studi kasus | Praktikum |
|---|---|---|---|---|---|
| Replikasi atau HyperSwap tidak siap | Status pasangan replikasi | — | 44.1, G.7 | K-15 | 9 |
| Uji DR gagal | Checklist DR (J.4) | — | 116.1, 121.1 | K-37, K-38 | 9 |
| Aktivitas user mencurigakan | SMF 80, 30, 119 | RB-049 | 112.1, 184.1, G.5 | K-31, K-32 | 5 |
| Linux on Z lambat | Laju paging z/VM | — | 160.1, 162.1 | K-41, K-43 | — |
| API lambat dari cloud | Jumlah panggilan per halaman | — | 54.1, 56.2 | K-52, K-58 | 10 |
| Tagihan MSU naik | Laporan SCRT, puncak 4HRA | — | 42.2, 155.1, 177.1 | K-40, K-54 | 7 |

Tanda — berarti belum ada runbook atau praktikum khusus untuk gejala itu. Kolom kosong seperti ini adalah daftar kerja yang baik untuk menambah runbook atau praktikum baru di tim Anda sendiri (Bab 88.1).

## Lampiran N. Menyesuaikan dengan Ukuran Bank

Praktik di buku ini disusun dengan bank menengah hingga besar sebagai gambaran utama. Bank yang lebih kecil tidak perlu, dan sering tidak mampu, menerapkan semuanya sekaligus. Bank yang lebih besar menghadapi tantangan lain: bukan kekurangan kontrol, tetapi kompleksitas yang membuat kontrol sulit dijalankan konsisten. Lampiran ini membantu memilih urutan penerapan yang sesuai.


*Gambar N.1 — Profil tiga ukuran bank*

## N.1 Matriks Kontrol per Ukuran

W berarti wajib, D disarankan, dan O opsional atau dapat ditunda.

| Kontrol | Kecil | Menengah | Besar | Bab |
|---|---|---|---|---|
| Backup terisolasi dengan uji restore berkala | W | W | W | 45.1, 134.1 |
| Runbook untuk insiden paling sering, teruji | W | W | W | 52.1, 88.1 |
| Inventaris hal yang kedaluwarsa (sertifikat, kata sandi teknis, batas GDG) | W | W | W | 188.1 |
| Validasi file | W | W | W | 61.1, 147.1 |

| Kontrol | Kecil | Menengah | Besar | Bab |
|---|---|---|---|---|
| interface dan total kontrol EOD | | | | |
| Review akses berprivilegi dan library APF | W | W | W | 29.1, 182.1 |
| Health check harian otomatis | W | W | W | 86.1 |
| Uji restart EOD di titik kegagalan tersulit | D | W | W | 58.1 |
| Deteksi drift konfigurasi produksi dan DR | D | W | W | 15.1 |
| Uji DR tahunan dengan pembacaan data terenkripsi | W | W | W | 121.1 |
| Tujuan WLM berbasis SLA dan persentil | D | W | W | 40.2 |
| Proyeksi kapasitas dengan skenario | D | W | W | 172.1 |
| Pipeline deploy dengan pemeriksaan otomatis | O | D | W | 87.1 |
| SIEM dengan aturan korelasi mainframe | D | W | W | 184.1 |
| SLO dan alert burn rate | O | D | W | 185.1 |
| Game day dan injeksi kegagalan | O | D | W | 91.1 |

| Kontrol | Kecil | Menengah | Besar | Bab |
|---|---|---|---|---|
| terkendali | | | | |
| Program kematangan kontrol | O | D | W | G.6 |

Baris-baris teratas adalah kontrol yang mencegah kehilangan data atau kegagalan layanan yang tidak dapat dipulihkan. Kontrol ini wajib di semua ukuran bank. Baris-baris bawah adalah kontrol yang mengukur dan mengoptimalkan. Kontrol ini sangat berharga, tetapi baru efektif setelah fondasinya berdiri.

## N.2 Bank Kecil: Sedikit Orang, Banyak Tanggung Jawab

Tim mainframe bank kecil sering berisi tiga sampai enam orang yang masing-masing memegang beberapa area. Faktor bus di hampir setiap area bernilai satu (Bab 3.4). Risiko terbesarnya bukan teknis, tetapi ketergantungan pada orang tertentu.

**Strategi yang terbukti membantu:**

- **Kontrak dukungan eksternal yang jelas.** Tentukan area mana yang bisa dibantu mitra atau IBM, dengan waktu respons tertulis. Pastikan mitra punya akses dan dokumentasi yang cukup sebelum dibutuhkan, bukan saat insiden.
- **Runbook untuk 20 insiden teratas.** Analisis Pareto (Bab 49.1) menunjukkan bahwa sebagian kecil jenis masalah menyumbang sebagian besar insiden. Runbook untuk jenis-jenis itu membuat anggota tim mana pun bisa menanganinya.
- **Otomasi pemeriksaan harian.** Tim kecil tidak punya waktu untuk memeriksa puluhan hal secara manual setiap pagi. Health check otomatis dengan laporan penyimpangan (Bab 86.1) adalah investasi dengan hasil tercepat.
- **DR yang sederhana tetapi teruji.** Replikasi asinkron atau backup terisolasi yang diuji setahun sekali lebih bernilai daripada solusi canggih yang tidak pernah dibuktikan.

- **Rotasi jaga yang manusiawi.** Dengan tiga orang, rotasi jaga sudah berat (Bab 173.1). Kurangi jumlah panggilan lewat perbaikan alert sebelum menambah beban.

Bank kecil juga punya keunggulan: perubahan lebih mudah dikoordinasikan, dan semua orang mengenal seluruh sistem. Manfaatkan keunggulan ini untuk menjaga dokumentasi tetap sederhana dan selalu mutakhir.

## N.3 Bank Menengah: Menjaga Konsistensi Saat Tumbuh

Bank menengah biasanya sudah memakai sysplex dua sistem dan replikasi ke situs DR. Tantangannya adalah menjaga konsistensi saat sistem dan tim tumbuh: konfigurasi yang dulu dikelola satu orang kini dikelola beberapa orang, dan situs DR yang dulu identik perlahan tertinggal.

| Tantangan umum | Kontrol yang paling membantu |
|---|---|
| Situs DR tertinggal dari produksi | Deteksi drift harian dan perbandingan IODF, CFRM, CSD, TCP/IP (Bab 15.1) |
| Sysplex dua sistem dijalankan terlalu penuh | Aturan utilisasi N+1 dan keputusan tertulis tentang mode terdegradasi (Bab 11.1) |
| Tagihan software naik seiring pertumbuhan | Pemantauan puncak 4HRA dan offload zIIP (Bab 42.2) |
| Pengetahuan mulai terpecah antartim | Peta kompetensi dan kerja berpasangan (Bab 3.4) |
| Perubahan makin sering | Pertanyaan wajib per jenis perubahan (Bab 187.1) |

Pada ukuran ini, keputusan arsitektur besar mulai muncul: apakah perlu sistem ketiga di sysplex, apakah perlu data sharing, apakah DR perlu ditingkatkan ke HyperSwap. Keputusan seperti ini sebaiknya diambil dari analisis yang terukur, seperti perhitungan N+1, ALE, dan proyeksi kapasitas, bukan dari perbandingan dengan bank lain yang ukurannya berbeda.

## N.4 Bank Besar: Mengelola Kompleksitas

Bank besar biasanya sudah memiliki hampir semua kontrol di matriks N.1. Tantangannya adalah membuat kontrol-kontrol itu berjalan konsisten di banyak sistem, banyak tim, dan banyak situs. Satu kontrol yang berjalan baik di satu sysplex tetapi terlewat di sysplex lain adalah celah yang nyata.

**Pendekatan yang membantu di skala besar:**

- **Standar platform bersama.** Satu set standar untuk konfigurasi, penamaan, dan otomasi dipakai di semua sysplex, dengan pengecualian yang tercatat.
- **Tim platform dan tim layanan.** Tim platform mengelola fondasi bersama, sedangkan tim layanan fokus pada aplikasi. Batas tanggung jawabnya ditulis jelas.
- **SLO sebagai bahasa bersama.** SLO dan anggaran kesalahan (Bab 185, 185.1) membuat tim teknis dan bisnis berbicara dengan angka yang sama.
- **Latihan kegagalan rutin.** Game day dan injeksi kegagalan terkendali (Bab 91.1) menguji interaksi antar-komponen yang tidak terlihat di uji per komponen.
- **Pengukuran kematangan.** Program kematangan kontrol (Bab G.6) menunjukkan area mana yang tertinggal di antara banyak tim.

Di bank besar, kesalahan yang paling mahal sering bukan kesalahan teknis tunggal, tetapi celah di antara tim: perubahan di satu area yang tidak diketahui area lain. Studi kasus K-25, K-35, dan K-38 adalah contohnya. Karena itu investasi pada koordinasi, seperti kalender perubahan bersama, peta dependensi, dan titik ukur bersama, sering memberi hasil lebih besar daripada investasi teknis tambahan.

## N.5 Kesalahan Umum Saat Meniru Praktik Bank Lain

| Kesalahan | Mengapa merugikan | Pendekatan yang lebih baik |
|---|---|---|
| Meniru arsitektur DR bank yang jauh lebih besar | Biaya tinggi tanpa tim untuk mengujinya | Pilih berdasarkan ALE dan kemampuan tim (Bab 46.1) |
| Menerapkan semua kontrol | Tim kewalahan; tidak ada | Ikuti urutan matriks N.1, dari |

| Kesalahan | Mengapa merugikan | Pendekatan yang lebih baik |
|---|---|---|
| sekaligus | yang selesai | atas ke bawah |
| Membeli alat sebelum proses siap | Alat tidak dipakai atau dipakai salah | Jalankan proses manual dulu, lalu otomasi |
| Menyalin ambang dan target dari tempat lain | Tidak sesuai beban dan data sendiri | Tetapkan dari data 30 hari terakhir (Bab 105.2) |
| Mengukur banyak hal tanpa keputusan | Laporan panjang yang tidak dibaca | Setiap metrik terkait satu keputusan atau tindakan |

Prinsip yang berlaku di semua ukuran bank sederhana: ukur lingkungan sendiri, mulai dari kontrol yang melindungi data dan layanan, lalu naik bertahap. Buku ini menyediakan cara berpikir dan rumusnya. Angka dan prioritasnya harus datang dari bank Anda sendiri.

## Lampiran O. Rencana 90 Hari

Tiga rencana berikut membantu mengubah isi buku menjadi langkah nyata dalam tiga bulan pertama. Rencana pertama untuk system engineer yang baru bergabung. Rencana kedua untuk kepala tim yang baru menjabat. Rencana ketiga untuk tim yang ingin menjalankan program perbaikan dari temuan-temuan di buku ini.


*Gambar O.1 — Tiga rencana 90 hari*

Setiap rencana dibagi menjadi tiga fase sekitar 30 hari. Setiap fase diakhiri hasil yang bisa dibuktikan, bukan sekadar “sudah belajar” atau “sudah dibahas”.

## O.1 System Engineer yang Baru Bergabung

| Fase | Kegiatan | Hasil yang dibuktikan |
|---|---|---|
| Hari 1–14: orientasi | Akses dan alat siap; baca Bab 1–3, 9.1, dan 192.2; kenali topologi sistem, sysplex, dan situs DR; ikut serah terima shift | Satu diagram topologi lingkungan kerja, digambar sendiri dan diperiksa pembimbing |
| Hari 15–45: jaga berpasangan | Ikut jaga bersama anggota senior; kerjakan Praktikum 1–5; baca runbook 20 insiden teratas | Menangani tiga insiden rutin dengan runbook, diawasi senior; portofolio praktikum |
| Hari 46–90: memimpin perubahan kecil | Memimpin satu perubahan berisiko rendah dari tiket sampai verifikasi (Templat J.1); satu uji pembaca baru untuk runbook (Bab 95.1) | Satu perubahan selesai tanpa insiden; satu runbook diperbaiki berdasarkan pengalaman sendiri |

**Pertanyaan yang sebaiknya bisa dijawab di hari ke-90:** Di mana jalur kritis EOD, dan berapa slack-nya? Apa yang dilakukan pertama kali bila transaksi teller lambat? Siapa yang dihubungi bila CF bermasalah pukul tiga pagi? Bagaimana membuktikan bahwa backup bisa dipulihkan?

Pembimbing sebaiknya bertemu anggota baru setiap pekan selama 30 menit. Pertemuan singkat yang rutin lebih efektif daripada sesi panjang yang jarang, dan memberi ruang untuk pertanyaan yang mungkin terasa terlalu sederhana untuk ditanyakan di depan tim.

## O.2 Kepala Tim yang Baru Menjabat

| Fase | Kegiatan | Hasil yang dibuktikan |
|---|---|---|
| Hari 1–30: mendengar dan memetakan | Pertemuan empat mata dengan setiap anggota tim; peta kompetensi dan faktor bus (Bab 3.4); daftar insiden | Peta kompetensi tim; lima risiko teratas yang disepakati tim |

| Fase | Kegiatan | Hasil yang dibuktikan |
|---|---|---|
| | 6 bulan terakhir dengan analisis Pareto (Bab 49.1); daftar risiko terbuka | |
| Hari 31–60: prioritas dan metrik | Pilih tiga metrik utama, misalnya beban jaga, MTTR, dan cakupan uji DR; tetapkan baseline; tentukan tiga perbaikan dengan dampak terbesar | Dasbor satu halaman dengan tiga metrik dan baseline-nya; tiga perbaikan dengan pemilik dan tenggat |
| Hari 61–90: rencana tahunan | Susun rencana tahunan: pemeliharaan, uji DR, audit, pelatihan (Bab 122–127); ajukan kebutuhan anggaran dengan angka (Bab 177.1, 172.1) | Rencana tahunan yang disetujui; kalender kegiatan besar; rencana pengembangan tiap anggota |

Godaan terbesar kepala tim baru adalah mengubah banyak hal di bulan pertama. Tahan godaan itu. Bulan pertama untuk memahami mengapa tim bekerja seperti sekarang. Banyak praktik yang terlihat aneh dari luar punya alasan yang baik, dan beberapa praktik yang terlihat wajar ternyata menyimpan risiko besar.

**Tiga angka yang sebaiknya diketahui kepala tim di hari ke-90:** jumlah area dengan faktor bus satu, beban jaga per orang per bulan, dan tanggal terakhir setiap fungsi kritikal terbukti berjalan di situs DR.

## O.3 Program Perbaikan 90 Hari

Program ini untuk tim yang sudah membaca buku ini dan ingin menerapkan pelajarannya secara terstruktur, bukan satu per satu secara acak.

| Fase | Kegiatan | Hasil yang dibuktikan |
|---|---|---|
| Hari 1–20: penilaian dasar | Nilai kematangan 10–15 area kontrol (Bab G.6); jalankan pemeriksaan matriks kontrol untuk ukuran bank Anda (Lampiran N); petakan insiden 12 bulan terakhir ke | Tabel kematangan dengan bukti; daftar celah terhadap matriks N.1 |

| Fase | Kegiatan | Hasil yang dibuktikan |
|---|---|---|
| | kategori akar masalah (Gambar L.1) | |
| Hari 21–70: lima kontrol utama | Terapkan lima kontrol dengan dampak terbesar dari penilaian. Kandidat umum ada di bawah | Setiap kontrol berjalan, diuji, dan punya pemilik |
| Hari 71–90: ukur dan laporkan | Bandingkan metrik dengan baseline; laporkan kepada manajemen; pilih lima kontrol berikutnya | Laporan satu halaman: apa yang berubah, apa buktinya, apa selanjutnya |

**Kandidat lima kontrol utama** (pelajaran dari 60 studi kasus, Bab 189.5 dan L.7):

1. Perbandingan konfigurasi harian antara produksi dan situs DR.
2. Pembuktian redundansi setelah setiap pemeliharaan: jalur sinyal, duplex, dan pasangan replikasi.
3. Kontrak interface eksplisit dengan setiap mitra: format, encoding, urutan, koneksi, sertifikat, dan batas laju.
4. Uji restart EOD di titik kegagalan tersulit untuk fase-fase jalur kritis.
5. Inventaris terpadu untuk hal yang kedaluwarsa dan komponen cadangan yang “diam”.

Pilih lima yang paling relevan dari hasil penilaian, bukan dari daftar ini secara buta. Tim yang insidennya didominasi masalah mitra sebaiknya mendahulukan kontrol nomor tiga. Tim yang baru gagal uji DR sebaiknya mendahulukan nomor satu.

## O.4 Setelah 90 Hari

Rencana 90 hari adalah awal, bukan akhir. Setelah tiga bulan, ulangi siklus yang sama dengan fokus berikutnya: anggota baru mulai masuk rotasi jaga penuh, kepala tim menjalankan rencana tahunan, dan program perbaikan memilih lima kontrol berikutnya.

Yang membuat perbaikan bertahan bukan besarnya perubahan, tetapi ritmenya. Tinjauan bulanan yang singkat, dengan angka yang sama setiap bulan, lebih kuat daripada proyek besar yang dijalankan sekali lalu dilupakan. Setelah setahun, tim yang menjalankan ritme ini dengan konsisten akan melihat hasilnya sendiri: insiden berkurang, beban jaga turun, dan uji DR yang dulu menegangkan menjadi rutinitas yang dapat diandalkan.

# Lampiran P. Jawaban Model Studi Latihan

Lampiran ini memberi satu contoh jawaban untuk setiap studi latihan di H.4. Jawaban model bukan satu-satunya jawaban benar. Jawaban lain yang berbeda urutan tetapi sama kuat buktinya tetap layak mendapat nilai tinggi. Setiap jawaban diakhiri catatan penilai tentang apa yang membedakan jawaban baik dari jawaban cukup.

## P.1 Studi 1 — EOD Melambat 40% Sejak Senin

**Langkah investigasi.**

1. Bandingkan durasi setiap fase dari tabel kontrol EOD untuk dua pekan terakhir (Bab 105.2). Tujuannya menemukan fase mana yang melambat, bukan hanya EOD secara keseluruhan.
2. Untuk fase yang melambat, ambil data SMF 30 per job: CPU, elapsed, dan jumlah EXCP. Jika CPU tetap tetapi elapsed naik, job menunggu sesuatu, bukan bekerja lebih banyak.
3. Lihat delay job-job itu di RMF. Jika delay device dominan, lanjut ke laporan DASD.
4. Bandingkan waktu respons per volume sebelum dan sesudah Jumat (Bab 6.3). Perhatikan komponen IOSQ.
5. Periksa volume baru: apakah HyperPAV aktif dan alias tersedia untuk control unit-nya, dan dataset apa saja yang kini dialokasikan ke volume itu.

**Dugaan akar masalah.** Volume baru ditambahkan ke storage group tanpa alias HyperPAV yang cukup, atau di control unit yang belum dikonfigurasi alias. Karena volume baru paling kosong, ACS dan SMS mengarahkan

banyak dataset baru dan dataset kerja EOD ke sana. Hasilnya, I/O menumpuk di satu volume dengan antrean di host. Dugaan ini didukung bila IOSQ volume baru tinggi sementara volume lain normal. Ini pola yang sama dengan kasus K-01.

**Tindakan jangka pendek.** Aktifkan atau tambah alias untuk volume baru. Bila tidak bisa segera, tandai volume baru agar tidak menerima alokasi baru sampai konfigurasinya selesai.

**Tindakan jangka panjang.** Tambahkan pemeriksaan alias dan konfigurasi PAV ke checklist penambahan volume. Tambahkan alert waktu respons per volume dibanding baseline, agar perlambatan seperti ini terdeteksi di hari pertama, bukan di hari keempat.

**Bukti penyelesaian.** IOSQ volume baru kembali setara volume lain. Durasi fase yang melambat kembali ke rentang normalnya selama tiga malam berturut-turut.

**Komunikasi.** “EOD melambat karena satu volume disk baru belum dikonfigurasi dengan benar dan menjadi antrean. Konfigurasinya sudah diperbaiki dan EOD kembali normal sejak malam ini. Pemeriksaan tambahan sudah dimasukkan ke prosedur penambahan disk.”

*Catatan penilai.* Jawaban cukup langsung menyimpulkan “disk lambat”. Jawaban baik membuktikan dari data bahwa CPU tidak berubah, menunjuk fase dan volume yang tepat, dan mengaitkan satu-satunya perubahan yang tercatat dengan gejala melalui bukti, bukan dugaan.

## P.2 Studi 2 — Mitra Melaporkan Transfer Ganda

**Langkah investigasi.**

1. Minta daftar referensi transaksi dari payment hub untuk sekitar 300 transfer yang dilaporkan ganda.
2. Cocokkan dengan catatan core banking. Untuk setiap referensi, berapa kali dibukukan, kapan, dan lewat koneksi mana.
3. Periksa log gateway dan z/OS Connect di sekitar pukul 14.10–14.20: timeout, retry, dan kode respons.

4. Periksa apakah ada takeover DVIPA atau perpindahan koneksi selama gangguan (Bab 37.1).
5. Periksa apakah permintaan membawa kunci idempotensi, dan apakah kunci itu dibuat oleh klien atau oleh server (Bab 102.3).

**Dugaan akar masalah.** Saat jaringan terganggu, sebagian permintaan sampai ke core dan diproses, tetapi responsnya tidak sampai ke payment hub. Payment hub menganggap permintaan gagal karena timeout dan mengirim ulang. Tanpa kunci idempotensi yang dibuat klien dan diperiksa di core, permintaan ulang diproses sebagai transaksi baru.

**Tindakan jangka pendek.** Siapkan jurnal pembalikan untuk transaksi ganda yang sudah terbukti, dengan persetujuan fungsi yang berwenang dan komunikasi kepada nasabah terdampak. Jangan membalik apa pun sebelum setiap pasangan terbukti ganda dari data kedua sisi.

**Tindakan jangka panjang.** Terapkan kunci idempotensi dari klien dengan index unik di core (Bab 102.3). Tambahkan API inquiry status, agar klien dapat menanyakan hasil transaksi setelah timeout sebelum mencoba ulang. Susun ulang rantai timeout (Bab 54.1, kasus K-52).

**Bukti penyelesaian.** Rekonsiliasi referensi transaksi antara payment hub dan core menunjukkan tidak ada selisih. Uji terkendali dengan memutus koneksi di tengah transaksi tidak lagi menghasilkan pembukuan ganda.

**Komunikasi.** “Gangguan jaringan singkat menyebabkan sebagian transfer dikirim ulang oleh sistem mitra dan tercatat dua kali. Semua transaksi ganda sudah diidentifikasi dan dibalik. Perlindungan terhadap pengiriman ulang sedang dipasang agar kejadian ini tidak berulang.”

*Catatan penilai.* Jawaban baik membedakan penyebab langsung (retry setelah timeout) dari akar masalah desain (tidak ada idempotensi). Jawaban baik juga menegaskan bahwa pembalikan hanya dilakukan setelah bukti per transaksi.

## P.3 Studi 3 — Uji DR: Aplikasi Tidak Bisa Membuka File

**Kemungkinan penyebab dan urutan pemeriksaan.** Mulailah dari pesan kesalahan yang paling spesifik, lalu periksa dari lapisan bawah ke atas.

| Urutan | Pemeriksaan | Penyebab yang diuji |
|---|---|---|
| 1 | Pesan kesalahan buka file di log CICS dan SYSLOG | Menentukan lapisan: katalog, volume, keamanan, atau enkripsi |
| 2 | Status volume tempat file berada | Volume offline karena IODF DR tertinggal (kasus K-38) |
| 3 | Status replikasi volume itu | Volume tidak masuk grup replikasi, sehingga datanya tidak ada atau usang |
| 4 | `LISTCAT` untuk ketiga file di situs DR | Katalog di DR tidak memuat file itu atau menunjuk volume yang salah |
| 5 | Kunci enkripsi dataset di ICSF situs DR | Label kunci tidak tersedia atau master key berbeda (kasus K-37) |
| 6 | Pesan penolakan RACF | Profil atau user teknis CICS berbeda di DR |
| 7 | Definisi file di CSD situs DR | Nama dataset atau atribut berbeda dari produksi (Bab 64.2) |

**Dugaan akar masalah.** Karena hanya tiga file dari satu aplikasi yang gagal, sementara aplikasi lain berjalan, penyebab yang paling mungkin adalah sesuatu yang spesifik untuk file-file itu. Kandidatnya adalah volume baru yang belum masuk replikasi atau IODF DR, atau file yang baru dienkripsi dengan label kunci yang belum ada di situs DR. Pemeriksaan 1 biasanya langsung mempersempit pilihan.

**Tindakan.** Perbaiki penyebab spesifik yang ditemukan, lalu ulangi uji untuk aplikasi itu. Untuk jangka panjang, tambahkan perbandingan konfigurasi yang terkait ke deteksi drift harian (Bab 15.1), dan tambahkan pembukaan ketiga file itu ke transaksi uji DR standar.

**Bukti penyelesaian.** Transaksi uji aplikasi berhasil di situs DR, total kontrol file cocok dengan produksi pada titik replikasi yang sama, dan cakupan uji DR diperbarui (Bab 121.1).

*Catatan penilai.* Jawaban cukup mendaftar penyebab tanpa urutan. Jawaban baik memakai fakta bahwa hanya tiga file yang gagal untuk mempersempit dugaan, dan menutup celahnya dengan kontrol yang mencegah kejadian serupa di uji berikutnya.

## P.4 Studi 4 — Lima Belas User dengan Atribut SPECIAL

**Prinsip.** Tujuannya bukan mencabut SPECIAL secepat mungkin, tetapi memastikan setiap pemakaian hak tertinggi punya alasan, tercatat, dan dibatasi. Mencabut hak dari orang yang ternyata membutuhkannya di tengah insiden malah menciptakan risiko baru.

| Pekan | Kegiatan | Hasil |
|---|---|---|
| 1–2 | Inventaris lengkap dari unload database RACF: siapa saja pemilik SPECIAL, OPERATIONS, dan AUDITOR, sejak kapan, dan untuk peran apa | Daftar 15 user dengan pemilik dan alasan tertulis |
| 2–6 | Aktifkan pencatatan pemakaian hak istimewa bila belum aktif. Kumpulkan data SMF 80 selama empat pekan: perintah apa yang benar-benar dijalankan dengan SPECIAL | Profil pemakaian nyata per user |
| 5–8 | Kelompokkan: (a) tidak pernah dipakai, (b) dipakai untuk tugas yang bisa diganti kewenangan terbatas, (c) benar-benar butuh | Rencana per user |
| 7–10 | Kelompok (a): cabut SPECIAL. Kelompok (b): ganti dengan kewenangan tingkat group atau kewenangan terbatas lain yang cukup untuk tugasnya | Jumlah user SPECIAL turun |
| 9–12 | Kelompok (c): pindahkan ke akun darurat (firecall) dengan prosedur peminjaman, | Sisa 2–3 akun darurat, semua pemakaian tercatat dan dialert |

| Pekan | Kegiatan | Hasil |
|---|---|---|
| | persetujuan, dan batas waktu. Tambahkan alert SIEM untuk setiap pemakaian (Bab 184.1) | |
| 13 | Laporan kepada auditor dan manajemen, dengan bukti sebelum dan sesudah | Temuan audit ditutup dengan bukti |

**Risiko yang dikelola.** Setiap pencabutan dilakukan setelah pemiliknya diberi tahu, dengan jalan kembali yang cepat bila ternyata ada tugas yang terlewat. Uji prosedur akun darurat dengan simulasi sebelum pencabutan terakhir, agar tim tidak terkunci saat insiden.

**Bukti penyelesaian.** Jumlah user SPECIAL turun ke angka yang disepakati. Setiap akun darurat punya prosedur tertulis, dan setiap pemakaiannya dalam 30 hari terakhir tercatat dan cocok dengan tiket.

*Catatan penilai.* Jawaban cukup langsung mencabut semua SPECIAL. Jawaban baik memakai data pemakaian nyata untuk memutuskan, menyediakan alternatif untuk kebutuhan sah, dan menjaga kemampuan tim menangani darurat.

## P.5 Studi 5 — Upgrade z/OS di Sysplex Dua Sistem

**Persiapan (pekan 1–6).**

1. Baca dokumentasi migrasi rilis tujuan. Daftar semua migration action yang berlaku, dan tetapkan pemilik untuk setiap action (Bab 128.2).
2. Pasang PTF koeksistensi dan toleransi di kedua sistem. IPL bila diperlukan, sebelum sistem pertama di-upgrade.
3. Bangun SYSRES dan target zone baru dengan SMP/E. Uji IPL di sistem sandbox dengan parmlib yang setara produksi.
4. Hitung kapasitas: selama satu sistem di-IPL, sistem lain menanggung seluruh beban. Pastikan utilisasinya tetap di bawah batas aman pada jam yang dipilih (Bab 11.1). Jika tidak, pilih jendela dengan beban lebih rendah.

5. Siapkan dan gladikan rencana fallback: IPL kembali dari SYSRES lama. Catat durasinya (Bab 131.1).

**Pelaksanaan (pekan 7–10).**

| Tahap | Kriteria lanjut ke tahap berikutnya |
|---|---|
| Upgrade SYSB (beban lebih ringan) di jendela terencana | SYSB bergabung ke sysplex, semua subsistem aktif, transaksi uji berhasil |
| Pantau SYSB minimal satu siklus EOD penuh, termasuk satu akhir pekan | Tidak ada abend baru; kinerja setara sebelum upgrade; health check bersih |
| Upgrade SYSA di jendela berikutnya | Kriteria yang sama seperti SYSB |
| Masa stabil dua sampai empat pekan dengan kedua sistem di rilis baru | Tidak ada masalah terbuka yang terkait upgrade |

**Setelah stabil (pekan 11 ke atas).** Aktifkan fungsi baru satu per satu, masing-masing dengan tiket perubahan tersendiri. Fallback ke rilis lama hanya dijamin selama fungsi baru belum dipakai. Karena itu langkah ini dilakukan paling akhir.

**Kriteria go/no-go setiap tahap** ditulis sebelumnya, lengkap dengan batas waktu keputusan dan orang yang berwenang memutuskan (Bab 130.1).

*Catatan penilai.* Jawaban cukup menyebut urutan sistem. Jawaban baik menghitung kapasitas saat satu sistem tidak tersedia, menggladi fallback, dan menunda pemakaian fungsi baru sampai kedua sistem stabil, karena langkah itulah yang menutup jalan fallback.

## Lampiran Q. Studi Kasus Berseri: Setahun di Bank Nusantara

Bank Nusantara adalah bank fiktif berukuran menengah: sysplex dua sistem di situs utama, replikasi asinkron ke situs DR, sekitar empat juta rekening, dan tim mainframe sembilan orang. Lampiran ini mengikuti tim itu selama setahun menjalankan program perbaikan dari Lampiran O. Semua nama dan angka fiktif, tetapi setiap episode disusun dari pola yang nyata.

## Q.1 Bulan 1: Titik Awal

Rina baru tiga bulan menjadi kepala tim mainframe. Dalam pertemuan empat mata dengan setiap anggota, keluhan yang paling sering adalah beban jaga: sebelas panggilan per pekan, sebagian besar tengah malam. Pareto insiden enam bulan terakhir menunjukkan 18 insiden per bulan, dan lebih dari separuhnya berasal dari empat jenis masalah yang terus berulang.

Peta kompetensi memperlihatkan risiko yang lebih besar. Area sysplex dan CF hanya dikuasai Pak Hendra, yang akan pensiun dalam delapan belas bulan. Area DR hanya dikuasai satu orang lain. Uji DR terakhir berlangsung empat belas bulan lalu, dan laporannya hanya menyebut “berhasil” tanpa angka.

Rina memilih tiga metrik untuk dipantau setiap bulan: jumlah insiden, panggilan jaga per pekan, dan MTTR. Ia menambahkan satu angka untuk manajemen: cakupan uji DR, yang diperkirakan sekitar 40%.

*Pelajaran:* sebelum memperbaiki apa pun, tim harus sepakat tentang angka titik awalnya.

## Q.2 Bulan 2: Tiga Puluh Tujuh Perbedaan

Tim menjalankan perbandingan konfigurasi pertama antara situs utama dan situs DR (Bab 15.1). Hasilnya tiga puluh tujuh perbedaan. Sebagian besar wajar, seperti nama sistem dan alamat IP. Namun ada empat yang serius: satu control unit storage belum ada di IODF situs DR, dua aturan TCP/IP yang hanya ada di produksi, dan definisi satu file CICS yang berbeda.

Setiap perbedaan diberi pemilik. Tiga pekan kemudian, keempat perbedaan serius diperbaiki. Perbandingan dijadwalkan setiap malam, dan perbedaan baru yang tidak ada di daftar pengecualian langsung menjadi tiket.

*Pelajaran:* drift tidak terasa sampai hari DR dibutuhkan. Perbandingan harian mengubah kejutan menjadi pekerjaan rutin.

## Q.3 Bulan 3: Backup yang Tidak Bisa Dipulihkan

Praktikum pemulihan mini (Praktikum 9) dijalankan untuk satu aplikasi pinjaman. Backup berhasil dipulihkan, tetapi total kontrolnya tidak cocok. Setelah diselidiki, satu dataset aplikasi itu tidak masuk ke daftar backup harian sejak dipindah ke HLQ baru setahun sebelumnya.

Tim lalu memeriksa semua aplikasi kritikal. Dua dataset lagi ditemukan di luar cakupan backup. Aturan backup diubah agar berbasis storage group dan HLQ, bukan daftar dataset satu per satu, dan uji restore dengan total kontrol dijadwalkan setiap kuartal.

*Pelajaran:* laporan "backup sukses" hanya membuktikan bahwa yang dicadangkan berhasil dicadangkan. Laporan itu tidak membuktikan bahwa semua yang penting ikut dicadangkan.

## Q.4 Bulan 4: Jalur Kritis EOD

Simulasi Monte Carlo dengan data tabel kontrol tiga bulan (Bab G.3) menunjukkan peluang EOD akhir bulan melewati batas sekitar 14%. Analisis jalur kritis menemukan fase aging kredit dan satu job laporan internal yang tidak perlu berada di jalur kritis tetapi menunggu dependensi yang sudah tidak relevan.

Dependensi job laporan diperbaiki, sehingga job itu berjalan paralel. Fase aging kredit dioptimalkan dengan index tambahan setelah analisis `EXPLAIN`. Simulasi diulang, dan peluang terlambat turun menjadi sekitar 5%.

*Pelajaran:* mempercepat job di luar jalur kritis tidak mengubah apa pun. Temukan jalur kritisnya dulu.

## Q.5 Bulan 5: Membersihkan Alert

Tim meninjau semua alert tiga bulan terakhir. Dari rata-rata sebelas panggilan jaga per pekan, sekitar enam ternyata tidak membutuhkan tindakan apa pun. Contohnya peringatan spool di atas 70% yang selalu turun sendiri, atau job housekeeping yang terlambat beberapa menit.

Alert tanpa tindakan dihapus atau diturunkan menjadi laporan pagi. Tiga alert yang sering dijawab dengan langkah yang sama diotomasi dengan pagar pengaman (Bab 26.4). Presisi alert naik dari sekitar 45% menjadi di atas 80%, dan panggilan jaga turun menjadi sekitar enam per pekan.

*Pelajaran:* menurunkan panggilan jaga dari sebelas menjadi enam setara dengan hampir menggandakan ukuran rotasi, tanpa menambah satu orang pun (Bab 173.1).

## Q.6 Bulan 6: Kontrak Interface dengan Mitra

Analisis insiden setengah tahun pertama menunjukkan bahwa hampir sepertiga insiden berasal dari perbatasan dengan mitra: sertifikat yang diganti tanpa pemberitahuan, file yang terlambat atau terpotong, dan satu mitra yang tiba-tiba mengirim pesan dengan encoding berbeda.

Tim menyusun kontrak interface satu halaman untuk dua belas mitra terpenting. Setiap kontrak mencantumkan format, CCSID, urutan data, jadwal, perilaku koneksi, sertifikat beserta tanggal kedaluwarsanya, batas laju, dan kontak darurat. Penyusunannya sendiri sudah menemukan masalah: tiga mitra memakai sertifikat yang akan kedaluwarsa dalam empat bulan, dan satu mitra ternyata membuka koneksi TLS baru untuk setiap permintaan (Bab 148.1).

*Pelajaran:* menuliskan apa yang “sudah semua orang tahu” sering menemukan hal yang ternyata tidak diketahui siapa pun.

## Q.7 Bulan 7: Uji DR yang Jujur

Uji DR tahun ini dirancang berbeda. Targetnya bukan “sistem menyala”, tetapi transaksi uji dari lima fungsi bisnis kritikal berhasil dan total kontrol cocok. Semua langkah dicatat waktunya (Bab 116.1).

Hasilnya: tiga dari lima fungsi berhasil, dan RTO nyata 150 menit dibanding target 120 menit. Satu fungsi gagal karena file terenkripsi tidak dapat dibuka. Label kuncinya belum dibuat di situs DR, pola yang sama dengan kasus K-37. Fungsi lain gagal karena satu definisi CICS yang belum masuk

perbandingan drift. Kedua masalah diperbaiki dalam dua pekan, dan cakupan uji DR naik dari 40% ke sekitar 70%.

*Pelajaran:* uji DR yang menemukan masalah adalah uji yang berhasil. Uji yang selalu "berhasil" tanpa angka patut dicurigai.

## Q.8 Bulan 8: Tagihan yang Turun

Laporan SCRT menunjukkan puncak 4HRA selalu terjadi pukul 10.00–14.00 pada hari kerja pertama setiap bulan. Penyebabnya gabungan transaksi gajian, laporan internal yang dijadwalkan pagi, dan beban DDF dari aplikasi analitik.

Tiga langkah diambil. Laporan internal dipindah ke malam hari. Beban DDF dan Java yang eligible diarahkan ke zIIP (Bab 42.2). Kueri analitik diberi batas ASUTIME dan resource group (Bab 155.1). Puncak 4HRA turun dari sekitar 1.000 menjadi sekitar 910 MSU. Tim pengadaan menghitung dampaknya dengan tarif kontrak (Bab 177.1), dan penghematannya cukup untuk membiayai pelatihan dua anggota tim.

*Pelajaran:* keputusan teknis kecil dapat mengubah tagihan besar, dan penghematan yang terukur mudah diubah menjadi investasi untuk tim.

## Q.9 Bulan 9: Insiden yang Tidak Terjadi

Inventaris hal yang kedaluwarsa (Bab 188.1) menandai sertifikat koneksi ke penyelenggara sistem pembayaran yang akan habis dalam 62 hari. Prosesnya membutuhkan koordinasi dengan pihak luar sekitar 45 hari (Bab 149.2). Pembaruan dimulai hari itu juga dan selesai 10 hari sebelum tenggat. Tahun sebelumnya, sertifikat yang sama diperbarui pada hari terakhir, dengan layanan transfer yang sempat terganggu.

Bulan yang sama, Dimas mulai bekerja berpasangan dengan Pak Hendra untuk semua pekerjaan sysplex dan CF. Dalam latihan tabletop kegagalan CF, Dimas memimpin dan Pak Hendra hanya mengamati. Faktor bus area sysplex naik dari satu menjadi dua.

*Pelajaran:* keberhasilan program perbaikan sering berupa hal yang tidak terjadi. Catat juga near miss seperti ini, karena tanpa catatan keberhasilannya tidak terlihat.

## Q.10 Bulan 10–12: Ritme yang Bertahan

Uji DR kedua di bulan kesebelas mencakup semua lima fungsi kritikal. RTO nyata 115 menit, di bawah target, berkat validasi total kontrol yang diotomasi. Cakupan uji DR mencapai sekitar 85%. Fase aging kredit yang sudah dioptimalkan dan perpanjangan window akhir tahun yang disepakati dengan bisnis menurunkan peluang EOD akhir bulan terlambat menjadi sekitar 3%.

Change freeze akhir tahun dijalankan dengan kategori pengecualian yang jelas (Bab 126.1). Desember berlalu tanpa insiden tingkat tinggi, untuk pertama kalinya dalam tiga tahun.


*Gambar Q.1 — Satu tahun program perbaikan: sebelum dan sesudah*

Dalam tinjauan akhir tahun, Rina menyajikan angka-angka dalam Gambar Q.1 kepada manajemen. Namun yang paling ia tekankan bukan angkanya, melainkan ritmenya. Setiap bulan, tim meninjau metrik yang sama, memilih perbaikan berikutnya, dan menutup yang sudah selesai. Tidak ada

proyek besar. Yang ada adalah puluhan perbaikan kecil yang masing-masing terukur.

**Rencana tahun kedua** disusun dengan cara yang sama: lima kontrol berikutnya dari penilaian kematangan (Bab G.6), dua anggota baru masuk rotasi jaga lewat rencana 90 hari (Bab O.1), dan target menaikkan faktor bus semua area kritikal menjadi minimal dua sebelum Pak Hendra pensiun.

### Q.11 Refleksi

Perjalanan Bank Nusantara menunjukkan pola yang sama dengan buku ini secara keseluruhan: ukur, bandingkan dengan batas, putuskan, lalu ukur lagi. Sebagian besar perbaikan dalam setahun itu tidak membutuhkan teknologi baru. Yang dibutuhkan adalah data yang sudah ada, dibaca dengan pertanyaan yang tepat, dan ditindaklanjuti dengan disiplin.

Pertanyaan untuk pembaca: dari sepuluh episode di atas, episode mana yang paling mirip dengan kondisi tim Anda saat ini? Mulailah dari episode itu. Dalam sebulan, Anda sudah punya episode pertama untuk cerita tim Anda sendiri.

## Lampiran R. Catatan Implementasi per Komponen

Lampiran ini merangkum praktik di seluruh buku menjadi checklist per komponen. Setiap komponen punya dua daftar: konfigurasi awal yang sebaiknya sudah benar sejak komponen dipasang atau diwarisi, dan pemeriksaan tahunan untuk memastikan konfigurasi itu tetap benar. Checklist ini juga dapat dipakai sebagai bahan audit internal. Setiap baris yang tidak bisa dijawab “ya” dengan bukti adalah temuan.

### R.1 z/OS dan Parmlib

**Konfigurasi awal.**

| Butir | Mengapa | Bab |
|---|---|---|
| Semua member parmlib, | Jalan kembali dan deteksi | 15.1 |

| Butir | Mengapa | Bab |
|---|---|---|
| proclib, dan konfigurasi sistem disimpan di Git dengan riwayat perubahan | drift | |
| Konfigurasi IPL alternatif (LOADxx atau suffix IEASYSxx cadangan) siap dan terdokumentasi | Pemulihan bila perubahan parmlib gagal | 12.5 |
| Perubahan dinamis (APF, LNKLST, parameter lain) selalu dicatat juga ke member permanen | Mencegah perubahan hilang setelah IPL | 38.1, K-49 |
| Health Checker aktif, dengan exception yang ditinjau | Deteksi konfigurasi berisiko | 15.1, 181.1 |
| Perekaman SMF ke logstream dengan jadwal pembongkaran harian | Data kinerja, keamanan, dan penagihan tidak hilang | 170.1 |
| LOGREC ke logstream, dengan retensi yang cukup untuk investigasi | Bahan diagnosis kegagalan | 51.1 |
| Dataset dump dengan alokasi otomatis dan ruang yang cukup | Dump utuh saat dibutuhkan | 51.1 |
| Sinkronisasi waktu lewat STP, dan sistem terbuka mengacu ke sumber waktu yang sama | Linimasa insiden yang akurat | 112.1, K-57 |

## Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Daftar APF dan LNKLST ditinjau: tidak ada dataset yang tidak ada, pemilik jelas | Laporan review APF (Bab 29.1) |
| Konfigurasi IPL alternatif diuji di sistem uji | Catatan uji dengan tanggal |
| Level maintenance sesuai kebijakan (RSU, HIPER, PE) | Laporan SMP/E (Bab 14.4) |
| Parmlib produksi dan DR cocok, kecuali perbedaan yang tercatat | Laporan perbandingan drift |

| Butir | Bukti yang diharapkan |
|---|---|
| Exception Health Checker yang terbuka punya pemilik dan tenggat | Daftar exception |

Kesalahan yang paling sering di area ini adalah perubahan dinamis yang tidak dicatat ke member permanen. Perubahan itu berjalan baik sampai IPL berikutnya, berbulan-bulan kemudian, saat tidak ada lagi yang ingat mengapa sistem berperilaku berbeda.

## R.2 JES2

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Checkpoint dengan dua dataset di volume terpisah | Checkpoint tetap tersedia bila satu volume bermasalah | 21 |
| Spool di beberapa volume, dengan ambang peringatan yang sesuai laju isi | Mencegah spool penuh saat EOD | 21, 59.1 |
| Kebijakan retensi output yang otomatis, per kelas output | Spool tidak dipenuhi output lama | 21 |
| Kelas job dan initiator dipisah untuk batch kritis, batch biasa, dan pekerjaan ad hoc | Pekerjaan ad hoc tidak menahan EOD | 21.3 |
| Batas jumlah job dan output sesuai volume puncak | Tidak kehabisan nomor job di akhir bulan | 21 |
| Definisi JES2 disimpan di Git dan dibandingkan dengan DR | Konsistensi saat pemulihan | 15.1 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Pemakaian spool puncak setahun terakhir dibanding kapasitas | Grafik tren dan proyeksi |
| Kelas initiator masih sesuai pola beban | Analisis antrean job (Bab 21.3) |
| Prosedur pemulihan checkpoint dan spool | Catatan latihan |

| Butir | Bukti yang diharapkan |
|---|---|
| pernah digladikan | |
| Retensi output sesuai kebijakan data | Konfigurasi retensi dan laporan pembersihan |

Spool penuh adalah salah satu kegagalan yang paling cepat menjalar: job tidak bisa menulis output, subsistem mulai tertahan, dan EOD berhenti. Ambang peringatan harus cukup rendah untuk memberi waktu bertindak di puncak EOD, bukan disetel dari rata-rata harian.

## R.3 SMS dan Storage

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Rutin ACS disimpan di Git dan diuji dengan kasus uji sebelum diaktifkan | Kesalahan ACS berdampak ke semua alokasi baru | 18 |
| Storage group dipisah per jenis data: produksi, kerja sementara, backup, dan data yang direplikasi | Kapasitas dan replikasi dapat dikelola terpisah | 18, G.7 |
| Ambang storage group dengan alert dan proyeksi | Mencegah abend ruang | 59.1, 89.1 |
| Volume cadangan siap ditambahkan ke storage group kritikal | Respons cepat saat kapasitas menipis | 89.1 |
| Management class sesuai kebijakan retensi dan backup | Data disimpan dan dibuang sesuai aturan | 45.1 |
| Semua volume baru dikonfigurasi dengan PAV atau HyperPAV sebelum menerima alokasi | Mencegah antrean I/O | 6.3, P.1 |
| Data sementara EOD ditempatkan di storage group yang tidak direplikasi bila tidak dibutuhkan di DR | Mengurangi beban replikasi | G.7 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Cakupan backup mencakup semua dataset kritikal, termasuk HLQ baru | Perbandingan daftar dataset kritikal dengan cakupan backup (Q.3) |
| Uji restore dengan total kontrol untuk aplikasi perwakilan | Catatan uji (Praktikum 2, 9) |
| Tren pertumbuhan per storage group dan proyeksi 12 bulan | Laporan kapasitas (Templat J.3) |
| Konfigurasi SMS produksi dan DR cocok | Laporan drift |

Kesalahan paling mahal di area storage biasanya bukan kehabisan ruang, tetapi data penting yang ternyata tidak ikut dicadangkan atau direplikasi. Cakupan backup dan replikasi harus diperiksa dari daftar dataset kritikal, bukan dari laporan backup yang sukses.

## R.4 RACF

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Perlindungan default untuk dataset yang tidak punya profil | Tidak ada dataset yang terbuka tanpa sengaja | 27 |
| Profil generik aktif untuk dataset dan class penting | Pengelolaan yang konsisten | 27.3 |
| Aturan kata sandi atau frasa sandi sesuai kebijakan, dan revoke otomatis untuk akun tidak aktif | Mengurangi akun yang dapat disalahgunakan | 28.2 |
| Atribut SPECIAL dan OPERATIONS hanya untuk akun darurat dengan prosedur | Hak tertinggi terkendali dan tercatat | P.4 |
| Pencatatan pemakaian hak istimewa dan perubahan profil aktif | Bahan audit dan deteksi | 30.1 |
| Database RACF memiliki salinan cadangan yang aktif | Keamanan tetap berjalan bila database utama bermasalah | 27 |

| Butir | Mengapa | Bab |
|---|---|---|
| dan backup terjadwal | | |
| Proses masuk, pindah, dan keluar pegawai terhubung ke administrasi RACF | Akses mengikuti peran saat ini | 28.2, K-32 |
| Event keamanan dikirim ke SIEM dengan aturan korelasi | Deteksi pola serangan | 184.1 |

**Pemeriksaan tahunan.**

| Butir | Bukti yang diharapkan |
|---|---|
| Recertification akses untuk semua user dengan hak ke data produksi | Tanda tangan pemilik data |
| Jumlah akun berprivilegi dan pemakaiannya | Laporan dari data SMF 80 |
| Profil dalam mode WARNING tidak ada, atau punya alasan dan tenggat | Laporan profil WARNING (Bab 137.1) |
| UACC di atas NONE pada data produksi punya alasan tertulis | Laporan profil |
| Aturan korelasi SIEM diuji dengan simulasi | Catatan uji |

RACF yang dirancang dengan baik tetap bisa melemah perlahan lewat pengecualian sementara yang tidak pernah dicabut. Pemeriksaan tahunan paling berguna bila fokus pada pengecualian: mode WARNING, UACC terbuka, dan akses yang diberikan untuk proyek yang sudah selesai.

## R.5 Db2 for z/OS

**Konfigurasi awal.**

| Butir | Mengapa | Bab |
|---|---|---|
| Dual logging untuk log aktif dan archive, serta dual BSDS | Satu salinan rusak tidak menghentikan Db2 | 70, G.2 |
| Kapasitas log aktif cukup untuk beberapa jam laju tulis puncak | Waktu menangani masalah offload | G.2 |
| Alert jumlah log aktif yang | Deteksi dini sebelum log | G.2 |

| Butir | Mengapa | Bab |
|---|---|---|
| belum di-offload | penuh | |
| Batas thread (`CTHREAD`, `MAXDBAT`, `CONDBAT`) dihitung dari beban nyata | Tidak ada antrean yang tidak perlu | 68.2 |
| RLF aktif dengan batas ASUTIME untuk kueri dinamis | Perlindungan dari kueri liar | 155.1 |
| Pembagian buffer pool sesuai pola akses; `PGFIX` untuk pool yang sibuk | Kinerja stabil | 69.1 |
| Kebijakan image copy per kelompok objek, dengan FlashCopy untuk objek besar | Waktu RECOVER sesuai RTO | 72.5 |
| Pemantauan RTS untuk kebutuhan REORG, dengan jadwal di luar window EOD | Kinerja terjaga tanpa mengganggu EOD | 156.1 |
| Perlindungan access path sebelum rebind massal | Mencegah regresi kinerja | 72, K-11 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Uji RECOVER tablespace perwakilan, dengan waktu dibanding RTO | Catatan uji |
| Laju tulis log puncak dibanding kapasitas log aktif | Statistik Db2 akhir bulan |
| Batas thread dibanding pemakaian puncak setahun | Statistik DDF |
| Distribusi ASUTIME dan kejadian `-905` | Laporan RLF (Bab 155.1) |
| Objek dengan partisi atau REORG yang semakin lama | Riwayat utilitas |

Db2 yang berjalan stabil bertahun-tahun sering menyimpan risiko yang tidak terlihat: log aktif yang dulu cukup kini sempit karena volume tumbuh, atau image copy yang jadwalnya tidak diperbarui untuk tabel baru. Pemeriksaan tahunan menyamakan konfigurasi dengan beban hari ini.

## R.6 CICS

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Topologi region memisahkan aplikasi berisiko dari transaksi inti | Masalah satu aplikasi tidak menjatuhkan layanan lain | 63.3 |
| Minimal dua AOR untuk setiap layanan kritikal, dengan workload routing | Restart bergilir tanpa gangguan | 63.3 |
| Batas storage dan MXT disetel dari pemakaian nyata | Mencegah SOS dan antrean | 65, 67.3 |
| Keamanan transaksi dan resource aktif | Akses sesuai peran | 32 |
| Definisi CSD disimpan di Git dan dipromosikan lewat pipeline | Konsistensi antar-lingkungan | 64.2 |
| Data monitoring kinerja transaksi (SMF 110) aktif | Dekomposisi waktu respons | G.1 |
| Program yang banyak memanggil Db2 diperiksa untuk threadsafe | Mengurangi perpindahan TCB | 65.4 |
| Kebijakan dump transaksi dan sistem yang jelas | Diagnosis tanpa membanjiri ruang dump | 51.1, 67 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Tren pemakaian EDSA dan puncak task per region | Laporan statistik bulanan (Bab 67.3) |
| CSD produksi, UAT, dan DR cocok | Laporan perbandingan |
| Pemetaan aplikasi ke region masih sesuai topologi yang direncanakan | Diagram topologi terkini |
| Program baru yang ditandai threadsafe sudah melewati tinjauan kode | Catatan tinjauan |

Masalah CICS yang paling sering muncul setelah rilis aplikasi baru adalah kebocoran storage dan kesalahan threadsafe (kasus K-23 dan K-24). Aplikasi baru sebaiknya berjalan di region terpisah pada bulan-bulan pertama, lalu dipindahkan setelah terbukti stabil.

## R.7 MQ

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Dual logging dan ukuran log sesuai volume pesan persisten | Pemulihan pesan yang andal | 164.1 |
| Page set atau structure cukup untuk `MAXDEPTH` antrean kritikal | Antrean tidak penuh sebelum batasnya | 34.1, 164.1 |
| Alert kedalaman antrean di beberapa ambang | Waktu bertindak sebelum penuh | 34.1 |
| Channel authentication aktif, dengan aturan tolak umum | Koneksi tidak dikenal ditolak | 166.1 |
| `MCAUSER` terbatas untuk setiap channel mitra | Pesan mitra tidak masuk dengan hak berlebih | 166.1 |
| TLS di semua channel keluar LPAR | Pesan pembayaran terlindungi | 166.1 |
| Dead-letter queue terdefinisi dan dipantau dengan alert | Pesan gagal terlihat cepat | K-26 |
| Parameter channel (`BATCHSZ`, `HBINT`, `DISCINT`) disetel per jenis aliran | Throughput dan deteksi koneksi mati | 165.1 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Audit keamanan MQ lengkap | Checklist Bab 166.1 dengan hasilnya |
| Sertifikat channel dan tanggal kedaluwarsanya | Inventaris sertifikat (Bab 149.2) |

| Butir | Bukti yang diharapkan |
|---|---|
| Kedalaman antrean puncak setahun dibanding MAXDEPTH | Statistik antrean |
| Throughput channel jarak jauh dibanding batas teoretis | Perhitungan Bab 165.1 |

Perubahan di sisi mitra adalah sumber masalah MQ yang paling sering: sertifikat baru, encoding baru, atau perubahan jaringan. Setiap channel mitra sebaiknya punya kontrak interface tertulis yang mencakup hal-hal itu (Bab Q.6).

## R.8 TCP/IP dan AT-TLS

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Port penting direservasi untuk job atau started task yang berhak | Listener yang benar selalu mendapat port-nya | 38.1 |
| Penyaringan IP dengan aturan tolak kecuali diizinkan, setiap aturan punya pemilik | Permukaan serangan minimum | 150.1 |
| Dynamic VIPA untuk layanan yang harus berpindah saat satu sistem gagal | Kanal tetap tersedia | 37.1 |
| Kebijakan AT-TLS untuk semua port yang dipakai pihak luar | Enkripsi tanpa mengubah aplikasi | 38, 148.1 |
| Cache sesi TLS aktif, dan klien besar memakai koneksi persisten | Biaya CPU TLS terkendali | 148.1 |
| HiperSockets untuk lalu lintas antar-LPAR di CPC yang sama | Latensi rendah | 36.1, K-44 |
| Semua perubahan dinamis dicatat ke profil utama | Tidak hilang saat restart stack | 38.1 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Aturan penyaringan tanpa lalu lintas 90 hari, tanpa pemilik, atau terlalu luas | Hasil algoritma tinjauan (Bab 150.1) |
| Inventaris sertifikat server dan klien, dengan tanggal kedaluwarsa | Kalender kedaluwarsa (Bab 149.2) |
| Rasio handshake penuh terhadap transaksi per mitra | Statistik AT-TLS |
| Uji pengalihan DVIPA | Catatan uji |

## R.9 Parallel Sysplex dan Coupling Facility

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Minimal dua CF, sebaiknya dengan failure isolation dari anggota data sharing | Kegagalan satu CF atau satu mesin tidak menghapus semuanya | 179.1 |
| Policy CFRM dengan `PREFLIST` yang disengaja untuk setiap structure | Penempatan structure dapat diprediksi | 181.1 |
| Keputusan duplexing tertulis per structure | Keseimbangan biaya dan perlindungan | 179.1 |
| Minimal dua jalur sinyal per pasangan sistem lewat CF berbeda | Signaling tetap berjalan saat satu jalur hilang | 178.1 |
| Couple dataset primer dan alternate di volume dan jalur berbeda | Tidak ada titik kegagalan tunggal | 43 |
| Policy SFM aktif dengan isolasi otomatis | Sysplex tidak tertahan menunggu operator | 180.1 |
| Policy ARM untuk subsistem kritikal | Restart otomatis setelah kegagalan | 71.1 |
| Kapasitas N+1: utilisasi normal memungkinkan satu sistem menanggung beban | Selamat saat satu sistem berhenti | 11.1 |

### Pemeriksaan tahunan.

| Butir | Bukti yang diharapkan |
|---|---|
| Laporan kesehatan sysplex harian berjalan dan peringatannya ditindaklanjuti | Riwayat laporan (Bab 181.1) |
| Uji perilaku SFM di lingkungan uji | Catatan uji dengan waktu isolasi |
| Buffer transport class XCF cukup di jam puncak | Laporan XCF RMF |
| Waktu layanan CF per structure dibanding tahun lalu | Laporan CF RMF |
| Redundansi dibuktikan kembali setelah setiap pemeliharaan CF | Checklist pemeliharaan (K-35, K-36) |

## R.10 GDPS, Replikasi, dan DR

### Konfigurasi awal.

| Butir | Mengapa | Bab |
|---|---|---|
| Semua volume data produksi yang dibutuhkan di DR masuk grup replikasi dan grup konsistensi | Data di DR lengkap dan konsisten | 44, P.3 |
| Status replikasi dan kesiapan HyperSwap dipantau terus dengan alert | Mencegah HyperSwap gagal diam-diam | 44.1, K-15 |
| Bandwidth replikasi asinkron di atas puncak laju tulis, atau RPO puncak diterima tertulis | RPO yang dijanjikan realistis | G.7 |
| Snapshot tidak dapat diubah dengan retensi bertingkat | Perlindungan dari serangan siber | 111.1, 113.2 |
| IODF, CFRM, CSD, TCP/IP, dan kunci kriptografi situs DR dibandingkan harian | DR siap saat dibutuhkan | 15.1, K-37, K-38 |
| Kriteria deklarasi DR penuh tertulis dan disetujui | Keputusan cepat dan tepat | G.8 |
| Runbook switchover dengan titik keputusan dan jalan kembali | Switchover terkendali | 115.1 |

**Pemeriksaan tahunan.**

| Butir | Bukti yang diharapkan |
|---|---|
| Uji DR dengan transaksi uji semua fungsi kritikal dan pembacaan data terenkripsi | Checklist J.4 terisi, RTO dan RPO nyata |
| Cakupan uji DR dibanding target | Laporan cakupan (Bab 121.1) |
| Uji pemulihan dari snapshot terisolasi | Catatan uji dengan waktu validasi |
| Perbandingan laju tulis puncak dengan bandwidth replikasi | Laporan RMF DASD dan kapasitas jaringan |

## R.11 Memakai Checklist Ini

Checklist per komponen paling berguna bila dijalankan dalam siklus tahunan yang terjadwal, bukan sekaligus menjelang audit. Satu cara yang sederhana adalah membagi sepuluh komponen ke dalam kalender tahunan (Bab 122–127), misalnya satu atau dua komponen per bulan, sehingga pemeriksaan menyebar dan tidak membebani tim di satu waktu.

| Kuartal | Komponen yang diperiksa |
|---|---|
| Pertama | z/OS dan parmlib, RACF, TCP/IP dan AT-TLS |
| Kedua | Db2, CICS, MQ |
| Ketiga | Sysplex dan CF, GDPS dan DR (bersama uji DR tahunan) |
| Keempat | JES2, SMS dan storage, serta persiapan akhir tahun |

Simpan hasil setiap pemeriksaan bersama buktinya. Setelah dua atau tiga tahun, kumpulan hasil ini menjadi riwayat kondisi platform yang sangat berharga: bukti bagi auditor, bahan bagi anggota tim baru, dan dasar bagi keputusan investasi berikutnya.

Seperti rumus dan tabel lain di buku ini, checklist ini adalah titik awal. Tambahkan butir dari setiap insiden dan temuan audit di lingkungan Anda sendiri. Checklist yang terus bertambah dari pengalaman nyata akan jauh lebih berharga daripada checklist yang disalin dari buku mana pun, termasuk buku ini.

# Lampiran S. Soal Latihan per Bagian

Soal-soal berikut melengkapi bank soal di Lampiran H, dengan fokus pada subbab analisis, studi kasus K-21 sampai K-60, dan lampiran lanjutan. Setiap kelompok bisa dipakai sebagai kuis singkat setelah membaca bagian terkait. Kunci jawaban tersebar merata di keempat pilihan.

## S.1 Operasi dan Batch

1. Fase EOD yang abend tepat setelah menulis semua akrual tetapi sebelum menandai status selesai, lalu dijalankan ulang tanpa pemeriksaan, akan: (a) mencatat akrual dua kali (b) menghapus akrual (c) aman (d) melewati semua rekening
2. Interval checkpoint optimal bila checkpoint memakan C menit dan rata-rata gagal setiap M menit mendekati: (a) C + M (b) C × M (c) √(2 × C × M) (d) M / C
3. Salah satu syarat window EOD yang sehat adalah: (a) tidak ada fase yang gagal (b) semua job berjalan paralel (c) EOD selalu di bawah 50% window (d) EOD ditambah fase terpanjang masih muat di window
4. File interface tanpa trailer yang cocok sebaiknya: (a) diabaikan (b) ditolak seluruhnya (c) diproses sebagian (d) dikarantina per record
5. Cara paling aman mencegah `S0C7` dari data mitra: (a) mengompilasi dengan `OPT(2)` (b) menaikkan region (c) menambah checkpoint (d) memeriksa `IS NUMERIC` dan mengkarantina record yang gagal
6. Perubahan kalender karena cuti bersama sebaiknya diperlakukan sebagai: (a) tugas operator (b) perubahan aplikasi yang diuji dengan simulasi EOD (c) perubahan yang tidak perlu dicatat (d) perubahan data biasa

## S.2 Kinerja dan Kapasitas

7. Transaksi CICS 120 ms dengan 70 ms menunggu Db2 paling tepat diperbaiki dengan: (a) menambah AOR (b) menambah CPU (c) memeriksa SQL dan lock (d) menaikkan MXT

8. Batas thread DDF tercapai sementara CPU masih longgar paling mungkin disebabkan: (a) pengelolaan koneksi di klien (b) buffer pool kecil (c) log penuh (d) CPU lambat
9. Hit ratio buffer pool naik dari 98% ke 99% pada 200.000 getpage per detik memotong I/O sinkron dari sekitar 4.000 menjadi: (a) 2.000 (b) 400 (c) 1.000 (d) 3.000
10. Memori guest Linux yang terlalu besar di z/VM berdampak: (a) mempercepat semua guest (b) tidak berdampak (c) hanya ke guest itu (d) menaikkan paging z/VM untuk semua guest
11. Satu kueri analitik besar di jam sibuk dapat menaikkan tagihan software karena: (a) menambah ruang disk (b) mendorong puncak 4HRA (c) memperbesar log (d) menambah jumlah user
12. Dengan puncak 60% dan batas aman 85%, pertumbuhan 15% per tahun menghabiskan kapasitas dalam sekitar: (a) 12 bulan (b) 60 bulan (c) 30 bulan (d) 6 bulan

## S.3 Ketersediaan dan DR

13. Replikasi asinkron dengan bandwidth di bawah puncak laju tulis menyebabkan: (a) HyperSwap otomatis (b) RPO membesar selama puncak tulis (c) data ganda (d) RPO konstan
14. Uji DR yang menemukan tiga masalah sebaiknya dinilai: (a) tidak perlu dilaporkan (b) perlu diulang tanpa perubahan (c) berhasil menemukan celah yang harus diperbaiki (d) gagal total
15. Setelah HyperSwap berhasil karena satu array bermasalah, situs utama berjalan: (a) tanpa salinan sinkron sampai array diperbaiki (b) dengan perlindungan penuh (c) tanpa sysplex (d) di situs DR
16. Kegagalan membaca data terenkripsi di situs DR paling sering disebabkan: (a) spool penuh (b) WLM berbeda (c) CPU DR lebih kecil (d) kunci atau master key yang tidak sama
17. Jalur jaringan cadangan yang tidak pernah membawa lalu lintas sebaiknya: (a) dibiarkan (b) dimatikan (c) diuji dan dipantau aktif (d) dipakai untuk uji saja

18. Kriteria deklarasi DR penuh sebaiknya: (a) diserahkan ke vendor (b) ditulis dan disetujui sebelumnya (c) diputuskan saat kejadian (d) tidak perlu

## S.4 Keamanan

19. Mengurangi user SPECIAL dengan aman dimulai dari: (a) wawancara saja (b) mengganti kata sandi (c) mencabut semuanya (d) data pemakaian nyata dari SMF 80
20. Aturan korelasi “banyak penolakan akses lalu berhasil” mendeteksi: (a) percobaan mencari celah yang berhasil (b) spool penuh (c) kinerja lambat (d) job gagal
21. Mengaktifkan audit baca pada profil generik yang sangat luas berisiko: (a) menghapus profil (b) memperlambat login (c) melonjakkan volume SMF (d) menonaktifkan RACF
22. Skrip pembersihan yang berjalan dengan UID 0 berbahaya karena: (a) satu kesalahan kecil bisa menjadi kerusakan besar (b) tidak tercatat di SMF (c) lebih lambat (d) tidak bisa dijadwalkan
23. `MCAUSER` channel mitra sebaiknya: (a) user berprivilegi (b) user terbatas sesuai kebutuhan (c) kosong (d) sama dengan user administrator
24. Bukti yang paling kuat untuk rantai bukti insiden adalah: (a) ingatan petugas (b) email ringkasan (c) tangkapan layar (d) salinan log yang di-hash, dicatat siapa dan kapan

## S.5 Integrasi dan Mitra

25. API yang mengembalikan kode kesalahan server untuk penolakan bisnis dapat: (a) mengurangi beban (b) tidak berpengaruh (c) memicu retry berulang dan melipatgandakan beban (d) mempercepat klien
26. Dalam rantai timeout yang benar, setiap lapisan sebaiknya menyerah: (a) bersamaan (b) sedikit lebih cepat daripada lapisan di atasnya (c) lebih lambat daripada lapisan di atasnya (d) tanpa batas

27. Perubahan nama field di respons API yang dipakai mitra sebaiknya: (a) dirilis sebagai versi API baru dengan masa transisi (b) dirilis langsung (c) dilakukan diam-diam (d) hanya diumumkan
28. Halaman aplikasi cloud yang memanggil delapan API berurutan lintas jaringan sebaiknya diperbaiki dengan: (a) cache browser (b) menambah AOR (c) CPU mainframe lebih cepat (d) API agregat
29. Pesan yang ditolak karena stempel waktunya kedaluwarsa menunjuk ke masalah: (a) encoding (b) sinkronisasi waktu (c) sertifikat (d) kapasitas
30. Dashboard yang menampilkan data replikasi tertinggal sebaiknya: (a) tidak diubah (b) memakai data kemarin (c) disembunyikan (d) menampilkan waktu data terakhir diperbarui

## S.6 Tim dan Program Perbaikan

31. Area dengan faktor bus satu berarti: (a) hanya satu orang mampu bekerja mandiri di area itu (b) satu tim menangani area itu (c) satu insiden per bulan (d) satu runbook tersedia
32. Menurunkan panggilan jaga dari sebelas menjadi enam per pekan setara dengan: (a) tidak ada efek (b) mengurangi kapasitas (c) hampir menggandakan ukuran rotasi (d) menambah satu orang
33. Bulan pertama kepala tim baru sebaiknya dipakai untuk: (a) menulis ulang runbook (b) mengubah banyak proses (c) membeli alat baru (d) mendengar dan memetakan
34. Laporan “backup sukses” membuktikan bahwa: (a) DR siap (b) restore pasti berhasil (c) yang dicadangkan berhasil dicadangkan (d) semua data penting tercadang
35. Kontrol yang wajib di semua ukuran bank adalah: (a) backup terisolasi dengan uji restore (b) program kematangan kontrol (c) game day (d) SLO dan burn rate
36. Yang membuat program perbaikan bertahan adalah: (a) konsultan eksternal (b) ritme pengukuran dan tindak lanjut yang rutin (c) alat baru (d) proyek besar sekali setahun

## S.7 Kunci

| No | Kunci | Rujukan | No | Kunci | Rujukan | No | Kunci | Rujukan |
|---|---|---|---|---|---|---|---|---|
| 1 | a | K-28 | 13 | b | G.7 | 25 | c | K-51 |
| 2 | c | 60.1 | 14 | c | Q.7 | 26 | b | K-52 |
| 3 | d | 104.1 | 15 | a | G.8 | 27 | a | K-53 |
| 4 | b | 147.1 | 16 | d | K-37 | 28 | d | K-58 |
| 5 | d | 77.2 | 17 | c | K-55 | 29 | b | K-57 |
| 6 | b | K-34 | 18 | b | G.8 | 30 | d | K-59 |
| 7 | c | G.1 | 19 | d | P.4 | 31 | a | 3.4 |
| 8 | a | K-27 | 20 | a | 184.1 | 32 | c | 173.1 |
| 9 | a | 69.1 | 21 | c | K-31 | 33 | d | O.2 |
| 10 | d | K-43 | 22 | a | K-50 | 34 | c | Q.3 |
| 11 | b | K-40 | 23 | b | 166.1 | 35 | a | N.1 |
| 12 | c | 172.1 | 24 | d | 112.1 | 36 | b | O.4 |

## S.8 Studi Latihan Tambahan

Tiga studi berikut dapat dinilai dengan rubrik yang sama seperti H.4.

**Studi 6 — Page space z/VM di 70%.** Pemakaian page space z/VM naik dari 35% ke 70% dalam sebulan setelah migrasi dua aplikasi ke Linux on Z. Belum ada gangguan. Susun langkah analisis, keputusan jangka pendek, dan perubahan proses agar kejadian K-41 tidak terjadi (Bab 160.1, 162.1).

**Studi 7 — Mitra ingin menaikkan volume API lima kali lipat.** Mitra e-commerce akan meluncurkan fitur baru yang memanggil API cek saldo lima kali lebih sering. Susun analisis dampak ke kapasitas, MSU, dan desain API, beserta syarat yang diajukan kepada mitra (Bab 54.1, 42.2, K-54).

**Studi 8 — Anggota senior terakhir di area DR akan pensiun dalam enam bulan.** Susun rencana transfer pengetahuan yang terukur, termasuk cara

membuktikan bahwa anggota lain sudah mampu memimpin uji DR (Bab 3.4, 91.1, O.1).

# Lampiran T. Tentang Buku Ini

## T.1 Ringkasan Isi

| Unsur | Jumlah |
|---|---|
| Bab utama | 193 |
| Runbook | 60 |
| Studi kasus | 60 (K-01 sampai K-60), ditambah satu studi kasus berseri |
| Praktikum | 10 |
| Soal pilihan ganda | 76 (Bab 138.1, H.1, S.1–S.6) |
| Soal hitungan dan studi latihan | 10 soal hitungan, 8 studi latihan |
| Gambar dan chart | 132 |
| Rumus | lebih dari 100 |
| Lampiran | A sampai T |

## T.2 Tentang Contoh dan Angka

Semua studi kasus, nama bank, nama orang, dan angka contoh dalam buku ini fiktif atau ilustratif, disusun dari pola yang umum terjadi di lingkungan mainframe perbankan. Angka seperti waktu respons, utilisasi, dan volume transaksi dipakai untuk menjelaskan cara berpikir, bukan sebagai acuan nilai yang benar untuk lingkungan tertentu.

Parameter, perintah, dan perilaku produk dapat berbeda antar-rilis. Selalu periksa dokumentasi resmi untuk rilis yang Anda pakai sebelum menerapkan perubahan di lingkungan produksi, dan uji setiap perubahan di lingkungan uji lebih dulu.

## T.3 Singkatan yang Sering Dipakai

| Singkatan | Kepanjangan |
|---|---|
| 4HRA | Four-hour rolling average (rata-rata bergerak empat jam) |
| APF | Authorized Program Facility |
| ARM | Automatic Restart Manager |
| AT-TLS | Application Transparent Transport Layer Security |
| BSDS | Bootstrap Data Set |
| CDC | Change Data Capture |
| CF | Coupling Facility |
| CFRM | Coupling Facility Resource Management |
| CKDS | Cryptographic Key Data Set |
| CPACF | CP Assist for Cryptographic Functions |
| DASD | Direct Access Storage Device |
| DDF | Distributed Data Facility |
| DVIPA | Dynamic Virtual IP Address |
| GDG | Generation Data Group |
| GDPS | Geographically Dispersed Parallel Sysplex |
| GRS | Global Resource Serialization |
| HCD | Hardware Configuration Definition |
| ICSF | Integrated Cryptographic Service Facility |
| IFL | Integrated Facility for Linux |
| IODF | I/O Definition File |
| IPL | Initial Program Load |
| LE | Language Environment |
| LPAR | Logical Partition |
| MLC | Monthly License Charge |
| MSU | Million Service Units |

| Singkatan | Kepanjangan |
|---|---|
| MTTR | Mean Time to Recover |
| PI | Performance Index |
| PR/SM | Processor Resource/Systems Manager |
| PSW | Program Status Word |
| RACF | Resource Access Control Facility |
| RLF | Resource Limit Facility |
| RLS | Record Level Sharing |
| RMF | Resource Measurement Facility |
| RPO / RTO | Recovery Point Objective / Recovery Time Objective |
| RTS | Real-Time Statistics |
| SCRT | Sub-Capacity Reporting Tool |
| SFM | Sysplex Failure Management |
| SLO | Service Level Objective |
| SMF | System Management Facilities |
| SMP/E | System Modification Program/Extended |
| SMS | Storage Management Subsystem |
| STP | Server Time Protocol |
| VSAM | Virtual Storage Access Method |
| WLM | Workload Manager |
| XCF | Cross-system Coupling Facility |
| zIIP | IBM Z Integrated Information Processor |

## T.4 Masukan dan Edisi Berikutnya

Buku ini akan terus diperbaiki. Koreksi teknis, pengalaman lapangan, studi kasus baru, dan usulan topik sangat diharapkan. Masukan dapat dikirim ke hello@rominur.com, dan pembaruan buku akan diumumkan di rominur.com.

Saat mengirim koreksi, sertakan nomor bab atau subbab, kutipan kalimat yang dimaksud, dan rujukan dokumentasi bila ada. Saat mengirim studi kasus, samarkan semua informasi yang dapat mengidentifikasi bank, nasabah, atau orang.

Topik yang direncanakan untuk edisi berikutnya antara lain pembaruan untuk generasi hardware dan rilis software terbaru, lebih banyak praktikum otomasi dengan Zowe dan Ansible, serta pendalaman keamanan kriptografi tahan komputer kuantum untuk data perbankan jangka panjang.

### T.5 Merek Dagang

IBM, IBM Z, z/OS, z/VM, CICS, Db2, IMS, MQ, RACF, Parallel Sysplex, GDPS, HyperSwap, FlashCopy, dan nama produk IBM lain yang disebut dalam buku ini adalah merek dagang atau merek dagang terdaftar International Business Machines Corporation di banyak yurisdiksi. Nama produk dan layanan lain dapat merupakan merek dagang milik perusahaan masing-masing. Penyebutan merek dalam buku ini hanya untuk keperluan penjelasan teknis.

## Indeks Tambahan

Indeks ini mencakup subbab analisis dan Lampiran F sampai T. Indeks Subjek sesudahnya mencakup bab-bab utama. Nomor seperti 65.4 menunjuk subbab, K-xx menunjuk studi kasus, dan huruf menunjuk lampiran.

| Subjek | Lokasi |
|---|---|
| Akrual tercatat ganda setelah rerun | 136.1, K-28 |
| ALE (perkiraan kerugian tahunan) | 46.1, I.4 |
| API agregat dan desain API lintas lokasi | 56.2, K-51 s.d. K-54, K-58 |
| ASUTIME dan RLF | 155.1, K-40 |
| Audit keamanan MQ | 166.1, R.7 |
| Backup: cakupan dan uji restore | 134.1, Q.3, R.3 |

| Subjek | Lokasi |
|---|---|
| Bandwidth replikasi dan RPO asinkron | G.7, R.10 |
| Bank Soal | 138.1, H, S |
| Batas thread dan koneksi Db2 | 68.2, K-27 |
| Beban jaga dan rotasi | 173.1, Q.5 |
| Benchmark dan selang kepercayaan | 135.1 |
| Biaya per transaksi | 1.3 |
| Burn rate dan anggaran kesalahan | 185.1 |
| CCSID dan titik konversi | 144.1, 145.1, K-26 |
| Change freeze | 126.1, Q.10 |
| Checkpoint, interval optimal | 60.1 |
| Checklist per komponen | R |
| CSD sebagai konfigurasi | 64.2 |
| Data sharing: retained lock | 71.1, F.3 |
| Dekomposisi waktu respons CICS | G.1 |
| DR: cakupan uji | 121.1, Q.7 |
| DR: kegagalan sebagian situs | G.8 |
| DR: switchover dan titik keputusan | 115.1, 116.1 |
| Duplexing structure | 179.1 |
| Dwell time dan retensi snapshot | 111.1, 113.2 |
| EOD: simulasi Monte Carlo | G.3 |
| EOD: terlambat, fase yang boleh ditunda | G.4 |
| EOD: utilisasi window | 104.1 |
| Faktor bus dan peta kompetensi | 3.4, Q.9 |
| Failure isolation | 179.1 |
| Frekuensi commit batch | 70.3, K-21 |
| GDG, batas generasi | 133.1 |
| Gladi rollback | 131.1, J.6 |

| Subjek | Lokasi |
|---|---|
| Hash total dan total kontrol | 61.1 |
| HiperSockets yang tidak terpakai | K-44 |
| Idempotensi | 102.3, P.2 |
| Image copy dan waktu pemulihan | 72.5 |
| Interface mitra: karantina file | 147.1, K-29 |
| Interface mitra: kontrak tertulis | Q.6, R.7 |
| Jawaban model studi latihan | P |
| Kasus K-21 s.d. K-40 | 189.2 s.d. 189.5 |
| Kasus K-41 s.d. K-60 | L |
| Kematangan kontrol | G.6 |
| Listing compile untuk diagnosis | 79.3 |
| Live Guest Relocation | 161.1, K-42 |
| Log aktif Db2 | G.2 |
| Matriks kontrol per ukuran bank | N.1 |
| Matriks migrasi kolektibilitas | 99.3 |
| Memori guest Linux on Z | 162.1, K-41, K-43 |
| Model ancaman mainframe | G.5 |
| MQT, manfaat bersih | 153.1 |
| Otomasi console berbasis pesan | 85.1, K-39 |
| Page space dan metrik z/VM | 160.1, K-41 |
| Parser SMF, validasi | 171.1 |
| Peta silang gejala | M |
| Pola dump yang sering muncul | 81.3 |
| Praktikum 6 s.d. 10 | F |
| Proyeksi kapasitas dengan skenario | 172.1 |
| PSW, membaca | 78.2 |
| Rantai bukti insiden | 112.1 |

| Subjek | Lokasi |
|---|---|
| REXX: kinerja dan penanganan error | 82.1, 84.1 |
| Rencana 90 hari | O |
| RLS, kapan diperlukan | 66.1 |
| Rumus, kumpulan | I |
| Runbook: kesehatan inventaris | 88.1 |
| Runbook: uji pembaca baru | 95.1 |
| S0C7, pencegahan | 77.2 |
| Sertifikat, perencanaan pembaruan | 149.2, Q.9 |
| SFM, penyetelan | 180.1 |
| SIEM, aturan korelasi | 184.1 |
| Signaling XCF | 178.1, K-36 |
| Sinyal dini sebelum insiden | 189.1 |
| Singkatan | T.3 |
| Studi kasus berseri | Q |
| Suspense, umur saldo | 100.3 |
| TCO, komponen | 177.1 |
| Templat kerja | J |
| Threadsafe | 65.4, K-24 |
| Timeout, rantai | K-52 |
| TLS, biaya CPU | 148.1, K-33 |
| Tipe numerik COBOL | 74.2 |
| Topologi region CICS | 63.3 |
| Ukuran bank, penyesuaian | N |
| Urutan sort EBCDIC dan ASCII | 146.1, K-30 |
| Velocity WLM, memilih tujuan | 168.1 |
| Zowe CLI dalam pipeline | 87.1, F.5 |

# Indeks Subjek


| Subjek | Bab |
|---|---|
| Abend, katalog kode | 16, 49, 79 |
| Akrual bunga | 58, 97, 98, 118 |
| Algoritma operasional | 57–62, 77, 96, 100, 102, 105, 106, 110, 113, 116, 126, 147, 149 |
| Anuitas dan kredit | 99 |
| AT-TLS dan sertifikat | 148, 149, 151 |
| Backup dan restore | 45, 120, 134 |
| Batch window dan EOD | 22, 35, 58, 61, 104–108 |
| BI-FAST, SKNBI, RTGS | 101, 102 |
| CCSID dan EBCDIC | 144–147 |
| CICS | 32, 63–67 |
| COBOL | 73–77, 135 |
| Capacity dan biaya MSU | 4, 42, 158 |
| Coupling Facility dan sysplex | 43, 71, 91, 121 |
| Cutover dan proyek | 128–131 |
| Cyber Vault dan ransomware | 113, 114 |
| Dataset, katalog, DFSMS | 16–18, 89 |
| Db2 for z/OS | 33, 68–72, 152–156 |
| DR, GDPS, HyperSwap | 44, 46, 115–117, 121 |
| Dump dan IPCS | 51, 78–81 |
| General Ledger | 61, 100 |
| HLASM | 78–80 |
| IPL dan PARMLIB | 12, 15, 60 |
| JCL dan JES2 | 19–22, 90, 133, 136 |
| Jaringan dan VIPA | 36–39, 93, 150 |
| Kalender bisnis | 106 |

| Subjek | Bab |
|---|---|
| Keamanan RACF | 27–31, 92, 111–114, 137 |
| Kolektibilitas | 99, 107 |
| Language Environment | 76 |
| LPAR, PR/SM, HMC | 5, 11 |
| MQ | 34, 93 |
| Pelaporan regulator | 109, 110 |
| Rekonsiliasi dan suspense | 103, 107, 147 |
| REXX dan otomasi | 26, 82–87 |
| Runbook, indeks | 52, 88, 95 |
| SMF dan RMF | 30, 41 |
| SMP/E dan pemeliharaan | 14, 128 |
| Tabel kontrol fase | 58, 105 |
| WLM | 40, 119 |
| z/OS UNIX dan zFS | 25, 89 |
| z/VM dan Linux on IBM Z | 8, 160–163 |
| IMS | 34, 167 |
| MQ mendalam dan queue sharing group | 164–166 |
| WLM rumus PI dan velocity | 168, 169 |
| SMF dengan Python | 170–172 |
| Jaga, eskalasi, postmortem | 173–175 |
| IBM Support, vendor, lisensi, TCO | 42, 176, 177 |