ANALISIS DAN IMPLEMENTASI *QUERY PERFORMANCE TUNING* PADA *MICROSOFT SQL SERVER MANAGEMENT STUDIO* DI PT ADICIPTA INOVASI TEKNOLOGI (ADINS)

PRAKTIK KERJA LAPANGAN

ALBERT WILLIAM SAPUTRA

NIM : 312210002

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG

2025

LEMBAR PENGESAHAN

PRAKTIK KERJA LAPANGAN

ANALISIS DAN IMPLEMENTASI *QUERY PERFORMANCE TUNING* PADA *MICROSOFT SQL SERVER MANAGEMENT STUDIO* DI PT ADICIPTA INOVASI TEKNOLOGI (ADINS)

Oleh:

ALBERT WILLIAM SAPUTRA

NIM. 312210002

dari:

PROGRAM STUDI TEKNIK INDUSTRI

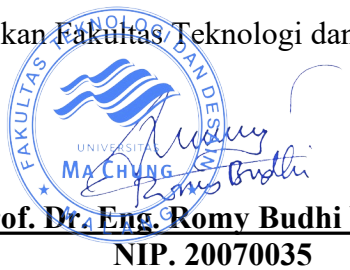
FAKULTAS TEKNOLOGI dan DESAIN

UNIVERSITAS MA CHUNG

Dosen Pembimbing,

Prof. Dr. Eng. Romy Budhi Widodo
NIP. 20070035

Dekan Fakultas Teknologi dan Desain,



Prof. Dr. Eng. Romy Budhi Widodo
NIP. 20070035

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmat yang dianugerahkan selama kegiatan Praktik Kerja Lapangan (PKL) berlangsung, sehingga penulis dapat menyusun dan menyelesaikan laporan dengan judul "**ANALISIS DAN IMPLEMENTASI *QUERY PERFORMANCE TUNING* PADA *MICROSOFT SQL SERVER MANAGEMENT STUDIO* DI PT ADICIPTA INOVASI TEKNOLOGI (ADINS)".** Laporan ini merupakan salah satu prasyarat untuk memperoleh gelar Sarjana Komputer di Universitas Ma Chung.

Terdapat banyak sekali dukungan yang diterima penulis dari awal hingga selesainya Praktik Kerja Lapangan ini. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu penulis, di antaranya:

1. Tuhan Yang Maha Esa, atas segala berkat dan karunia-Nya selama proses PKL hingga laporan ini dapat terselesaikan dengan baik.
2. Bapak Hendry Setiawan, ST., M.Kom., selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung.
3. Prof. Dr.Eng. Romy Budhi Widodo, selaku Dosen Pembimbing Praktik Kerja Lapangan yang telah memberikan banyak bimbingan, saran, dan motivasi dalam proses penyelesaian laporan ini.
4. Bapak Samsudin, selaku *Team Leader* di PT Adicipta Inovasi Teknologi yang telah membantu dan membimbing selama kegiatan Praktik Kerja Lapangan.
5. Ibu Calista Angie, selaku Pembimbing Lapangan di PT Adicipta Inovasi Teknologi yang telah membantu dan membimbing selama kegiatan Praktik Kerja Lapangan.
6. Seluruh tim *Database Administrator* (DBA) di PT Adicipta Inovasi Teknologi yang telah mendukung penulis dalam menyelesaikan tugas yang diberikan dan memberikan lingkungan kerja yang kondusif.
7. Universitas Ma Chung yang telah memberikan kesempatan kepada penulis untuk dapat memperoleh ilmu pengetahuan serta pengalaman dalam dunia kerja melalui kegiatan Praktik Kerja Lapangan.

8. Orang tua dan keluarga yang telah memberikan doa serta dukungan moril maupun materiil dalam menyelesaikan Laporan Praktik Kerja Lapangan ini.
9. Teman-teman Teknik Informatika angkatan 2022 yang telah membantu dan berdiskusi selama proses pembelajaran.

Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna, yang disebabkan oleh keterbatasan ilmu pengetahuan dan pengalaman dari penulis. Oleh karena itu, penulis ingin memohon maaf apabila terdapat kesalahan, baik dari segi ejaan maupun kalimat. Penulis juga sangat mengharapkan kritik dan saran yang membangun dari para pembaca.

Akhir kata, penulis berharap laporan ini dapat bermanfaat dan berguna bagi semua pihak yang membutuhkan.

Malang, 1 Agustus 2025

Albert William Saputra

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab I
Pendahuluan

1.1. Latar Belakang

Dalam bidang teknologi finansial (*fintech*) yang sangat kompetitif, performa aplikasi merupakan faktor krusial yang menentukan keberhasilan bisnis. PT Adicipta Inovasi Teknologi (AdIns), sebagai penyedia solusi perangkat lunak terkemuka untuk industri pembiayaan, sangat bergantung pada efisiensi dan keandalan sistem basis datanya. Produk unggulannya, CONFINS (*Consumer Finance and Leasing Solution*), mengelola volume data transaksi keuangan yang masif dan kompleks, di mana setiap milidetik waktu respons dapat berdampak langsung pada kepuasan klien, efisiensi operasional, dan profitabilitas perusahaan. Kinerja sistem yang superior bukan lagi sekadar keunggulan teknis, melainkan sebuah keharusan strategis untuk mempertahankan keunggulan kompetitif.

Salah satu tantangan utama dalam pemeliharaan sistem basis data berskala besar adalah degradasi performa yang disebabkan oleh kueri yang berjalan lambat. Kueri yang tidak efisien dapat memicu serangkaian masalah, termasuk waktu muat aplikasi yang lama, *timeout* pada level aplikasi, peningkatan beban sumber daya server secara signifikan (CPU, memori, dan I/O disk), serta pengalaman pengguna yang buruk secara keseluruhan. Menurut (Patil dkk., 2015), optimasi kueri adalah bagian tak terpisahkan dari pemeliharaan sistem basis data yang sehat dan efisien. Oleh karena itu, kemampuan untuk mengidentifikasi, mendiagnosis, dan menyelesaikan *bottleneck* performa pada level kueri menjadi kompetensi inti bagi tim teknis.

Selama pelaksanaan Praktik Kerja Lapangan (PKL) di PT AdIns, ditemukan bahwa keluhan klien sering kali berakar pada kueri atau *stored procedure* yang lambat. Analisis awal, seperti yang didokumentasikan dalam draf laporan sebelumnya, sering kali mengidentifikasi operator seperti *Index Scan* dan *Key Lookup* sebagai sumber masalah, dan menyelesaikannya melalui *Index Tuning*. Meskipun pendekatan ini valid dan sering kali efektif, ia cenderung menangani gejala daripada akar penyebabnya. Performa optimal yang sesungguhnya dicapai ketika sebuah kueri yang ditulis dengan baik (*well-written query*) dapat secara efisien memanfaatkan struktur basis data yang dirancang dengan baik (*well-designed database structure*).

Praktik kerja lapangan ini bertujuan untuk memperdalam analisis tersebut dengan memfokuskan pada aspek "kueri yang ditulis dengan baik". Laporan ini akan menggeser fokus dari sekadar pembuatan indeks baru menjadi sebuah investigasi mendalam terhadap logika dan struktur kueri itu sendiri. Tesis utama dari praktik kerja lapangan ini adalah bahwa banyak masalah performa yang bermanifestasi sebagai *Index Scan* atau penggunaan I/O yang tinggi sebenarnya berasal dari kueri yang ditulis secara sub-optimal, yang "memaksa" *Query Optimizer* SQL Server untuk memilih *execution plan* yang tidak efisien. Dengan demikian, praktik kerja lapangan ini akan mengeksplorasi teknik-teknik optimasi yang berpusat pada kueri (*query-centric tuning*) untuk mencapai peningkatan performa yang lebih fundamental dan berkelanjutan.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, praktik kerja lapangan ini dirumuskan untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana metodologi sistematis untuk mengidentifikasi dan mendiagnosis *bottleneck* performa pada kueri T-SQL di lingkungan produksi PT AdIns, dengan memanfaatkan kombinasi alat diagnostik seperti *Dynamic Management Views* (DMV), *SQL Server Extended Events*, dan analisis *Execution Plan*?

2. Apa saja teknik-teknik optimasi yang berpusat pada kueri (*query-centric*), seperti *refactoring* sintaks untuk memastikan SARGability, manajemen logika kompleks (misalnya, perbandingan antara *Common Table Expressions* (CTE) dan *temporary tables*), dan penanganan tantangan lingkungan produksi seperti *parameter sniffing*, yang dapat diimplementasikan untuk menyelesaikan *bottleneck* tersebut?

3. Bagaimana dampak kuantitatif dari implementasi teknik-teknik optimasi tersebut terhadap metrik performa kunci (waktu eksekusi, penggunaan CPU, *logical I/O*, dan *wait statistics*) sebelum dan sesudah intervensi dilakukan?

1.3. Tujuan Praktik Kerja Lapangan

Tujuan yang ingin dicapai melalui praktik kerja lapangan ini adalah sebagai berikut:

1. Menganalisis dan menerapkan sebuah metodologi diagnostik performa kueri yang komprehensif dan berbasis data di lingkungan teknis PT AdIns.

2. Mengimplementasikan dan memvalidasi efektivitas dari berbagai teknik optimasi kueri tingkat lanjut yang tidak hanya bergantung pada pembuatan atau modifikasi indeks.
3. Mengukur, mendokumentasikan, dan menyajikan peningkatan performa secara kuantitatif sebagai bukti empiris dari keberhasilan implementasi solusi yang diusulkan.

1.4. Manfaat Praktik Kerja Lapangan

Praktik kerja lapangan ini diharapkan dapat memberikan manfaat yang signifikan bagi berbagai pihak:

1. **Bagi Mahasiswa:**
    a. Memperoleh pemahaman teoretis dan praktis yang mendalam mengenai cara kerja internal *SQL Server Query Optimizer* dan mekanisme pemrosesan kueri.
    b. Mengembangkan keahlian praktis dalam *performance tuning* yang sangat relevan dan dicari di industri teknologi informasi, melampaui pengetahuan akademis dasar.
    c. Menghasilkan laporan PKL dengan kualitas analitis dan teknis yang tinggi, yang dapat menjadi portofolio berharga untuk karir di masa depan.
2. **Bagi Perusahaan (PT AdIns):**
    a. Mendapatkan solusi konkret dan terdokumentasi untuk masalah-masalah performa yang ada pada sistem CONFINS, yang dapat meningkatkan kualitas produk dan kepuasan klien.
    b. Menerima dokumentasi *best practices* untuk penulisan kueri yang efisien, yang dapat disosialisasikan sebagai standar internal bagi tim pengembang untuk mencegah masalah serupa di masa depan.
    c. Memperoleh wawasan dari analisis pihak ketiga yang objektif dan mendalam mengenai potensi optimasi dalam sistem basis data yang ada.

1.5. Batasan Masalah

Untuk memastikan praktik kerja lapangan ini tetap fokus dan mendalam, ditetapkan batasan-batasan sebagai berikut:

1. Praktik kerja lapangan ini difokuskan secara eksklusif pada optimasi kueri T-SQL, yang mencakup *stored procedure*, *view*, dan *ad-hoc query* yang berjalan pada sistem manajemen basis data Microsoft SQL Server 2017 di lingkungan PT AdIns.
2. Fokus utama dari optimasi adalah pada teknik-teknik yang mengubah atau memperbaiki logika kueri itu sendiri (*query rewriting*, *logic refactoring*, optimasi predikat). Analisis terhadap penggunaan indeks yang ada tetap dilakukan sebagai bagian integral dari proses diagnosis, namun pembuatan indeks baru secara masif bukanlah tujuan utama.
3. Lingkup optimasi tidak mencakup perubahan pada level kode aplikasi, konfigurasi server secara umum (di luar penggunaan *query hints* yang spesifik), atau perubahan pada infrastruktur perangkat keras dan jaringan.

Bab II
Gambaran Umum Perusahaan

2.1. Jenis Usaha Perusahaan

PT Adicipta Inovasi Teknologi, atau yang lebih dikenal dengan nama AdIns, merupakan perusahaan yang bergerak di bidang teknologi informasi dan didirikan pada tanggal 21 April 2000 oleh Bapak Guntur Gozali. Sejak awal pendiriannya, AdIns telah menempatkan fokus utamanya pada pengembangan solusi bisnis inovatif yang ditujukan bagi sektor keuangan, khususnya dalam industri multifinance dan leasing. Kantor pusat perusahaan ini berada di Jl. Kebon Jeruk Raya No. 80, Jakarta Barat, dan untuk memperluas jangkauan layanan, AdIns juga memiliki kantor cabang yang berlokasi di Jl. Lembah Dieng No. 7, Sumberjo, Kalisongo, Dau, Malang. Keberadaan dua kantor ini memperkuat posisi AdIns sebagai salah satu pelaku utama dalam industri, didukung oleh tim profesional muda yang penuh dedikasi. Informasi lebih lengkap mengenai perusahaan ini dapat diakses melalui website resminya di *https://www.ad-ins.com/*, yang memuat berbagai profil, layanan, serta pencapaian AdIns. Logo resmi perusahaan dapat dilihat pada Gambar 2.1.

Gambar 2.1 Logo AdIns

AdIns memiliki visi untuk *"menjadi berkat bagi masyarakat melalui inovasi di bidang Teknologi Informasi dan Komunikasi (TIK)"*. Visi ini diwujudkan tidak hanya melalui produk dan solusi yang ditawarkan, tetapi juga melalui lingkungan kerja yang mendukung pertumbuhan karyawan serta menciptakan dampak positif secara berkelanjutan. Misi perusahaan dirumuskan dalam empat pilar utama yaitu, mengembangkan potensi sumber daya manusia, menjadi mitra terpercaya dengan solusi kelas dunia, berkontribusi sebagai aset bangsa berdasarkan nilai-nilai Pancasila, serta memberikan hasil terbaik bagi para pemegang saham. Visi dan misi ini mencerminkan komitmen AdIns terhadap inovasi, kualitas layanan, dan integritas dalam menjalankan kegiatan operasionalnya.

Sejak awal perjalanannya, AdIns telah menghadirkan berbagai solusi teknologi yang mendukung aktivitas keuangan, mulai dari pengelolaan pembiayaan hingga sistem leasing yang terintegrasi. Pengalaman Bapak Guntur Gozali, selaku pendiri, yang sebelumnya bekerja di perusahaan pembiayaan besar seperti *Astra Credit Company* (ACC) menjadi salah satu faktor kunci keberhasilan dalam membangun landasan perusahaan ini. Keahliannya di bidang teknologi informasi, yang

Gambar 2.2 Logo CONFINS 1

dikombinasikan dengan kerja sama tim yang solid, memungkinkan AdIns untuk terus berkembang dan memperkuat posisinya di industri.

Salah satu produk unggulan yang menjadi bukti nyata dari inovasi AdIns adalah CONFINS (*Consumer Finance and Leasing Solution*). Produk ini telah menjadi andalan banyak klien AdIns dalam mengelola proses bisnis mereka secara efisien dan terstruktur. Identitas visual dari produk CONFINS ditunjukkan pada Gambar 2.2. Platform ini dirancang secara khusus untuk membantu perusahaan pembiayaan dalam mengelola data dan proses operasional nasabah secara efisien dan terstruktur. Keberhasilan CONFINS, bersama dengan berbagai solusi lain yang ditawarkan, mencerminkan komitmen AdIns dalam menjaga kualitas layanan dan membangun kepercayaan klien. AdIns juga terus melakukan pengembangan produk secara berkelanjutan agar tetap relevan dengan kebutuhan pasar yang terus berkembang. Dengan menjunjung tinggi nilai integritas dan semangat inovasi, AdIns senantiasa menghadirkan solusi teknologi yang kompetitif dan adaptif terhadap perubahan zaman.

2.1.1. Jenis dan Produk Layanan PT Adicipta Inovasi Teknologi

PT Adicipta Inovasi Teknologi (AdIns) menyediakan portofolio produk dan layanan yang komprehensif, dirancang untuk mendukung transformasi digital dan keunggulan operasional di industri keuangan.

1. Aplikasi *Core System* – CONFINS

Gambar 2.3 Logo CONFINS 2

CONFINS merupakan sistem inti yang dirancang khusus untuk mendukung pengelolaan menyeluruh bagi perusahaan pembiayaan, mulai dari data nasabah hingga proses transaksi keuangan. Sebagai produk utama AdIns, CONFINS berperan penting dalam mendorong efisiensi operasional dan mendukung pertumbuhan bisnis yang berkelanjutan.

2. *Digital Signature*

Gambar 2.4 Logo Digital Signature

Solusi tanda tangan digital dari AdIns memungkinkan proses otorisasi dokumen dilakukan secara elektronik, cepat, dan aman. Fitur ini mempercepat alur kerja administratif serta mendukung digitalisasi proses bisnis secara menyeluruh.

3. *Optical Character Recognition* (OCR)

Gambar 2.5 Logo Optical Character Recognition (OCR)

Teknologi OCR berbasis AI ini mempermudah konversi dan input data dari dokumen fisik ke format digital, seperti KTP, NPWP, STNK, dan dokumen

penting lainnya. Dengan demikian, proses validasi data menjadi lebih cepat dan efisien.

4. Aplikasi *Mobile Multifinance* – AdIns Mobile

Gambar 2.6 Logo Aplikasi Mobile Multifinance – AdIns Mobile

AdIns Mobile adalah aplikasi yang dikembangkan untuk mendukung kegiatan operasional di lapangan. Dengan fitur-fitur *mobile-friendly*, aplikasi ini membantu tim multifinance dalam meningkatkan produktivitas, mobilitas, dan pengawasan kerja secara *real-time*.

5. Lite DMS (*Document Management System*)

Gambar 2.7 Logo Lite DMS

Lite DMS merupakan sistem pengelolaan dokumen digital yang dilengkapi pencarian cepat dan sistem enkripsi. Solusi ini memastikan dokumen penting tersimpan dengan aman serta mudah diakses kapan saja sesuai kebutuhan.

6. EKYC (*Electronic Know Your Customer*) – PROFIND

Gambar 2.8 Logo PROFIND

PROFIND mempermudah proses pencarian dan verifikasi data calon debitur secara cepat, akurat, dan otomatis. Aplikasi ini mendukung proses analisis risiko kredit yang lebih tepat sasaran dan efisien.

7. SUPRBOARD – *IT Network Monitoring*

Gambar 2.9 Logo SUPRBOARD

SUPRBOARD adalah *platform monitoring* yang berfungsi untuk memantau performa jaringan dan aplikasi. Dengan sistem ini, perusahaan dapat mengidentifikasi masalah teknis lebih awal dan menjaga stabilitas infrastruktur TI.

8. ARS – *Business Intelligence Dashboard*

Gambar 2.10 Logo ARS

ARS menyediakan tampilan visual berbasis dashboard yang menyajikan data bisnis secara *real-time*. Informasi ini mendukung manajemen dalam mengambil keputusan strategis berbasis data yang akurat dan relevan.

9. IT *Service*

Gambar 2.11 Logo IT Service

AdIns menawarkan layanan konsultasi dan dukungan teknis di bidang infrastruktur IT, pengelolaan sistem, keamanan jaringan, dan *service desk*. Layanan ini dirancang untuk meningkatkan efektivitas operasional teknologi di perusahaan.

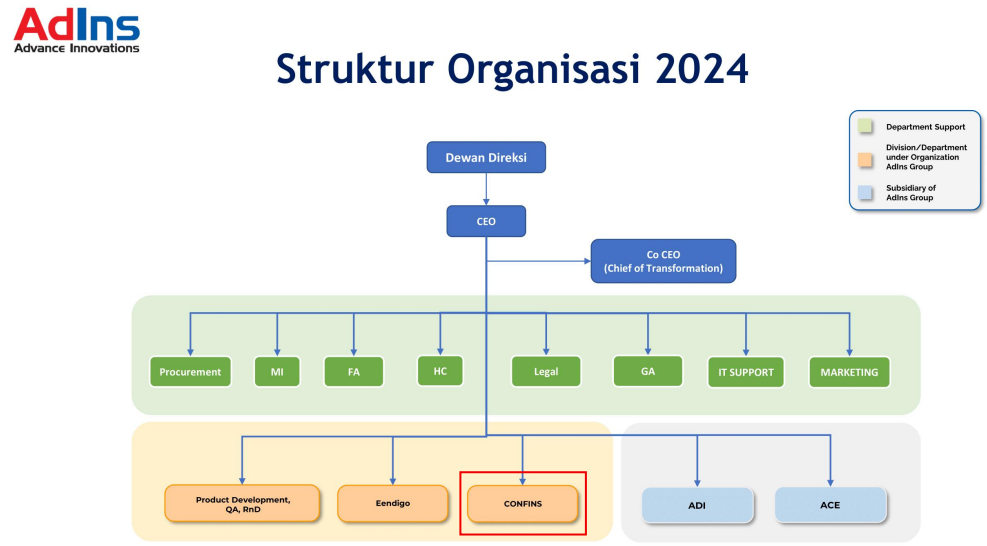
10. DOCUPRO – *Data Entry Services*

Gambar 2.12 Logo DOCUPRO

DOCUPRO menyediakan layanan entri dan verifikasi data dalam skala besar oleh tenaga profesional. Layanan ini menjamin keakuratan data dan efisiensi pengolahan informasi yang dibutuhkan oleh perusahaan klien.

2.2. Struktur Organisasi

Berikut adalah struktur organisasi dari PT Adicipta Inovasi Teknologi yang ditampilkan pada Gambar 2.13.



Gambar 2.13 Struktur Organisasi AdIns

Struktur organisasi PT Adicipta Inovasi Teknologi (AdIns) seperti pada Gambar 2.13 dirancang secara hierarkis untuk memastikan alur komando dan koordinasi yang jelas, yang dipimpin oleh Dewan Direksi dan seorang CEO. Di bawah CEO, terdapat beberapa departemen pendukung (*support*) yang melayani seluruh lini bisnis, seperti *Procurement, Management Information (MI), Finance & Accounting (FA), Human Capital (HC), Legal, General Affairs (GA), IT Support*, dan *Marketing*. Departemen-departemen ini berfungsi sebagai tulang punggung operasional perusahaan, memastikan semua kegiatan berjalan lancar.

Selanjutnya, struktur organisasi terbagi menjadi dua kelompok bisnis utama. Kelompok pertama, yang berfokus pada pengembangan produk inti, mencakup tim *Product Development, QA, RMO*, tim *Eendigo*, dan tim CONFINS. Di sinilah posisi saat PKL dilakukan, yaitu di dalam tim CONFINS pada bagian *Infra & Database* sebagai seorang DBA. Kelompok kedua adalah anak perusahaan (*subsidiary*) dari AdIns Group, yaitu ADI dan ACE, yang beroperasi sebagai entitas bisnis terpisah namun tetap berada di bawah naungan grup AdIns.

2.3.1. Peran dan Tanggung Jawab

Praktik Kerja Lapangan dilakukan dalam tim CONFINS pada bagian *Infrastructure & Database* (Infra DB) dan berperan sebagai DBA. Dalam posisi ini, DBA bertanggung jawab terhadap pengelolaan dan pemeliharaan sistem basis data yang digunakan dalam produk CONFINS. Tugas-tugas yang dijalankan antara lain meliputi pemantauan performa *database*, optimasi *query*, pengelolaan *backup* dan *restore*, penanganan isu teknis, serta memastikan keamanan dan integritas data. DBA juga turut terlibat dalam proses deployment dan pengujian, guna memastikan sistem berjalan stabil dan dapat melayani kebutuhan klien dengan baik.

2.3.2. Hubungan Kerja Antar Divisi

Dalam menangani isu performa sistem, tim DBA berkoordinasi erat dengan tim DOCs dan tim pengembang lainnya. Proses dimulai saat keluhan dari klien atau temuan internal dicatat dalam sistem Jira. Tim DOCs bertanggung jawab melakukan validasi awal untuk memastikan masalah tidak berasal dari sisi pengguna atau *front-end*. Setelah dipastikan berkaitan dengan basis data, tiket akan dialihkan ke tim DBA untuk ditindaklanjuti. Hubungan kerja yang terstruktur dan kolaboratif antar divisi ini memungkinkan penanganan masalah dilakukan secara lebih cepat, efisien, dan tepat sasaran.

2.3. Operasional Perusahaan

2.3.1. Visi

Visi PT Adicipta Inovasi Teknologi (AdIns) adalah *"Menjadi berkat bagi masyarakat melalui inovasi di bidang Information and Communication Technology (ICT), baik melalui layanan produk maupun budaya kerja yang mendukung pertumbuhan karyawan serta menciptakan dampak positif yang berkelanjutan."*

Visi ini mencerminkan komitmen AdIns untuk tidak hanya berperan sebagai penyedia solusi teknologi, tetapi juga sebagai organisasi yang memberikan nilai tambah nyata bagi masyarakat. AdIns tidak semata-mata berorientasi pada keuntungan bisnis, melainkan berupaya menghadirkan perubahan positif melalui inovasi teknologi dan pengembangan sumber daya manusia. Budaya kerja yang inklusif, kolaboratif, dan berorientasi pada pembelajaran berkelanjutan menjadi

fondasi utama dalam menciptakan dampak yang bersifat jangka panjang dan berkelanjutan.

2.3.2. Misi

Untuk mewujudkan visinya, AdIns memiliki empat misi utama yang menjadi pedoman dalam seluruh aktivitas dan pengambilan keputusan strategis perusahaan:

1. Mendorong dan mendukung karyawan untuk bertumbuh
2. Menjadi mitra andalan dengan solusi terbukti menggunakan inovasi kelas dunia di bidang teknologi informasi dan komunikasi
3. Menjadi aset negara berdasarkan Pancasila
4. Memberikan imbal balik terbaik kepada pemegang saham

2.4. Tim Database

Praktik kerja lapangan ini dilakukan dalam lingkungan teknis PT AdIns yang menggunakan *Microsoft SQL Server* 2017 sebagai *Database Management System* (DBMS) utama. Basis data yang dianalisis umumnya beroperasi dalam mode *Full Recovery*, yang penting untuk kebutuhan *point-in-time restore* dan *disaster recovery* di industri keuangan, namun juga memberikan tantangan tersendiri terkait manajemen *transaction log*. Beban kerja (*workload*) pada sistem ini didominasi oleh transaksi *online* (*Online Transaction Processing* - OLTP), yang ditandai oleh banyaknya transaksi singkat (baca, tulis, perbarui), diselingi oleh kueri pelaporan analitis yang lebih berat dan kompleks. Kombinasi beban kerja ini menuntut basis data yang mampu menangani konkurensi tinggi sekaligus pemrosesan data yang efisien.

Tim *Database* di PT AdIns, yang berada di bawah departemen Infra & Database, memegang tanggung jawab krusial dalam menjaga kesehatan ekosistem basis data ini. Tugas utama tim ini meliputi perancangan dan pemeliharaan skema database yang efisien, serta pengembangan dan pemeliharaan objek basis data seperti *stored procedure*, *function*, dan *trigger*. Selain itu, tim juga bertanggung jawab untuk melakukan *troubleshooting* dan perbaikan terhadap isu-isu yang berkaitan dengan database, termasuk *bug fixing* dan *enhancement*. Yang paling relevan dengan praktik kerja lapangan ini adalah tugas mereka dalam melakukan optimalisasi performa kueri untuk memastikan aplikasi berjalan dengan responsif dan efisien. Konteks

inilah yang menjadi latar belakang utama dari kegiatan Praktik Kerja Lapangan yang diuraikan dalam laporan ini.

2.5. Peraturan Kerja

Selama periode Praktik Kerja Lapangan (PKL), perusahaan menerapkan sistem kerja hibrida yang menggabungkan *Work From Home* (WFH) dan *Work From Office* (WFO). WFH ditetapkan pada hari Senin, Rabu, serta akhir bulan (*End of Month/EOM*), di mana karyawan dan peserta PKL bekerja dari rumah dengan tetap memenuhi seluruh target dan tanggung jawab pekerjaan. Sebaliknya, WFO dilaksanakan pada hari Selasa, Kamis, dan Jumat, yang mewajibkan peserta PKL hadir di kantor untuk melakukan kolaborasi secara langsung dengan anggota tim dan atasan.

Seluruh aktivitas kerja peserta PKL wajib mematuhi Standar Operasional Prosedur (SOP) yang berlaku di perusahaan. Proses penugasan dilakukan secara terstruktur, dengan Supervisor (SPV) menugaskan tugas melalui *platform* komunikasi internal, yaitu Microsoft Teams. Tugas-tugas tersebut umumnya berasal dari Jira Helpdesk, sistem pelaporan resmi yang digunakan oleh PIC tim Docs masing-masing klien untuk menyampaikan keluhan atau permintaan perbaikan berdasarkan hasil pemantauan sistem. Peserta PKL diwajibkan mencatat dan memperbarui perkembangan tugas secara berkala melalui Jira Cloud, sebuah platform manajemen proyek yang digunakan untuk memantau status pekerjaan setiap individu.

Pelaporan perkembangan tugas ini bertujuan untuk menjamin transparansi dan akuntabilitas dalam pelaksanaan pekerjaan, sekaligus memudahkan SPV dalam melakukan evaluasi kinerja. Komunikasi yang efektif antara peserta PKL dan SPV menjadi aspek krusial untuk mengatasi kendala teknis yang mungkin timbul selama proses pengerjaan tugas. Setelah tugas selesai, SPV akan melakukan peninjauan untuk memastikan hasil pekerjaan memenuhi standar kualitas yang ditetapkan. Apabila hasil pekerjaan disetujui, peserta PKL bersama SPV akan menyusun sebuah paket yang berisi hasil akhir pekerjaan, dokumentasi teknis, serta dokumen pendukung seperti *User Manual Testing* (UMT). Paket tersebut kemudian diunggah dan dilampirkan pada Jira Helpdesk, yang berfungsi sebagai platform resmi untuk penyerahan tugas kepada PIC tim Docs klien. Melalui Jira Helpdesk, diskusi teknis

lanjutan dengan klien dapat dilakukan untuk keperluan klarifikasi atau tindak lanjut lebih lanjut.

Bab III
Tinjauan Pustaka

3.1. SQL Server

SQL Server adalah sebuah sistem manajemen basis data relasional (*Relational Database Management System* - RDBMS) yang dikembangkan oleh Microsoft dan digunakan secara luas untuk mengelola data dalam skala besar dengan fokus pada transaksi dan integritas data (Chen, 2023). Selama kegiatan PKL, DBA secara langsung mengelola dan memelihara basis data SQL Server yang mendukung berbagai sistem krusial, mulai dari aplikasi *lending system* untuk klien hingga sistem manajemen internal perusahaan. Penggunaannya di PT AdIns tidak hanya terbatas pada penyimpanan data, tetapi juga sebagai mesin pemrosesan transaksi berkecepatan tinggi yang menuntut ketersediaan dan integritas data yang solid.

Dalam praktik sehari-hari, fitur-fitur *SQL Server* dimanfaatkan secara ekstensif. Sebagai contoh, untuk menjaga performa aplikasi yang mengelola data transaksi nasabah, DBA secara rutin melakukan analisis dan membuat rekomendasi *indexing* pada tabel-tabel yang tumbuh pesat. *SQL Server* juga memiliki fitur seperti *Data Tools* dan *SQL Server Integration Services* (SSIS) yang mendukung kebutuhan pengolahan data. Kelebihan lainnya meliputi kemampuan dalam ketersediaan tinggi (*high availability*) seperti *Always on Availability Groups*, fitur *backup & restore* yang kuat, serta integrasi yang baik dengan *SQL Server Management Studio* (SSMS). Fitur seperti *stored procedure* juga digunakan untuk mengenkapsulasi logika bisnis yang kompleks, sehingga mengurangi lalu lintas jaringan antara aplikasi dan server basis data. Namun, penggunaan SQL Server di lingkungan PT AdIns juga menghadirkan tantangan. Salah satu yang paling signifikan adalah biaya lisensi yang mahal jika dibandingkan dengan alternatif *open-source*. Selain itu, pada server dengan beban transaksi yang sangat tinggi, konsumsi sumber daya (CPU dan Memori) dapat menjadi *bottleneck* yang memerlukan pemantauan ketat dan konfigurasi yang cermat untuk dihindari.

Peran sebagai DBA tidak hanya berfokus pada pengembangan atau *tuning*, melainkan juga pada aspek operasional dan pemeliharaan. Tanggung jawab utama meliputi pemantauan kesehatan server, memastikan jadwal *backup* harian berjalan sukses untuk pemulihan bencana, mengelola utilisasi penyimpanan (*disk space*), dan menjadi garda terdepan dalam menangani insiden yang berkaitan dengan ketersediaan atau konektivitas basis data. Menurut (Fritchey, 2018), administrasi

basis data yang proaktif adalah kunci untuk mencegah masalah performa, sebuah prinsip yang diterapkan melalui pemeliharaan rutin di PT AdIns.

3.2. SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) adalah sebuah aplikasi perangkat lunak yang menyediakan lingkungan terpadu (*integrated environment*) untuk mengakses, mengonfigurasi, mengelola, dan mengembangkan semua komponen SQL Server (Nevarez, 2022). SQL Server Management Studio (SSMS) memungkinkan pengguna untuk mengakses, mengonfigurasi, mengelola, dan mengembangkan semua komponen SQL Server dalam satu antarmuka terpadu (Microsoft, 2025). SSMS juga dianggap sebagai alat esensial bagi administrator dan developer karena menyederhanakan manajemen, eksekusi kueri, serta monitoring kinerja sistem (GeeksforGeeks, 2024). Melalui antarmuka grafisnya yang intuitif, pengguna dapat melakukan berbagai tugas, mulai dari menulis dan mengeksekusi kueri T-SQL hingga merancang skema basis data, serta mengelola objek-objek seperti tabel, *views*, *stored procedure*, dan *function*. SSMS juga mendukung pengelolaan keamanan seperti pengaturan *role* dan *permission* bagi pengguna, serta fitur untuk melakukan *backup* dan *restore* data.

Fitur inti dari SSMS yang paling sering digunakan adalah *Query Editor*. Editor ini tidak hanya berfungsi sebagai tempat untuk menulis skrip SQL, tetapi juga dilengkapi dengan fitur modern seperti *syntax highlighting* yang membedakan sintaks dengan warna untuk kemudahan pembacaan, dan *IntelliSense* yang memberikan saran pelengkapan kode secara otomatis untuk mengurangi kesalahan pengetikan dan mempercepat proses penulisan kueri. Selain itu, SSMS menyediakan kemampuan untuk memvisualisasikan *Execution Plan* (Rencana Eksekusi), sebuah alat diagnostik vital yang membantu DBA menganalisis bagaimana sebuah kueri diproses oleh SQL Server dan mengidentifikasi potensi masalah performa.

Di luar fungsionalitas pengembangan, SSMS adalah pusat komando untuk tugas-tugas administratif. Administrator dapat mengelola keamanan dengan mudah, seperti membuat *login*, mengatur pengguna basis data, dan memberikan hak akses (*permissions*) pada level yang sangat spesifik untuk memastikan hanya pihak berwenang yang dapat mengakses atau memodifikasi data. Tugas-tugas pemeliharaan rutin, seperti melakukan pencadangan data (*backup*) dan pemulihan (*restore*), menjadwalkan tugas otomatis dengan SQL Server Agent, serta memantau

kesehatan dan aktivitas server secara *real-time* melalui *Activity Monitor*, semuanya dapat dilakukan secara efisien melalui SSMS.

3.3. Execution Plan

Execution Plan adalah representasi grafis atau tekstual dari strategi yang digunakan oleh mesin basis data untuk mengeksekusi suatu perintah SQL (Nevarez, 2022). Seorang DBA menggunakan *Execution Plan* untuk memahami bagaimana sistem memilih metode akses data, seperti *Index Scan* atau *Index Seek*.

Dengan menganalisis representasi grafis dari Execution Plan di SSMS, seorang DBA dapat secara cepat mengidentifikasi operator atau langkah-langkah yang menjadi sumber masalah performa. Setiap operator dalam rencana tersebut memiliki "biaya" (cost) yang diestimasi, yang merepresentasikan persentase penggunaan sumber daya relatif terhadap total biaya kueri. Operator dengan biaya tertinggi (*high-cost operators*) sering kali menjadi fokus utama dalam investigasi. Misalnya, sebuah operator *Index Scan* pada tabel yang sangat besar dengan biaya 90% menunjukkan bahwa kueri tersebut harus membaca terlalu banyak data yang tidak perlu, yang mungkin dapat diatasi dengan membuat indeks yang lebih sesuai.

Analisis mendalam terhadap *Execution Plan* memberikan wawasan yang dapat ditindaklanjuti untuk optimasi. Berdasarkan temuan, DBA dapat menentukan langkah perbaikan yang paling efektif. Solusi bisa bervariasi, mulai dari tindakan sederhana seperti memperbarui statistik data agar *Query Optimizer* memiliki informasi yang akurat, hingga intervensi yang lebih kompleks. Ini mungkin termasuk menyusun ulang logika kueri agar lebih efisien (misalnya, menghindari fungsi dalam klausa *WHERE*), merancang ulang atau menambahkan indeks baru untuk mengubah *Index Scan* menjadi *Index Seek*, atau bahkan dalam beberapa kasus, merekomendasikan perubahan pada desain skema basis data untuk mendukung pola kueri yang lebih baik

3.4. Query Performance Tuning

Query Performance Tuning adalah proses sistematis untuk meningkatkan kecepatan dan efisiensi eksekusi kueri SQL (lebih dari 50%) dan melibatkan banyak metode seperti statistik, index, reorganization, hingga materialized view (Patil dkk., 2015). Tujuannya adalah untuk mengurangi waktu respons aplikasi, mengoptimalkan penggunaan sumber daya *server* (CPU, memori, I/O), dan

meningkatkan *throughput* sistem basis data. Dalam lingkungan basis data yang besar dan kompleks seperti di PT AdIns, *query performance tuning* menjadi sangat penting untuk menjaga kelancaran operasional bisnis dan kepuasan klien.

Proses ini bukanlah sekadar aktivitas reaktif, melainkan siklus proaktif yang melibatkan beberapa tahapan. Tahap pertama adalah identifikasi, di mana DBA mencari kueri-kueri yang berjalan lambat atau mengonsumsi sumber daya secara berlebihan, sering kali menggunakan alat pemantauan seperti *Dynamic Management Views* (DMVs) atau *SQL Server Profiler*. Setelah kueri bermasalah ditemukan, tahap selanjutnya adalah analisis, yang melibatkan pemeriksaan *Execution Plan* untuk memahami akar penyebab inefisiensi. Berdasarkan analisis ini, DBA akan merumuskan hipotesis dan mengimplementasikan solusi optimasi yang paling sesuai.

Terdapat berbagai teknik yang umum digunakan dalam *query performance tuning*. Teknik yang paling fundamental adalah manajemen indeks, yang mencakup pembuatan indeks baru untuk mendukung pencarian data, memodifikasi indeks yang ada, atau menghapus indeks yang tidak lagi digunakan dan hanya menambah beban pada operasi tulis. Teknik lainnya adalah optimasi sintaks kueri itu sendiri, seperti menulis ulang kueri untuk menghindari predikat yang tidak dapat diindeks (*non-SARGable*), serta memastikan statistik basis data selalu diperbarui agar *Query Optimizer* dapat membuat keputusan yang cerdas. Proses ini diakhiri dengan validasi untuk memastikan bahwa perubahan yang diterapkan benar-benar memberikan peningkatan performa yang diharapkan.

3.5. Jira

Jira adalah perangkat lunak manajemen proyek berbasis Agile yang digunakan untuk melacak bug, tugas, dan pengembangan perangkat lunak, serta memfasilitasi kerja kolaboratif tim melalui sistem tiket dan dashboard yang dapat dikustomisasi (Atlassian, 2023). Dalam konteks pekerjaan sebagai DBA di PT AdIns, Jira berfungsi sebagai platform utama untuk manajemen tugas dan alur kerja terkait masalah performa basis data. Setiap kali ada keluhan mengenai kelambatan sistem yang berasal dari klien atau terdeteksi oleh tim internal, sebuah *ticket* akan dibuat di Jira. *Ticket* ini berfungsi sebagai wadah tunggal untuk semua informasi terkait isu tersebut, termasuk deskripsi masalah, lingkungan yang terdampak, dan tingkat urgensi.

Penggunaan Jira memungkinkan alur kerja yang terstruktur dan transparan. Ketika sebuah *ticket* performa dibuat, *ticket* tersebut akan masuk ke dalam *backlog* tim DBA dan kemudian ditugaskan kepada salah satu orang dari tim DBA. Sejak saat itu, Jira digunakan untuk melacak seluruh progres penanganan, mulai dari tahap analisis awal, investigasi penyebab masalah, implementasi solusi (misalnya, optimasi kueri atau pembuatan indeks), hingga koordinasi dengan tim DOCs untuk melakukan verifikasi. Setiap pembaruan, komentar, dan lampiran (seperti skrip SQL atau hasil analisis) dicatat dalam *ticket*, menciptakan jejak audit yang lengkap.

Selain melacak progres, Jira juga digunakan untuk mengukur efektivitas dan efisiensi kerja tim melalui metrik *Key Performance Indicator* (KPI). Salah satu metrik penting yang dipantau adalah *maindays* (sering disebut juga MD), yang merepresentasikan estimasi atau jumlah hari kerja efektif yang dibutuhkan untuk menyelesaikan sebuah tugas. Dengan mencatat estimasi *maindays* di awal dan membandingkannya dengan waktu aktual yang dihabiskan, manajemen dapat mengevaluasi akurasi perencanaan, mengidentifikasi *bottleneck* dalam proses kerja, dan memastikan bahwa sumber daya tim dialokasikan secara efisien untuk memenuhi target penyelesaian yang telah ditetapkan.

Lebih dari sekadar alat pelacakan tugas, Jira memfasilitasi kolaborasi dan akuntabilitas. Melalui sistem notifikasi dan komentar, DBA dapat dengan mudah berdiskusi dengan pelapor masalah atau tim pengembang aplikasi untuk mendapatkan klarifikasi. Status *ticket* yang dapat diubah, seperti *"To Do", "In Progress", "In Review",* dan *"Done"* hal ini memberikan visibilitas yang jelas kepada semua pemangku kepentingan, termasuk manajer proyek dan *team leader*, mengenai kemajuan penyelesaian masalah. Setelah solusi divalidasi dan dikonfirmasi berhasil, *ticket* akan ditutup, menandakan bahwa siklus penanganan isu tersebut telah selesai.

3.6. Microsoft Teams

Microsoft Teams merupakan platform kolaborasi terpadu yang menyediakan layanan komunikasi real-time seperti chat, video meeting, serta integrasi dengan aplikasi Microsoft 365 untuk mendukung produktivitas dan kolaborasi tim dalam satu tempat kerja digital (Microsoft, 2025). Selama pelaksanaan Praktik Kerja Lapangan, Teams digunakan sebagai sarana komunikasi utama untuk kolaborasi antar tim secara sinkron maupun asinkron. Platform ini sangat vital untuk menjaga

koordinasi yang cepat dan efektif, terutama ketika menangani masalah performa yang memerlukan masukan dari berbagai pihak.

Dalam tugas sehari-hari, DBA menggunakan Microsoft Teams untuk berdiskusi secara langsung dengan tim pengembang (*developer*). Misalnya, ketika sebuah kueri yang lambat memiliki logika bisnis yang kompleks, sesi panggilan video dengan fitur berbagi layar (*screen sharing*) memungkinkan untuk menunjukkan *Execution Plan* dan mendiskusikan bagian mana dari kueri yang perlu ditulis ulang. Selain itu, Teams juga digunakan untuk berkoordinasi dengan tim *DOCs* dalam proses pengujian solusi optimasi, memastikan bahwa perbaikan tidak menimbulkan efek samping yang tidak diinginkan pada fungsionalitas aplikasi. Lebih dari itu, Teams diintegrasikan dengan sistem pemantauan untuk otomatisasi notifikasi. Sebagai contoh, notifikasi mengenai status sinkronisasi *database* dan *log shipping* dikirimkan secara otomatis ke *channel* khusus DBA. Hal ini sangat memudahkan pengecekan proaktif dan memungkinkan tim untuk segera mengetahui jika terjadi kegagalan atau keterlambatan sinkronisasi data antar server.

Efisiensi komunikasi melalui Teams membantu mempercepat siklus penyelesaian masalah. Daripada mengandalkan email yang cenderung lambat, percakapan cepat melalui obrolan atau panggilan dapat menyelesaikan ambiguitas dalam hitungan menit. Teams juga memungkinkan pembuatan "Channel" khusus untuk topik atau proyek tertentu, misalnya channel #Infra DB, di mana anggota tim dapat berbagi temuan, skrip, dan melaporkan hasil optimasi kepada *team leader*. Integrasinya dengan alat lain seperti Jira dan OneDrive semakin memperkuat perannya sebagai pusat kolaborasi digital di lingkungan kerja PT AdIns.

3.7. Microsoft Outlook

Microsoft Outlook adalah aplikasi manajemen informasi pribadi yang paling banyak digunakan di lingkungan bisnis, mencakup fungsi email, kalender, manajemen tugas, dan kontak, serta dapat terintegrasi dengan platform lain seperti Teams dan SharePoint (Microsoft, 2023). Berbeda dengan Microsoft Teams yang lebih fokus pada kolaborasi real-time, Outlook digunakan untuk diskusi yang bersifat asinkron, terdokumentasi, dan sering kali melibatkan pihak eksternal. Seluruh komunikasi resmi seperti pengiriman proposal, tindak lanjut dengan klien, atau penyebaran informasi kebijakan internal dilakukan melalui email untuk memastikan adanya jejak digital yang jelas dan dapat dipertanggungjawabkan.

Salah satu fungsi terpenting Outlook dalam kegiatan sehari-hari adalah sebagai alat penjadwalan. Melalui fitur kalender yang terintegrasi, tim DBA dapat dengan mudah mengatur jadwal rapat, baik dengan tim internal maupun dengan klien. Fitur Scheduling Assistant sangat membantu dalam menemukan waktu yang cocok bagi semua peserta tanpa perlu bertanya satu per satu. Undangan rapat yang dikirim melalui Outlook secara otomatis terintegrasi dengan kalender penerima dan dapat menyertakan tautan rapat Microsoft Teams, menciptakan alur kerja yang mulus dari penjadwalan hingga pelaksanaan rapat.

Selain untuk komunikasi dan penjadwalan, Outlook juga menjadi kanal krusial untuk pengiriman aset teknis dan penerimaan notifikasi otomatis dari server. Tim DBA sering menggunakan email untuk mengirim deployment packages, skrip SQL, atau laporan analisis performa yang terperinci kepada tim terkait. Lebih penting lagi, Outlook berfungsi sebagai titik akhir untuk sistem peringatan dini. Laporan notifikasi otomatis mengenai kegagalan sinkronisasi basis data, seperti pada proses *log shipping* (*database logshipping not sync*), dikonfigurasikan untuk dikirim sebagai email prioritas tinggi ke tim DBA. Hal ini memastikan bahwa setiap isu kritis segera mendapatkan perhatian dan dapat ditindaklanjuti dengan cepat.

Secara keseluruhan, Microsoft Outlook berperan sebagai pusat komando untuk komunikasi yang terstruktur dan terdokumentasi. Integrasinya yang erat dengan ekosistem Microsoft 365 lainnya menjadikan Outlook alat yang tak tergantikan dalam menjaga profesionalisme, mengatur waktu, dan memastikan responsivitas tim terhadap berbagai kebutuhan bisnis maupun teknis di lingkungan kerja.

3.8. Aigoo!

Aigoo! adalah aplikasi internal yang dikembangkan secara khusus oleh PT Adicipta Inovasi Teknologi (AdIns) untuk berfungsi sebagai sistem manajemen talenta dan kinerja sumber daya manusia (AdIns, 2024). Aplikasi ini dirancang untuk beralih dari sekadar penilaian tahunan tradisional menjadi sebuah platform yang mendukung pengembangan berkelanjutan. Fitur unggulannya adalah mekanisme penilaian kinerja 360 derajat, di mana kontribusi seorang karyawan tidak hanya dievaluasi oleh atasan, tetapi juga oleh rekan sejawat (*peers*) dan bawahan (*subordinates*), memberikan pandangan yang holistik dan seimbang terhadap kompetensi dan perilaku kerja.

Hasil dari penilaian komprehensif ini menjadi dasar bagi berbagai inisiatif pengembangan SDM. Data yang terkumpul digunakan secara strategis untuk merencanakan program pengembangan kompetensi yang dipersonalisasi, merekomendasikan pelatihan yang relevan, dan mengidentifikasi kandidat internal untuk promosi jabatan atau suksesi kepemimpinan. Selain modul penilaian, Aigoo! juga memfasilitasi proses penetapan tujuan (*goal setting*) tahunan, di mana setiap karyawan dapat menyelaraskan target individunya dengan tujuan strategis tim dan perusahaan. Adanya modul untuk memberikan *feedback* secara kontinu juga mendorong budaya keterbukaan dan perbaikan yang berkelanjutan.

Secara strategis, Aigoo! berfungsi sebagai alat analitik bagi manajemen dan tim HR. *Dashboard* yang tersedia menampilkan berbagai metrik kinerja, baik pada level individu, tim, maupun departemen, memungkinkan para pemimpin untuk memantau progres dan membuat keputusan berbasis data. Dengan kemampuannya untuk berintegrasi dengan sistem HRIS lainnya di perusahaan, Aigoo! memastikan bahwa data penilaian kinerja dapat mengalir dengan lancar untuk mendukung proses administrasi HR lainnya, seperti penentuan bonus atau kenaikan gaji, menciptakan ekosistem HR yang terpadu dan efisien.

3.9. Cherry

Cherry adalah aplikasi *Human Resources Information System* (HRIS) yang diimplementasikan di PT AdIns untuk mengelola seluruh aspek administrasi kepegawaian yang bersifat transaksional dan operasional (AdIns, 2024). Sistem ini menjadi tulang punggung bagi departemen HR untuk memastikan semua proses harian berjalan secara akurat, efisien, dan sesuai dengan kebijakan perusahaan serta regulasi pemerintah. Cherry mengotomatisasi berbagai tugas administratif yang sebelumnya dilakukan secara manual, sehingga mengurangi potensi *human error* dan membebaskan waktu tim HR untuk fokus pada inisiatif yang lebih strategis.

Sistem ini terdiri dari beberapa modul utama yang saling terintegrasi. Modul Manajemen Absensi menjadi salah satu yang paling krusial, dengan kemampuan mencatat kehadiran karyawan melalui berbagai metode seperti pemindai sidik jari (*fingerprint*), *check-in/out* melalui aplikasi seluler, hingga alur persetujuan manual oleh atasan untuk kasus-kasus khusus. Modul Pengelolaan Cuti dan Izin mendigitalisasi seluruh proses pengajuan, mulai dari permintaan oleh karyawan, persetujuan oleh manajer, hingga pencatatan oleh tim HR. Sementara itu, modul

Payroll & Benefit secara otomatis menghitung komponen gaji, termasuk upah pokok, tunjangan, lembur, potongan pajak (PPh 21), dan iuran BPJS, memastikan pembayaran yang tepat waktu dan akurat.

Kekuatan Cherry tidak hanya terletak pada fungsionalitasnya, tetapi juga pada kemampuannya untuk berintegrasi secara mulus dengan aplikasi HR lainnya seperti Aigoo!. Integrasi ini menciptakan sebuah ekosistem HR yang terpusat dan terstandarisasi. Sebagai contoh, data hasil penilaian kinerja dari Aigoo! dapat secara otomatis disinkronkan ke Cherry untuk menjadi dasar perhitungan bonus atau insentif dalam modul *payroll*. Dengan demikian, seluruh siklus hidup karyawan, mulai dari rekrutmen (*recruitment*) hingga pemberhentian (*offboarding*), dapat dikelola dalam satu platform terpadu, memastikan konsistensi data dan efisiensi proses di seluruh organisasi.

Bab IV
Deskripsi Data dan Hasil Kerja Lapangan

4.1. Pelatihan dan Pengembangan Diri

Selama Praktik Kerja Lapangan di PT Adicipta Inovasi Teknologi, kegiatan ini tidak hanya terlibat dalam tugas-tugas teknis, tetapi juga diwajibkan untuk mengikuti serangkaian pelatihan di luar bidang IT. Pelatihan ini bertujuan untuk membekali setiap karyawan, termasuk peserta magang, dengan pemahaman bisnis yang komprehensif, yang pada akhirnya akan mendukung kualitas pekerjaan teknis yang dilakukan.

Sebagai seorang DBA, memahami konteks bisnis dari data yang dikelola adalah hal yang krusial. Pelatihan ini memberikan wawasan mendalam mengenai industri *multifinance*, yang merupakan domain utama klien AdIns. Berikut adalah topik-topik pelatihan yang telah diikuti:

1. *Introduction to Multifinance (MF)*
2. *Basic Finance*
3. *CONFINS Overview*
4. *Tax Calculation*
5. *Introduction to CONFINS Foundation*
6. *Loan Origination System (LOS) part 1: Supplier and Customer*
7. *Loan Origination System (LOS) part 2: MoU and New Application*
8. *Loan Origination System (LOS) part 3: New Application*
9. *Loan Origination System (LOS) part 4: New Application*
10. *Loan Origination System (LOS) part 5: New Application, Credit Approval*
11. *Loan Origination System (LOS) part 6: Agreement Activation, Golive, and Additional process*
12. *Account Maintenance (AccMnt) part 1: Setting, Receive Form*

13. *Account Maintenance (AccMnt) part 2: Cashier Transaction*
14. *Account Maintenance (AccMnt) part 3: Suspend Transaction, Prepaid Transaction and Refund*
15. *Account Maintenance (AccMnt) part 4: Posted Date Check, and Wave Transaction*
16. *Account Maintenance (AccMnt) part 5: Non-Accrual, Write Off, and Auto Debit*

Selain pelatihan bisnis, dilakukan juga pelatihan teknis intensif selama dua minggu yang berfokus pada *Microsoft SQL Server*. Pelatihan ini bertujuan untuk memperdalam pengetahuan dan keterampilan yang menjadi fondasi utama dalam menjalankan tugas sebagai DBA. Materi pelatihan mencakup:

1. T-SQL Lanjutan: Penulisan kueri kompleks, penggunaan *window functions*, dan logika transaksional.
2. Strategi *Indexing*: Pemahaman mendalam tentang arsitektur B-Tree, perbedaan antara *clustered* dan *non-clustered index*, serta desain *covering index*.
3. Analisis *Execution Plan*: Teknik untuk membaca dan menginterpretasi *graphical execution plan*, mengidentifikasi operator berbiaya tinggi, dan memahami aliran data.
4. Manajemen Fragmentasi dan Statistik: Pentingnya mengatur dan melihat kondisi fragmentasi index dan statistik yang mutakhir dan cara memperbaruinya secara efektif.

Melalui pelatihan ini, mahasiswa - mahasiswa yang sedang melakukan magang dapat memahami alur bisnis, istilah-istilah kunci, dan pentingnya setiap data dalam sistem CONFINS. Pengetahuan ini sangat membantu dalam melakukan *query tuning*, karena dapat membuat keputusan optimasi yang tidak hanya benar secara teknis, tetapi juga relevan dengan kebutuhan bisnis. Pemahaman alur bisnis *multifinance* menyelaraskan pemahaman *query tuning* dengan kebutuhan aplikasi CONFINS, seperti mempercepat proses kredit.

4.2. Metodologi Pelaksanaan Tuning

Di PT Adicipta Inovasi Teknologi (AdIns) semua proses pengembangan, pengujian, dan pemeliharaan perangkat lunak dijalankan dalam *environment* yang tertata rapi dan terbagi menjadi beberapa level. Struktur ini penting agar kualitas, stabilitas, dan keandalan sistem benar-benar terjamin sebelum akhirnya digunakan oleh klien. Setiap environment memiliki fungsi yang spesifik dalam siklus pengembangan perangkat lunak (SDLC) untuk memastikan fitur baru atau perbaikan sistem berjalan optimal, baik dari sisi teknis maupun bisnis.

Environment pertama adalah *Production* (PROD), yaitu lingkungan utama yang langsung digunakan klien multifinance untuk operasional bisnis sehari-hari, seperti menjalankan aplikasi *CONFINS*. Karena bersentuhan langsung dengan aktivitas klien, stabilitas menjadi prioritas utama. Segala bentuk perubahan, baik itu optimas kueri, penambahan indeks, maupun perubahan struktur *database*, hanya dapat dilakukan setelah melewati persetujuan tim *DOCs* dan diuji secara menyeluruh di environment lain. PROD juga diawasi terus-menerus dengan tools monitoring seperti *SQL Server Profiler* dan *Extended Events* untuk mendeteksi anomali performa atau penggunaan resource yang melonjak. Apabila muncul keluhan performa dari klien, tim *DBA* tidak serta-merta memperbaiki langsung di *PROD*, melainkan akan melakukan investigasi lebih lanjut di environment pengujian.

Kemudian ada environment *Pre-Production* (PREPROD) yang menjadi tahap uji akhir sebelum perubahan dibawa ke PROD. Lingkungan ini digunakan oleh klien tertentu maupun tim internal untuk memvalidasi fitur baru, bug fix, atau optimasi performa dalam kondisi yang hampir sama dengan PROD. Basis data di PREPROD umumnya hasil replikasi dari PROD (dengan data yang telah dianonimkan untuk menjaga privasi), sehingga hasil uji sangat mendekati real case. Misalnya, sebelum fitur baru pada *Loan Origination System* (LOS) dirilis, seluruh proses pengajuan kredit sampai aktivasi agreement akan disimulasikan di PREPROD untuk memastikan sistem berjalan lancar dan performanya stabil.

Selanjutnya terdapat *environment Quality Assurance/User Acceptance Testing* (QA/UAT) yang dirancang untuk pengujian fungsional dan performa. Lingkungan ini dihandle tim *QA* internal, terkadang juga melibatkan klien untuk

UAT. QA/UAT dibuat menyerupai kondisi PROD, baik dari sisi konfigurasi hardware maupun volume data, supaya pengujian lebih valid. Tim QA akan memastikan fungsi aplikasi seperti perhitungan pajak di CONFINS sudah tepat dan kueri laporan keuangan berjalan dalam waktu yang wajar. Di tahap ini juga dilakukan pemeriksaan execution plan secara detail untuk mencari potensi masalah seperti *index scan* mahal, *key lookup* berlebihan, atau *cardinality estimation* yang tidak akurat. Misalnya, jika ditemukan kueri pada modul *Tax Calculation* yang memicu *table scan*, tim DBA akan memberi rekomendasi *indexing* tambahan sebelum naik ke PREPROD.

Terakhir adalah environment *Development* (DEV) atau biasa disebut juga *database lokal* AdIns untuk testing. Lingkungan ini menjadi tempat para developer dan DBA bereksperimen, menulis kode, serta melakukan pengujian awal. DEV sangat fleksibel karena bisa bebas melakukan perubahan objek database tanpa khawatir mengganggu operasional. Di sinilah biasanya DBA melakukan *testing* tuning kueri, mencoba membuat atau menghapus indeks, memodifikasi struktur tabel, hingga mengecek *execution plan* untuk melihat bagaimana query plan terbentuk. Hal ini memudahkan *troubleshooting* performa sebelum solusi dibawa ke QA/UAT.

Dengan pembagian environment seperti ini, pengembangan di AdIns berjalan lebih sistematis dan terukur, sehingga setiap perubahan telah terverifikasi dari sisi fungsi maupun performa sebelum akhirnya diterapkan di lingkungan produksi klien. Sejalan dengan lingkungan kerja terstruktur di PT Adicipta Inovasi Teknologi yang mencakup lingkungan Production (PROD), Pre-Production (PREPROD), *Quality Assurance/User Acceptance Testing* (QA/UAT), dan *Development* (DEV), proses penanganan masalah performa di PT AdIns mengikuti alur kerja yang terstruktur, mulai dari identifikasi masalah hingga implementasi solusi. Ada dua Langkah utama dalam proses ini seperti identifikasi masalah dan eskalasi tiket, dan proses tuning oleh DBA.

Masalah performa pada sistem umumnya pertama kali terdeteksi melalui keluhan yang disampaikan oleh pengguna akhir, dalam hal ini klien multifinance, atau dari tim internal yang mengamati adanya anomali pada sistem. Gejala yang paling sering dilaporkan antara lain respon aplikasi yang lambat ketika pengguna

berinteraksi dengan antarmuka, waktu tunggu yang terasa panjang saat mengakses menu atau fitur tertentu, serta proses transaksi yang berjalan lebih lama dari biasanya. Selain itu, indikator lain yang sering menjadi pemicu investigasi adalah meningkatnya durasi proses batch, misalnya waktu eksekusi *End of Day* (EOD) yang tercatat jauh lebih lama dibanding periode bulan sebelumnya.

Semua keluhan dan indikasi masalah ini kemudian dicatat secara formal ke dalam tiket pada sistem Jira, yang menjadi sentral manajemen isu di perusahaan. Tiket awalnya akan diterima dan divalidasi oleh tim DOCs yang bertanggung jawab memastikan bahwa keluhan memang terkait dengan masalah performa dan bukan isu aplikasi di sisi front-end atau kesalahan prosedur pengguna. Setelah tervalidasi sebagai persoalan yang berakar pada basis data, tiket akan dieskalasi dan dialokasikan ke tim DBA untuk dilakukan analisis lebih lanjut. Dengan pola penanganan yang terstruktur seperti ini, penelusuran akar masalah dapat dilakukan lebih cepat dan solusi optimasi dapat segera diterapkan agar sistem kembali stabil dan responsif.

Setelah menerima tiket terkait kelambatan sistem, DBA akan menjalankan serangkaian langkah analisis dan optimasi yang terstruktur. Langkah pertama dimulai dengan melakukan monitoring session, yaitu memeriksa kondisi server secara real-time untuk melihat aktivitas yang sedang berlangsung. Pada tahap ini, prosedur sistem seperti sp_who2 digunakan untuk mengidentifikasi sesi aktif, statusnya (running, runnable, suspended), serta yang paling penting memeriksa apakah ada proses yang mengalami blocking. Jika ditemukan blocking, kolom BlkBy akan menunjukkan SPID (Server Process ID) dari sesi yang menyebabkan blocking tersebut. Dengan informasi ini, DBA dapat langsung melacak kueri pemblokir dan melakukan tindakan yang diperlukan. Sebagai ilustrasi, pada Tabel 4.1 tampak bahwa SPID 58 berada dalam status suspended karena diblokir oleh SPID 62 yang sedang menjalankan operasi UPDATE pada database CONFINS_PROD.

Tabel 4. 1 Contoh Output sp_who2

SPID	Status	Login	HostName	Blk By	DBName	CMD	CPU Time	Disk IO
58	suspen-	app_us	APP_SRV01	62	CONFINS_	SELECT	15.23	120.45

	ded	er			PROD			
62	running	app_user	APP_SRV01	.	CONFINS_PROD	UPDATE	89.45	250.3

Langkah berikutnya adalah melakukan analisis indeks untuk memastikan kesehatan objek basis data yang terlibat (Choi dkk., 2021). Tahap ini meliputi pemeriksaan fragmentasi indeks untuk mengetahui sejauh mana struktur indeks mengalami split atau tidak teratur, serta memverifikasi apakah statistik *database* masih mutakhir. Statistik yang usang dapat menyebabkan SQL Server membuat rencana eksekusi yang kurang optimal. Oleh karena itu, update statistik menjadi penting agar estimasi cardinality tetap akurat. Langkah ini sering kali menjadi titik awal penting dalam optimasi performa, seperti yang juga ditekankan dalam laporan PKL rekan satu tim, Kevin, yang menitikberatkan pada perbaikan strategi indexing.

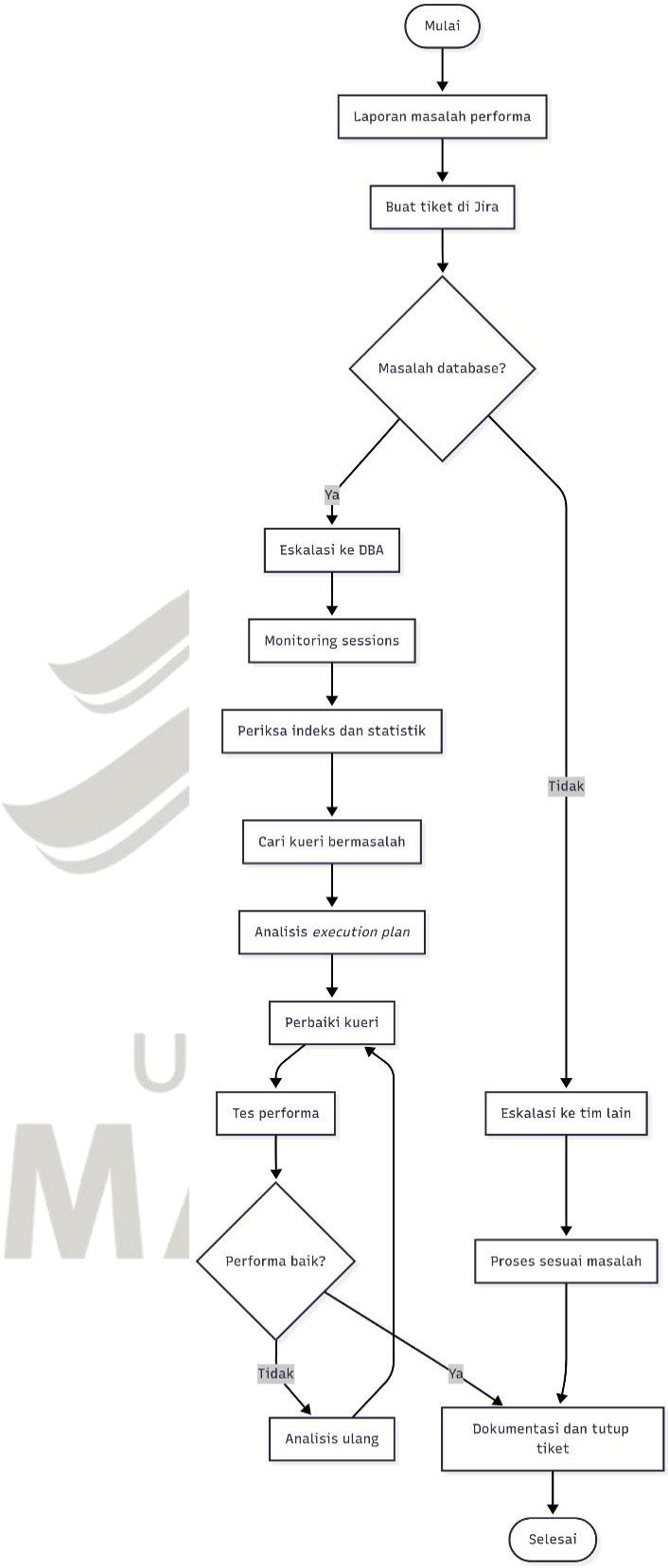
Setelah itu, dilakukan identifikasi kueri-kueri bermasalah menggunakan alat diagnostik seperti *Dynamic Management Views* (DMV), salah satunya sys.dm_exec_query_stats. DMV ini memungkinkan DBA untuk mengidentifikasi kueri mana yang paling banyak mengonsumsi CPU atau logical reads. Kueri-kueri ini kemudian menjadi kandidat utama untuk ditinjau lebih lanjut. *Execution plan* dari kueri tersebut akan dianalisis mendetail guna menemukan penyebab *bottleneck*, misalnya *table scan* atau *index scan* yang tidak perlu, *key lookup* yang mahal, hingga estimasi *cardinality* yang keliru yang menyebabkan *operator join* atau *sort* bekerja lebih berat dari yang semestinya.

Setelah akar masalah teridentifikasi, DBA akan melakukan query tuning. Fokus utama dari tahap ini adalah memodifikasi sintaks kueri agar lebih efisien. Salah satu prinsip penting dalam query tuning adalah memastikan kueri ditulis secara *SARGable* (*Search Argument Able*). Artinya, kondisi dalam WHERE atau JOIN diekspresikan sedemikian rupa sehingga SQL Server dapat memanfaatkan indeks yang tersedia secara optimal. Contohnya, menulis WHERE OrderDate >= '2025-01-01' lebih baik daripada WHERE YEAR(OrderDate) = 2025 karena yang terakhir memaksa SQL Server menghitung fungsi pada setiap baris sehingga tidak dapat langsung memanfaatkan seek pada indeks OrderDate.

Terakhir, hasil optimasi akan diuji ulang dengan menjalankan kueri yang telah dituning dan membandingkan metrik performa seperti waktu eksekusi, jumlah

logical reads, serta *resource usage* sebelum dan sesudah tuning. Semua temuan dan perubahan kemudian didokumentasikan dalam tiket Jira, lengkap dengan *package* yang berisi *README*, *step deployment*, dan *file deployment*, untuk memastikan implementasi dapat dilakukan dengan tertib dan mudah ditelusuri di masa depan.

Deskripsi alur kerja penanganan masalah performa di PT AdIns dapat dilihat pada Gambar 4.1, yang menyajikan *flowchart* proses tuning dan optimasi.



Gambar 4.1 Flowchart Tuning Optimizing

Berdasarkan alur kerja yang dijelaskan dalam Gambar 4.1, langkah-langkah analisis dan optimasi selanjutnya akan dibahas secara rinci pada subbab berikut.

4.2.1. Analisis Kesehatan Indeks

Sebelum menganalisis kueri, penting untuk memastikan fondasi indeks dan statistik dalam kondisi baik. Salah satu isu yang sering memengaruhi performa adalah fragmentasi indeks yaitu kondisi ketika urutan logis halaman-halaman indeks tidak lagi selaras dengan urutan fisiknya pada *disk*. Situasi ini umumnya timbul akibat aktivitas *Data Manipulation Language* (DML) seperti *INSERT*, *UPDATE*, dan *DELETE* yang terjadi secara terus-menerus. Misalnya, operasi *INSERT* yang menambahkan data baru ke tabel dan indeks akan memicu pembentukan halaman baru yang dapat tidak berurutan secara fisik. Seiring waktu, hal ini menyebabkan fragmentasi yang tinggi dan membuat SQL Server perlu melakukan lompatan acak antar halaman saat membaca data, sehingga memperlambat proses scan.

Untuk itu, DBA perlu memastikan tingkat kesehatan indeks dengan menjalankan query menggunakan fungsi dinamis sys.dm_db_index_physical_stats. Contoh kueri yang digunakan untuk memeriksa tingkat fragmentasi dapat dilihat pada Gambar 4.2.

sql Collapse Wrap Copy

```sql
SELECT
    OBJECT_NAME(ips.object_id) AS TableName,
    i.name AS IndexName,
    ips.index_type_desc AS IndexType,
    ips.avg_fragmentation_in_percent AS FragmentationPercent,
    ips.page_count AS PageCount
FROM sys.dm_db_index_physical_stats(DB_ID('AdventureWorks'), NULL, NULL, NULL, 'LIMITED')
JOIN sys.indexes i ON ips.object_id = i.object_id AND ips.index_id = i.index_id
WHERE ips.avg_fragmentation_in_percent > 10
ORDER BY ips.avg_fragmentation_in_percent DESC;
```

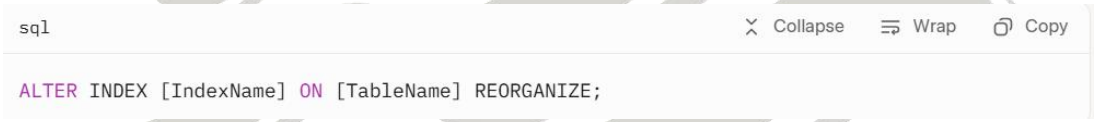
Gambar 4.2 Check Fragmentation Index Query

Query tersebut akan menghasilkan daftar indeks lengkap dengan rata-rata tingkat fragmentasi (avg_fragmentation_in_percent) untuk setiap indeks. Berdasarkan praktik standar yang banyak digunakan di industri fragmentasi antara 5% hingga 30% dianjurkan untuk ditangani dengan *REORGANIZE* dan fragmentasi di atas 30% (khususnya jika melebihi 90%) perlu dilakukan tindakan REBUILD agar struktur halaman tersusun ulang secara optimal.

Sebagai ilustrasi, apabila terdapat indeks *NONCLUSTERED* pada tabel SalesOrderDetail dengan tingkat fragmentasi sebesar 45%, maka tindakan *REORGANIZE* dapat digunakan untuk menata ulang halaman secara ringan tanpa membangun ulang seluruh struktur indeks. Namun jika fragmentasinya telah mencapai 95%, maka prosedur *REBUILD* harus dijalankan menggunakan perintah seperti pada Gambar 4.3.

Gambar 4.3 Rebuild Index Query

Sedangkan untuk fragmentasi sedang (misalnya antara 10% sampai 50%), tindakan cukup dengan *REORGANIZE* atau seperti pada Gambar 4.4.



Gambar 4.4 Reorganize Index Query

Perlu dipahami bahwa perintah *SELECT* standar (misalnya 'SELECT TOP 1000 * FROM Tabel') tidak secara langsung meningkatkan fragmentasi. Fragmentasi terjadi akibat aktivitas penulisan data seperti *INSERT* dan *UPDATE* dalam jumlah besar. Namun demikian, *SELECT* pada tabel dengan tingkat fragmentasi tinggi akan berjalan lebih lambat karena *engine database* harus membaca halaman yang letaknya tidak berurutan di disk.

Dengan pemantauan rutin terhadap fragmentasi dan penjadwalan maintenance seperti *REORGANIZE* atau *REBUILD* sesuai ambang batas yang direkomendasikan, performa query terutama yang melibatkan operasi scan dapat tetap terjaga optimal, sekaligus meminimalkan risiko penurunan kinerja sistem di masa mendatang. Contoh *output* dapat dilihat pada Tabel 4.2.

Tabel 4. 2 Output Index Fragmentation

TableName	SalesOrderDetail
IndexName	IX_SalesOrderDetail
IndexType	NONCLUSTERED
FragmentationPercent	45,3
PageCount	500

Selain indeks, statistik database berfungsi memberikan informasi distribusi data yang digunakan oleh Query Optimizer untuk menentukan rencana eksekusi (execution plan) yang paling efisien. Statistik yang usang atau tidak mutakhir dapat menyebabkan Query Optimizer salah memperkirakan jumlah baris (cardinality estimate), sehingga memilih execution plan yang tidak optimal. Hal ini dapat mengakibatkan penggunaan sumber daya yang lebih tinggi dan waktu eksekusi query yang lebih lama.

Untuk memeriksa apakah statistik masih akurat dan relevan, DBA dapat menjalankan kueri seperti yang ditunjukkan pada Gambar 4.5 Update Statistic Query, yang berfungsi untuk menampilkan informasi mengenai waktu terakhir statistik diperbarui serta jumlah perubahan data atau modifikasi yang terjadi sejak pembaruan terakhir. Informasi ini sangat penting karena akurasi statistik sangat memengaruhi kualitas rencana eksekusi yang dipilih oleh Query Optimizer. Jika statistik tidak diperbarui dalam jangka waktu yang lama atau jumlah modifikasi yang terjadi cukup signifikan, maka risiko kesalahan estimasi cardinality meningkat, yang pada akhirnya dapat menyebabkan pemilihan *execution plan* yang tidak efisien. Dengan memantau dan memperbarui statistik secara berkala, DBA dapat menjaga performa sistem tetap optimal dan memastikan bahwa query yang dijalankan memperoleh rencana eksekusi yang paling tepat. Sebagai ilustrasi, hasil dari kueri tersebut dapat dilihat pada Tabel 4.3, yang menunjukkan data riwayat pembaruan dan indikator potensi perlunya pembaruan statistik lebih lanjut.

```sql
SELECT
    OBJECT_NAME(s.object_id) AS TableName,
    s.name AS StatisticName,
    STATS_DATE(s.object_id, s.stats_id) AS LastUpdated,
    sp.rows AS RowCount,
    sp.modification_counter AS Modifications
FROM sys.stats s
JOIN sys.objects o ON s.object_id = o.object_id
CROSS APPLY sys.dm_db_stats_properties(s.object_id, s.stats_id) sp
WHERE o.type = 'U' AND sp.modification_counter > 1000
ORDER BY sp.modification_counter DESC;
```

Gambar 4.5 Update Statistic Query

Tabel 4. 3 Update Statistic Output

TableName	StatisticName	LastUpdated	RowCount	Modifications
SalesOrderDetail	_WA_Sys_00000001	2023-01-01 10:00	121.317	5.000

Jika nilai modification_counter (jumlah perubahan sejak terakhir kali statistik diperbarui) cukup tinggi, ini menandakan adanya potensi statistik yang tidak lagi akurat. Umumnya, jika jumlah modifikasi mendekati atau melebihi 20% dari total baris, maka update statistik sangat dianjurkan. Solusi yang dapat diterapkan adalah memperbarui statistik secara manual menggunakan perintah 'UPDATE STATISTICS [SalesOrderDetail] WITH FULLSCAN'.

Opsi *WITH FULLSCAN* memastikan SQL Server membaca seluruh data dalam tabel untuk menghasilkan histogram statistik yang paling akurat. Dengan memastikan statistik selalu mutakhir, *Query Optimizer* dapat membuat keputusan yang lebih tepat terkait pemilihan *execution plan*, sehingga performa *query* tetap optimal.

4.2.2. Analisis Execution Plan

Execution plan adalah alat utama untuk menemukan *bottleneck* dalam sebuah kueri (Nevarez, 2022). Dalam execution plan, terdapat sejumlah komponen penting yang menjadi dasar analisis mengapa sebuah query berjalan lambat atau efisien. Salah satu yang utama adalah "*Seek Keys*," yakni informasi tentang kolom mana

yang digunakan sebagai kunci pencarian ketika operator *Index Seek* dijalankan. *Seek Keys* memperlihatkan kondisi predikat yang diterapkan langsung pada indeks, contohnya ketika *Index Seek* dijalankan untuk mengeksekusi filter DepartmentID = 1 seperti pada Gambar 4.6, maka *Seek Keys* akan menampilkan kolom DepartmentID yang dibandingkan dengan nilai literal 1 yang tercatat sebagai "*Scalar Operator*." Scalar Operator ini mengindikasikan nilai konstan atau ekspresi yang digunakan SQL Server dalam pencarian tersebut.

sql Collapse Wrap Copy

```sql
SELECT e.FirstName, e.LastName, d.DepartmentName
FROM HR.Employees e
JOIN HR.Departments d ON e.DepartmentID = d.DepartmentID
WHERE e.DepartmentID = 1;
```

Gambar 4.6 Studi Kasus

Selain itu, terdapat "*Output List*" yang menampilkan kolom-kolom apa saja yang diambil dari hasil *Index Seek* tersebut. Misalnya, pada kasus sebuah *query* yang hanya memiliki indeks *non-clustered* di kolom DepartmentID, *Output List* mungkin hanya mengembalikan EmployeeID. Namun jika *query* membutuhkan FirstName dan LastName, *SQL Server* harus melakukan operasi tambahan berupa *Key Lookup* ke *clustered index* untuk melengkapi data yang dibutuhkan. Hal ini terjadi karena indeks non-*clustered* tidak memiliki kolom-kolom tersebut, sehingga optimizer memilih untuk mengambilnya langsung dari *clustered index* menggunakan nilai EmployeeID yang ditemukan sebelumnya.

Sebagai ilustrasi, pada *database* CorporateERPSystem terdapat *query* yang menampilkan daftar karyawan berdasarkan DepartmentID dan menampilkan FirstName serta LastName. *Execution plan* menunjukkan tahap pertama berupa *Index Seek* pada indeks IX_Employees_DepartmentID dengan *Seek Keys* [CorporateERPSystem].[HR].[Employees].DepartmentID = Scalar Operator(1). *Output List* hanya berisi EmployeeID. Karena FirstName dan LastName tidak terdapat dalam indeks tersebut, *SQL Server* kemudian melakukan *Key Lookup* untuk setiap EmployeeID yang ditemukan guna mengambil data nama karyawan dari *clustered index*. Proses dua tahap ini memperkenalkan *overhead* yang signifikan, terutama bila jumlah baris hasil seek cukup besar.

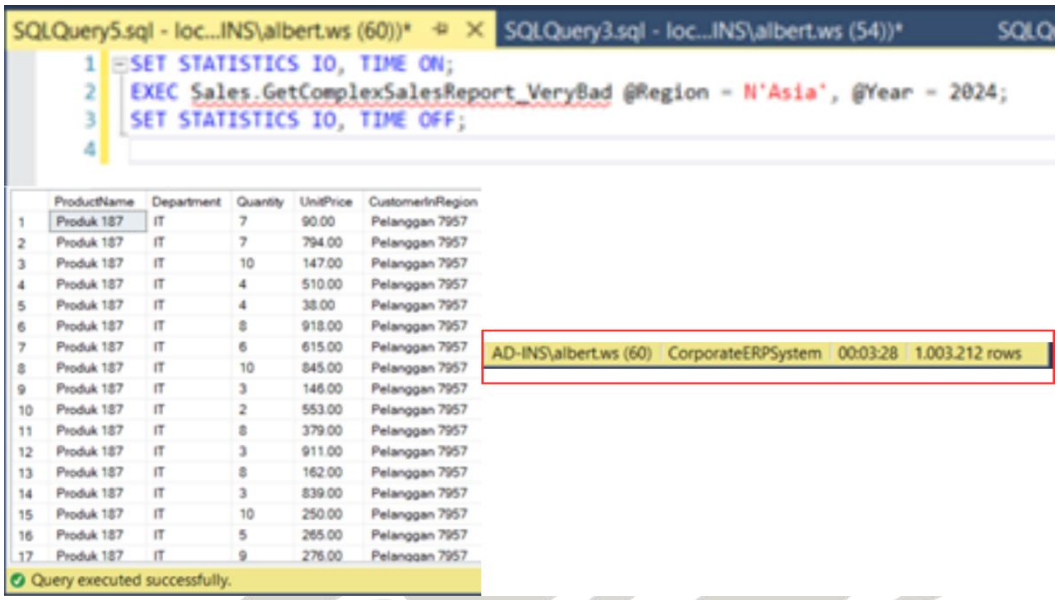
Solusi paling efektif dalam kasus seperti ini adalah dengan membuat *covering index*, yaitu menambahkan kolom FirstName dan LastName ke dalam indeks menggunakan klausa *INCLUDE*. Dengan demikian, data yang dibutuhkan *query* dapat langsung diambil pada satu langkah *Index Seek* tanpa memerlukan *Key Lookup* tambahan. Contohnya, dengan membuat indeks 'CREATE NONCLUSTERED INDEX IX_Employees_DepartmentID_Covering ON HR.Employees (DepartmentID) INCLUDE (FirstName, LastName);', *optimizer* dapat langsung memenuhi permintaan *query* hanya dari satu indeks, mengurangi *logical reads* dan meningkatkan kecepatan eksekusi *query* secara signifikan. Dengan strategi ini, DBA tidak hanya memperbaiki performa, tetapi juga mengurangi beban I/O pada server yang pada akhirnya menjaga kesehatan sistem secara keseluruhan.

4.2.3. Analisis Tuning Kueri

Untuk mendemonstrasikan metodologi tuning secara praktis, sebuah studi kasus komprehensif dirancang dengan membuat basis data baru bernama CorporateERPSystem. Lingkungan ini sengaja disiapkan untuk mensimulasikan sistem ERP skala besar, lengkap dengan skema untuk departemen yang berbeda seperti HR, Sales, dan Production. Tabel-tabel seperti Employees, Products, Customers, dan SalesHistory dibuat untuk mencerminkan struktur data yang realistis. Agar analisis performa menjadi signifikan dan relevan dengan tantangan industri, sebuah stored procedure bernama dbo.PopulateERPData dieksekusi untuk mengisi tabel Sales.SalesHistory dengan 5 juta baris data transaksi. Proses ini memastikan bahwa setiap bottleneck performa yang muncul bukanlah akibat dari volume data yang kecil, melainkan cerminan dari masalah nyata yang dihadapi pada sistem produksi.

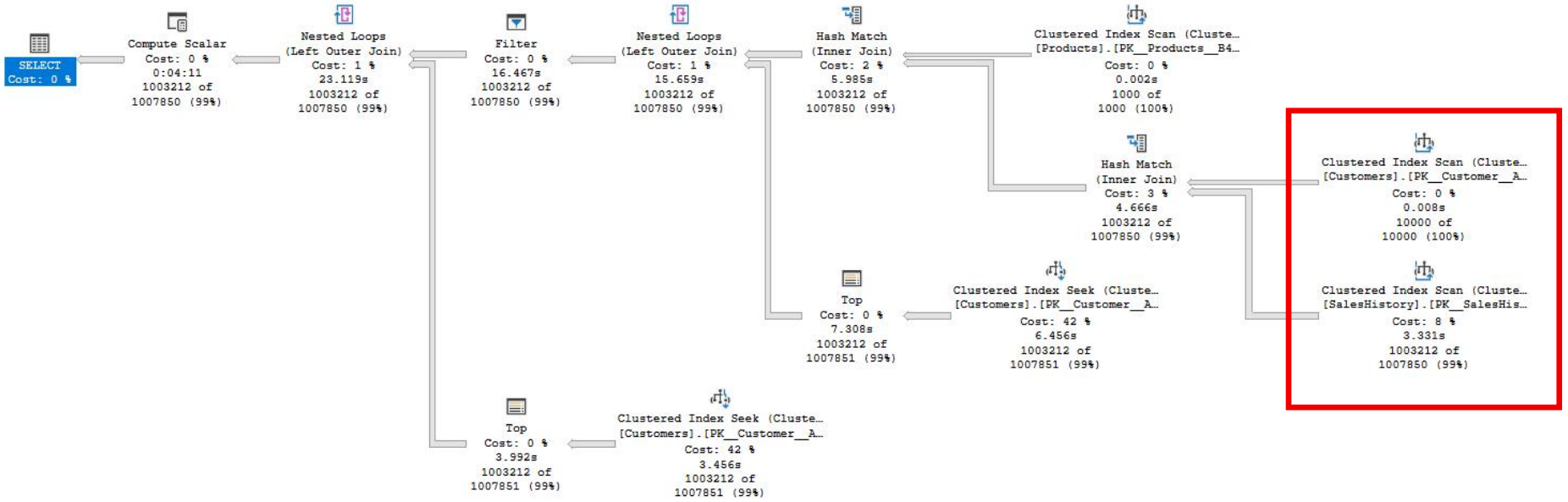
Setelah lingkungan disiapkan, sebuah stored procedure laporan bernama Sales.GetDepartmentSalesReport_VeryBad dibuat sebagai target optimasi. Prosedur ini sengaja dirancang dengan beberapa anti-pattern performa yang umum ditemui. Masalah pertama adalah penggunaan SELECT * dalam sebuah *Common Table Expression* (CTE), yang tidak efisien karena mengambil semua kolom, bukan hanya yang dibutuhkan. Masalah kedua adalah adanya correlated subquery untuk mengambil nama pelanggan, yang memaksa SQL Server untuk mengeksekusi pencarian terpisah untuk setiap baris yang diproses (Firmansyah dkk., 2025). Terakhir, dan yang paling signifikan, adalah penggunaan fungsi YEAR() pada

kolom SaleDate di dalam klausa WHERE, yang membuat predikat tersebut menjadi non-SARGable dan menghalangi Query Optimizer untuk menggunakan indeks yang ada pada kolom tanggal secara efisien.



Gambar 4.7 Contoh Kueri Tidak Optimal

Untuk menetapkan *baseline* performa, GetDepartmentSalesReport_VeryBad dieksekusi dengan parameter untuk departemen 'Sales' pada tahun 2024. Hasilnya sangat buruk, dengan waktu eksekusi mencapai 55 detik dan penggunaan logical reads pada tabel SalesHistory melebihi 4138336 halaman. Analisis pada Actual Execution Plan mengonfirmasi diagnosis ini terlihat jelas adanya operator *Table Scan* yang sangat mahal pada tabel SalesHistory (akibat predikat non-SARGable) dan Clustered Index Scan berulang pada tabel Customers (akibat correlated subquery).

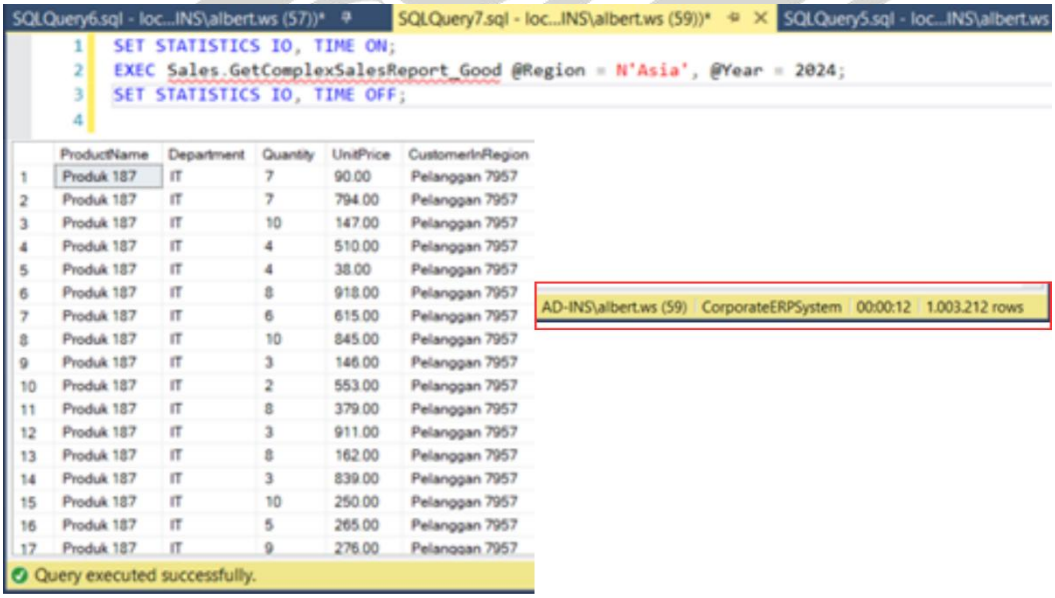


Gambar 4.8 Execution Plan Sebelum Optimasi

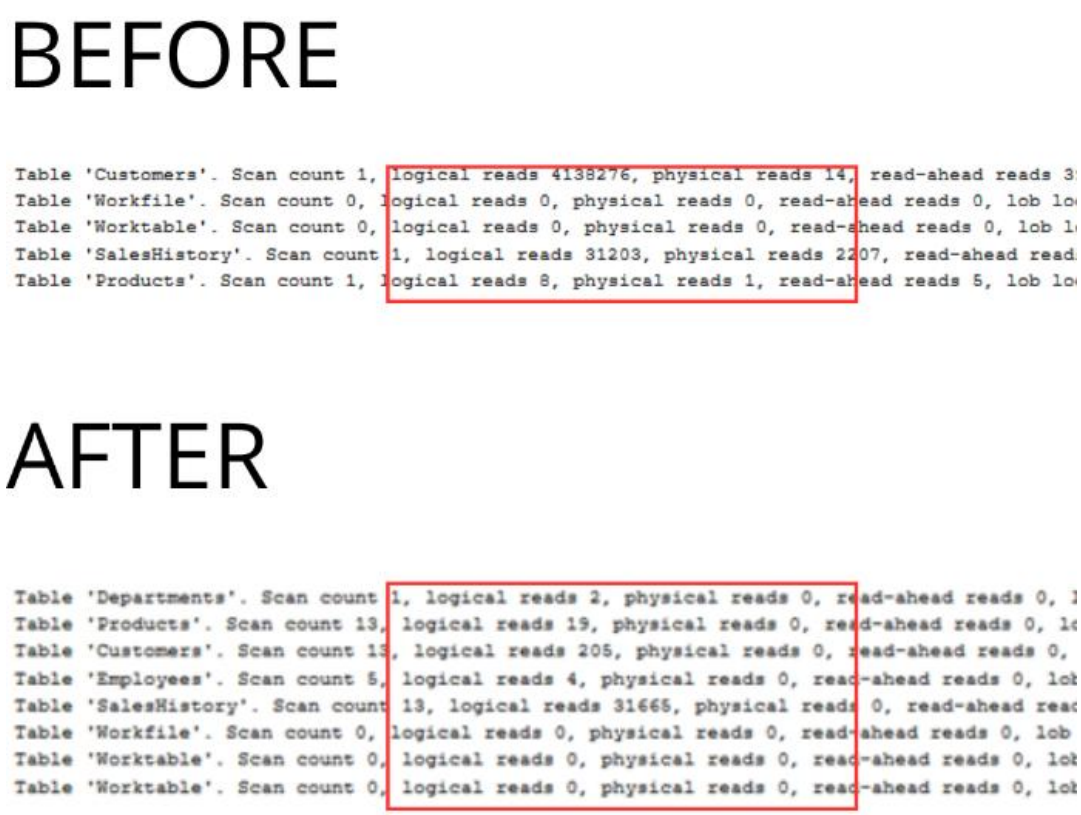
Proses validasi dan tuning kemudian dilakukan dengan membuat versi baru dari stored procedure tersebut, yaitu Sales.GetDepartmentSalesReport_Good. Implementasi solusi berfokus pada perbaikan semua anti-pattern yang teridentifikasi. Penggunaan SELECT * diganti dengan daftar kolom yang spesifik. Correlated subquery untuk nama pelanggan dihilangkan dan diganti dengan operasi JOIN langsung ke tabel Sales.Customers. Terakhir, predikat non-SARGable YEAR(s.SaleDate) = @Year ditulis ulang menjadi klausa rentang tanggal yang SARGable: s.SaleDate >= @StartDate AND s.SaleDate <= @EndDate.

```
;WITH FilteredSales AS (
    SELECT SaleID, ProductID, EmployeeID, CustomerID, Quantity, UnitPrice
    FROM Sales.SalesHistory
    WHERE SaleDate >= @StartDate AND SaleDate < @EndDate
),
```

Gambar 4.9 Optimasi Kueri



Gambar 4.10 Hasil Optimasi Kueri



Gambar 4.11 Logical Reads Before and After Tuning

Setelah stored procedure yang dioptimalkan dieksekusi dengan parameter yang sama, hasilnya menunjukkan peningkatan performa yang dramatis. Waktu eksekusi turun drastis menjadi hanya 12 detik (kurang dari 1 detik), dan *logical reads* pada tabel SalesHistory berkurang lebih dari 96% menjadi sekitar 31.895 halaman. Analisis *Execution Plan* yang baru menunjukkan bahwa operator Table Scan yang mahal telah berhasil diubah menjadi Index Seek yang sangat efisien pada kolom SaleDate, dan tidak ada lagi operasi scan berulang pada tabel Customers.

Berdasarkan hasil pengujian yang diperoleh sebelum dan sesudah optimasi stored procedure, dapat dilakukan analisis performa sebagai berikut. Sebelum dilakukan optimasi, eksekusi *stored procedure* memakan waktu sekitar 258.160 ms atau sekitar 4 menit 18 detik (tercatat completion time di 2025-07-07T16:15:36), dengan total CPU time sebesar 240.250 ms. Beban I/O pun sangat tinggi terlihat pada logical reads tabel Customers yang mencapai 4.138.336 halaman, sedangkan tabel SalesHistory tercatat 31.203 logical reads. Execution plan menunjukkan

terjadinya table scan besar-besaran serta correlated subquery yang menimbulkan nested loop mahal.

Setelah optimasi diterapkan, execution time turun drastis menjadi hanya 12.490 ms atau sekitar 12 detik (tercatat completion time di 2025-07-06T15:17:12). CPU time pun menurun signifikan ke hanya sekitar 10.316 ms. Logical reads pada tabel Customers juga turun menjadi hanya 205 halaman, sedangkan tabel SalesHistory sedikit naik menjadi 31.665 namun lebih merata akibat penggunaan join yang efisien. Hal ini terjadi karena seluruh anti-pattern seperti SELECT * dihilangkan, correlated subquery diganti menjadi explicit join, serta predicate YEAR() diganti dengan filter range yang SARGable.

Secara garis besar, proses optimasi ini berhasil menurunkan waktu eksekusi dari sekitar 3,28 menit menjadi hanya 12 detik. CPU usage berkurang lebih dari 95%, logical reads pada tabel utama juga turun drastis lebih dari 99%, yang secara keseluruhan membuat *query* jauh lebih ringan, cepat, dan hemat *resource*. Dari tabel perbandingan performa dapat disimpulkan bahwa optimasi ini sangat efektif dalam meningkatkan efisiensi *query* terhadap data skala besar. Untuk merangkum dampak kuantitatif dari proses tuning ini, metrik performa sebelum dan sesudah disajikan dalam Tabel 4.4.

Tabel 4. 4 Perbandingan Sebelum dan Sesudah Tuning

Metrik	**Sebelum Optimasi**	**Sesudah Optimasi**	**Perubahan (%)**
Waktu Eksekusi (elapsed time)	258.160 ms (3,28 menit)	12.490 ms (~12 detik)	-95,2%
Waktu CPU (CPU time)	240.250 ms	10.316 ms	-95,7%
Logical Reads (Customers)	4.138.336 halaman	205 halaman	-99,9%
Logical Reads (SalesHistory)	31.203 halaman	31.665 halaman	+1,5% (lebih merata)

Tabel 4.4 menunjukkan dampak optimasi yang signifikan, terutama pada pengurangan waktu eksekusi dan pemakaian CPU. Logical reads pada tabel Customers turun drastis lebih dari 99%, menunjukkan beban I/O jauh lebih ringan,

sementara logical reads SalesHistory sedikit naik karena distribusi join yang lebih wajar dan efisien.

4.2.3.1. CTE dan Temporary Table

Tugas seorang DBA bukan hanya membuat kueri yang bisa berjalan, tetapi juga harus memastikan kueri tersebut berjalan secepat mungkin (Atma & Suhartati, 2017). Salah satu tantangan yang sering ditemui adalah saat harus menangani logika yang rumit, di mana data perlu diolah dalam beberapa tahap. Untuk ini, ada dua cara yang paling efektif di T-SQL yaitu *Common Table Expressions* (CTE) dan *Temporary Tables*. Keduanya sama-sama membuat set data sementara, tetapi cara kerja dan dampaknya ke performa sangat berbeda. Memilih alat yang salah bisa membuat kueri yang seharusnya cepat jadi sangat lambat.

CTE bisa dibilang cara yang elegan untuk menulis kueri yang rumit. Fungsinya adalah untuk memecah satu kueri panjang menjadi beberapa blok logika yang lebih kecil dan mudah dibaca, seolah-olah kita sedang bercerita. Ini sangat membantu agar kode lebih rapi dan gampang diurus di kemudian hari. Namun, di balik kelebihannya itu, CTE punya kelemahan besar dari sisi performa. Masalah utamanya adalah *Query Optimizer* tidak bisa "melihat" dengan baik berapa banyak data yang ada di dalam CTE. Akibatnya, ia sering salah tebak dan memilih rencana eksekusi (*execution plan*) yang tidak efisien, terutama jika hasil CTE tersebut digabungkan dengan tabel lain yang sangat besar. Parahnya lagi, jika kita memanggil CTE yang sama beberapa kali dalam satu kueri, SQL Server akan menjalankan ulang seluruh logika CTE itu setiap kali dipanggil. Ini jelas pemborosan sumber daya.

Di sinilah *Temporary Table* (tabel sementara, biasanya diawali dengan #) menjadi solusi yang lebih tangguh. Berbeda dengan CTE yang hanya "hidup" di dalam satu kueri, *temporary table* adalah tabel fisik sungguhan yang dibuat di dalam database tempdb. Karena statusnya sebagai tabel fisik, kita bisa melakukan banyak hal padanya, seperti membuat indeks. Kemampuan untuk menambahkan indeks pada data sementara inilah yang menjadi kunci kemenangannya. Dengan adanya indeks, proses pencarian atau penggabungan data di tahap selanjutnya bisa berjalan jauh lebih cepat karena SQL Server bisa langsung menunjuk ke data yang dicari, bukan lagi memindai seluruh isi tabel.

Untuk memudahkan pengambilan keputusan, berikut perbandingan penggunaan CTE dan Temp Table dalam Tabel 4.5. Seperti yang terlihat pada tabel, CTE cocok digunakan untuk logika yang sederhana, sekali pakai, dan saat kerapian kode menjadi prioritas. Sebaliknya, saat kita berhadapan dengan data perantara yang jumlahnya besar, perlu digunakan berulang kali, atau perlu digabungkan dengan tabel besar lainnya, *Temporary Table* adalah pilihan yang jauh lebih bijak. Baris "Kebutuhan Indeks" pada tabel menegaskan hal ini hanya *Temporary Table* yang memberikan kita kemewahan untuk membuat indeks demi mendongkrak performa. Di lingkungan kerja PT AdIns, misalnya, untuk laporan bulanan yang rumit, akan jauh lebih efisien jika data awal diolah dan disimpan dulu di *temporary table*, diberi indeks, baru kemudian digunakan untuk proses-proses selanjutnya.

Tabel 4. 5 Tabel Perbandingan CTE dan Temp Table

Kriteria	**Gunakan Common Table Expression (CTE)**	**Gunakan Temporary Table**
Penggunaan Ulang	Hasil hanya dibutuhkan satu kali dalam kueri.	Hasil dibutuhkan berulang kali dalam *stored procedure* atau *batch*.
Volume Data	Set data hasil cenderung kecil.	Set data hasil besar dan akan di-*join* dengan tabel besar lainnya.
Kebutuhan Indeks	Tidak memerlukan indeks pada hasil sementara.	Memerlukan indeks untuk optimasi JOIN atau filter.
Keterbacaan	Prioritas utama adalah menyederhanakan kueri yang kompleks atau hierarkis.	Logika multi-langkah yang kompleks perlu dipisahkan secara fisik.

Tentu, penggunaan *temporary table* juga ada konsekuensinya. Setiap kali kita membuat dan mengisi *temporary table*, ada aktivitas tulis ke tempdb. Jika terlalu banyak proses yang melakukan ini secara bersamaan, tempdb bisa menjadi kewalahan dan malah menjadi sumber masalah baru. Karena itu, pemilihan antara CTE dan *temporary table* harus didasarkan pada analisis kebutuhan di setiap kasus. Kemampuan untuk menimbang antara kerapian CTE dan kekuatan performa *Temporary Table* adalah keahlian penting untuk menjaga agar aplikasi di PT AdIns tetap berjalan cepat dan responsif.

4.2.3.2. Parameter Sniffing

Salah satu tantangan performa yang paling umum dan signifikan dalam lingkungan produksi adalah *parameter sniffing*. Parameter sniffing merupakan fitur fungsional SQL Server di mana optimizer menyimpan rencana eksekusi berdasarkan nilai parameter pertama yang digunakan, yang dapat menyebabkan performa suboptimal untuk nilai parameter yang berbeda (Nevarez, 2022). Fenomena ini bukan merupakan sebuah galat (*bug*), melainkan sebuah fitur fungsional dari mekanisme *caching* SQL Server yang dapat menimbulkan dampak negatif pada kondisi tertentu. Mekanismenya adalah, saat sebuah *stored procedure* dieksekusi untuk pertama kalinya, *Query Optimizer* akan menganalisis nilai parameter yang digunakan. Berdasarkan nilai spesifik tersebut, *optimizer* akan menghasilkan sebuah rencana eksekusi (*execution plan*) yang dianggap paling efisien, lalu menyimpan rencana tersebut di dalam *plan cache* untuk digunakan kembali pada eksekusi selanjutnya. Tujuan dari mekanisme ini adalah untuk meningkatkan efisiensi dengan mengurangi beban kompilasi kueri secara berulang.

Permasalahan serius dapat timbul ketika terdapat distribusi data yang tidak seragam (*data skew*) pada tabel yang diakses. Sebagai ilustrasi, sebuah *stored procedure* pada sistem CONFINS digunakan untuk mencari riwayat pembayaran nasabah. Jika eksekusi pertama menggunakan parameter untuk nasabah dengan kardinalitas rendah (misalnya, 10 baris transaksi), *optimizer* akan membuat rencana eksekusi yang sangat efisien untuk volume data kecil. Namun, ketika *procedure* yang sama kemudian dieksekusi untuk nasabah korporat dengan kardinalitas tinggi (misalnya, 2 juta baris transaksi), SQL Server akan menggunakan kembali rencana eksekusi yang telah tersimpan di *cache*. Konsekuensinya, rencana yang tidak sesuai tersebut akan menyebabkan degradasi performa yang signifikan, konsumsi sumber daya yang sangat tinggi, dan berpotensi menyebabkan kelambatan pada aplikasi.

Untuk mengatasi tantangan ini, terdapat beberapa strategi yang dapat diimplementasikan. Strategi yang paling umum diterapkan karena risikonya yang rendah adalah penggunaan variabel lokal. Dengan mendeklarasikan sebuah variabel di dalam *stored procedure* dan menetapkan nilai parameter input ke dalamnya, *Query Optimizer* tidak lagi dapat menginspeksi nilai parameter asli. Karena tidak mengetahui nilai spesifiknya, *optimizer* akan membuat rencana eksekusi yang lebih generik berdasarkan histogram statistik distribusi data secara keseluruhan, bukan

berdasarkan satu nilai yang bias. Pendekatan ini sering kali efektif untuk menstabilkan performa pada berbagai skenario eksekusi.

Apabila strategi variabel lokal tidak memberikan hasil yang memadai atau ketika performa optimal yang konsisten menjadi sebuah keharusan, pendekatan yang lebih eksplisit dapat digunakan. Opsi kueri OPTION (RECOMPILE) dapat ditambahkan untuk memaksa SQL Server melakukan kompilasi ulang dan membuat rencana eksekusi baru pada setiap pemanggilan. Strategi ini menjamin rencana yang selalu optimal untuk parameter yang sedang digunakan, namun dengan konsekuensi peningkatan penggunaan sumber daya CPU. Alternatif lain yang lebih terkontrol adalah OPTIMIZE FOR, yang memberikan instruksi kepada *optimizer* untuk membuat rencana yang dioptimalkan bagi nilai parameter tertentu yang dianggap paling representatif, atau OPTIMIZE FOR UNKNOWN yang menghasilkan efek serupa dengan penggunaan variabel lokal.

Pemilihan strategi di PT AdIns sangat bergantung pada analisis kasus per kasus. Untuk *stored procedure* dengan frekuensi pemanggilan tinggi, penggunaan variabel lokal merupakan pendekatan yang paling seimbang antara stabilitas dan *overhead*. Namun, untuk proses *batch* krusial seperti perhitungan akhir hari (*End of Day*) yang menuntut performa optimal, penggunaan OPTION (RECOMPILE) dapat menjadi justifikasi yang valid. Kemampuan untuk mengidentifikasi dan menerapkan solusi yang tepat untuk *parameter sniffing* merupakan kompetensi esensial bagi seorang DBA dalam menjaga stabilitas sistem di lingkungan produksi.

4.2.3.3. Analisis Wait Statistics

Metrik performa seperti waktu eksekusi dan penggunaan CPU memberikan gambaran kuantitatif mengenai hasil akhir (apa), namun tidak menjelaskan penyebab utama kelambatan (mengapa). Untuk melakukan analisis akar masalah (*root cause analysis*), diperlukan pemeriksaan yang lebih mendalam menggunakan Wait Statistics. *Wait Statistics* merupakan mekanisme diagnostik internal SQL Server yang mencatat setiap kali sebuah proses eksekusi harus berhenti sementara untuk menunggu ketersediaan sebuah sumber daya, seperti I/O disk, CPU, atau *lock*. Dengan menganalisis akumulasi waktu dan jenis penungguan (*wait types*), sumber utama *bottleneck* dapat diidentifikasi secara presisi.

Dalam studi kasus kueri GetDepartmentSalesReport_VeryBad sebelum optimasi, *execution plan* mengindikasikan adanya operator *Table Scan* yang berbiaya tinggi. Temuan ini akan terefleksikan secara langsung pada *wait statistics*. Jenis penungguan (*wait type*) yang akan mendominasi adalah PAGEIOLATCH_SH. Secara teknis, *wait type* ini mengindikasikan bahwa proses eksekusi menghabiskan sebagian besar waktunya untuk menunggu halaman data selesai dibaca dari media penyimpanan (disk) ke dalam memori (*buffer pool*). Tingginya akumulasi waktu pada *wait type* ini merupakan indikator klasik dari kueri yang terhambat oleh performa I/O (*I/O-bound*).

Setelah implementasi optimasi, di mana *Table Scan* berhasil dieliminasi dan digantikan dengan *Index Seek*, profil *wait statistics* akan mengalami transformasi signifikan. Karena kueri tidak lagi memerlukan pembacaan data ekstensif dari disk, waktu tunggu yang disebabkan oleh aktivitas I/O akan berkurang secara drastis. Akibatnya, akumulasi waktu untuk *wait type* PAGEIOLATCH_SH akan menurun secara signifikan atau bahkan tidak lagi menjadi *wait type* yang dominan. Hal ini menjadi bukti kuantitatif bahwa *bottleneck* telah berhasil dipindahkan dari subsistem I/O ke CPU, yang merupakan sebuah indikator dari kueri yang efisien dan berorientasi pada pemrosesan.

Seluruh temuan dari analisis ini dirangkum secara komprehensif dalam Tabel 4.6. Tabel ini menyajikan bukti kuantitatif yang holistik mengenai keberhasilan proses optimasi. Dapat diamati bahwa metrik Waktu Eksekusi dan Waktu CPU mengalami reduksi signifikan lebih dari 95%. Lebih lanjut, Logical Reads pada tabel Customers menurun secara drastis hingga 99,9%, yang mengonfirmasi bahwa operasi baca yang tidak efisien telah berhasil dihilangkan. Poin paling krusial terdapat pada baris terakhir, di mana Wait Type Dominan yang sebelumnya adalah PAGEIOLATCH_SH kini menjadi tidak signifikan. Ini adalah bukti final yang menghubungkan semua metrik lainnya dan membuktikan bahwa intervensi yang dilakukan telah berhasil mengatasi sumber *bottleneck* I/O.

Tabel 4. 6 Tabel Perbandingan Sebelum dan Sesudah Tuning dengan Wait Statistic

Metrik	**Sebelum Optimasi**	**Sesudah Optimasi**	**Perubahan (%)**

Waktu Eksekusi (elapsed time)	258.160 ms (3,28 menit)	12.490 ms (~12 detik)	**-95,2%**
Waktu CPU (CPU time)	240.250 ms	10.316 ms	**-95,7%**

Metrik	**Sebelum Optimasi**	**Sesudah Optimasi**	**Perubahan (%)**
Logical Reads (Customers)	4.138.336 halaman	205 halaman	**-99,9%**
Logical Reads (SalesHistory)	31.203 halaman	31.665 halaman	**+1,5%**
Wait Type Dominan	PAGEIOLATCH_SH (Tinggi)	PAGEIOLATCH_SH (Rendah/Tidak Signifikan)	**Bottleneck I/O Teratasi**

Rangkaian analisis ini menunjukkan sebuah alur yang logis. *Execution plan* mengidentifikasi operator yang tidak efisien. *Logical reads* mengukur tingkat inefisiensi operator tersebut. Dan *wait statistics*, sebagaimana diringkas dalam Tabel 4.6, mengonfirmasi bahwa inefisiensi I/O tersebut adalah penyebab utama waktu tunggu yang lama. Dengan demikian, dapat disimpulkan bahwa intervensi yang dilakukan tidak hanya meningkatkan kecepatan eksekusi kueri, tetapi juga telah berhasil mengatasi akar permasalahan performa secara fundamental.

Bab V
Penutup

5.1. Kesimpulan

Berdasarkan hasil Praktik Kerja Lapangan (PKL) yang telah dilaksanakan di PT Adicipta Inovasi Teknologi, dapat ditarik kesimpulan bahwa analisis dan implementasi *query performance tuning* merupakan komponen vital dalam pemeliharaan sistem basis data skala besar. Menjawab rumusan masalah pertama sekaligus mencapai tujuan awal praktik kerja lapangan ini, DBA telah menerapkan metodologi yang sistematis untuk mendiagnosis bottleneck performa kueri T-SQL, dimulai dari monitoring sesi aktif menggunakan prosedur sp_who2, pemeriksaan indeks dengan sys.dm_db_index_physical_stats, hingga evaluasi statistik melalui sys.dm_db_stats_properties. Studi kasus yang dilakukan menunjukkan bahwa identifikasi dan perbaikan anti-pattern seperti predikat non-SARGable dan *correlated subqueries* mampu menghasilkan peningkatan performa yang dramatis, dengan penurunan waktu eksekusi dan penggunaan sumber daya I/O lebih dari 95%.

Menjawab rumusan masalah kedua, berbagai teknik optimasi yang berpusat pada kueri (*query-centric*) telah diimplementasikan secara efektif. Strategi seperti refactoring sintaks agar SARGable, penggantian *correlated subquery* dengan *JOIN* eksplisit, pemilihan tepat antara Common Table Expressions (CTE) dan Temporary Tables, serta penanganan parameter sniffing menggunakan variabel lokal dan opsi OPTION (RECOMPILE), semuanya memberikan kontribusi nyata dalam menyelesaikan bottleneck. Temuan pada Tabel 4.4 memperlihatkan peningkatan signifikan dalam semua aspek performa. Pengalaman ini menegaskan bahwa performa optimal tidak hanya dicapai melalui desain indeks, tetapi juga sangat dipengaruhi oleh kualitas dan efisiensi penulisan kueri.

Terakhir, menjawab rumusan masalah ketiga sekaligus membuktikan keberhasilan dari solusi yang diterapkan, evaluasi kuantitatif terhadap metrik performa utama menunjukkan hasil yang sangat positif. Waktu eksekusi turun hingga lebih dari 95%, penggunaan CPU dan *logical reads* berkurang drastis, dan jenis *wait statistics* yang sebelumnya dominan seperti PAGEIOLATCH_SH menjadi tidak signifikan, sebagaimana tercantum dalam Tabel 4.6. Keberhasilan ini turut ditunjang oleh kolaborasi yang erat antara tim DBA dan tim pengembang melalui Jira dan Microsoft Teams, serta pemahaman mendalam terhadap konteks bisnis yang

diperoleh dari pelatihan internal. Secara keseluruhan, kegiatan PKL ini memberikan wawasan praktis bahwa performa basis data yang unggul merupakan hasil dari kombinasi antara pemeliharaan proaktif, penulisan kueri yang disiplin, serta pendekatan holistik yang mengintegrasikan aspek teknis dan fungsional sistem.

5.2. Saran

Berdasarkan pengalaman dan temuan selama pelaksanaan Praktik Kerja Lapangan, berikut adalah beberapa saran yang dapat dipertimbangkan untuk pengembangan di masa depan:

a. Bagi Perusahaan (PT Adicipta Inovasi Teknologi):

    1. Sangat disarankan untuk terus mengintegrasikan praktik *performance* tuning ke dalam siklus pengembangan perangkat lunak sejak tahap awal (*shift-left approach*). Mengadakan sesi *code review* yang secara spesifik berfokus pada efisiensi kueri dapat membantu mendeteksi potensi masalah sebelum dirilis ke produksi.

    2. Perusahaan dapat mempertimbangkan untuk mengembangkan atau mengadopsi *dashboard monitoring* performa basis data yang lebih terpusat. Dashboard ini dapat memvisualisasikan metrik-metrik kunci dari DMV secara historis, sehingga tren degradasi performa dapat diidentifikasi secara lebih proaktif oleh tim DBA maupun tim pengembang.

    3. Melanjutkan program pelatihan internal, baik yang berfokus pada aspek bisnis maupun teknis, terbukti sangat bermanfaat. Disarankan untuk membuat modul pelatihan khusus mengenai *best practices* penulisan T-SQL yang performan bagi para pengembang baru untuk menanamkan budaya sadar-performa sejak dini.

b. Bagi Universitas Ma Chung:

    1. Diharapkan pihak universitas dapat terus memperkaya kurikulum, khususnya pada mata kuliah sistem basis data, dengan menyertakan lebih banyak studi kasus praktis mengenai *performance tuning* dan

optimasi. Memberikan mahasiswa proyek nyata atau simulasi yang melibatkan analisis *Execution Plan* dan *query rewriting* akan sangat mempersiapkan mereka untuk tantangan di dunia industri.

2. Sebaiknya universitas memberikan pembekalan yang lebih spesifik mengenai penggunaan alat-alat kolaborasi profesional seperti Jira dan *platform* komunikasi tim sebelum mahasiswa memulai PKL. Pengenalan terhadap alur kerja standar industri ini akan membantu mahasiswa beradaptasi lebih cepat di lingkungan kerja.

c. Bagi Mahasiswa yang Akan Melaksanakan PKL:

1. Disarankan bagi mahasiswa untuk tidak hanya mengandalkan pengetahuan teoretis dari perkuliahan, tetapi juga secara proaktif mempelajari konsep-konsep praktis seperti membaca *Execution Plan* dan memahami anti-pattern umum dalam penulisan kueri.

2. Mahasiswa sebaiknya lebih aktif dalam berkomunikasi dan bertanya kepada pembimbing lapangan. Jangan ragu untuk meminta penjelasan mengenai konteks bisnis di balik tugas teknis yang diberikan, karena hal ini akan memperkaya pemahaman dan kualitas pekerjaan yang dihasilkan.

3. Mempersiapkan diri dengan mempelajari dasar-dasar alat manajemen proyek seperti Jira akan menjadi nilai tambah yang signifikan dan memudahkan proses adaptasi selama kegiatan PKL.

Daftar Pustaka

AdIns. (2024). *Dokumentasi internal PT Adicipta Inovasi Teknologi terkait sistem Aigoo dan Cherry*.

Atlassian. (2023). *Jira Software Documentation*. <https://www.atlassian.com/software/jira> [Diakses pada 23 Juli 2025]

Atma, Y. D., & Suhartati. (2017). *Optimasi Query Sederhana Guna Kecepatan Query Pada Database Server*. *Media Informatika*, 1(1), 6.

Chen, W. (2023). *Database Design and Implementation. Arkansas Tech University*. <https://orc.library.atu.edu/context/atu_oer/article/1002/viewcontent/OER_DatabaseDesignImplementation_WeiruChen.pdf> [Diakses pada 13 Juli 2025]

Choi, H., Lee, S., & Jeong, D. (2021). *Optimasi Query SQL Server dengan Teknik Indexing dan Performance*. *IEEE Access, 9*, 14564–14575. <https://doi.org/10.1109/access.2021.3052505> [Diakses pada 12 Juli 2025]

Firmansyah, E., Firdaus, H., & Samidi, S. (2025). *Optimasi Performa Query Subsidi Debitur dengan Index and Table Partition, Subquery and Indexing, dan Parallel Query Execution. Jurnal Pendidikan dan Teknologi Indonesia*, 5(6), 1769–1785. <https://doi.org/10.52436/1.jpti.824?> [Diakses pada 10 Juli 2025]

Fritchey, G. (2018). *SQL Server 2017 Query Performance Tuning: Troubleshoot and Optimize Query Performance*. Apress. <https://doi.org/10.1007/978-1-4842-3888-2> [Diakses pada 9 Juli 2025]

GeeksforGeeks. (2024). *SQL Server Management Studio (SSMS)*. <https://www.geeksforgeeks.org/sql-server/sql-server-management-studio-ssms/> [Diakses pada 9 Juli 2025]

Microsoft. (2023). *Microsoft Outlook Features and Overview*. <https://www.microsoft.com/en-us/microsoft-365/outlook/email-and-calendar-software> [Diakses pada 23 Juli 2025]

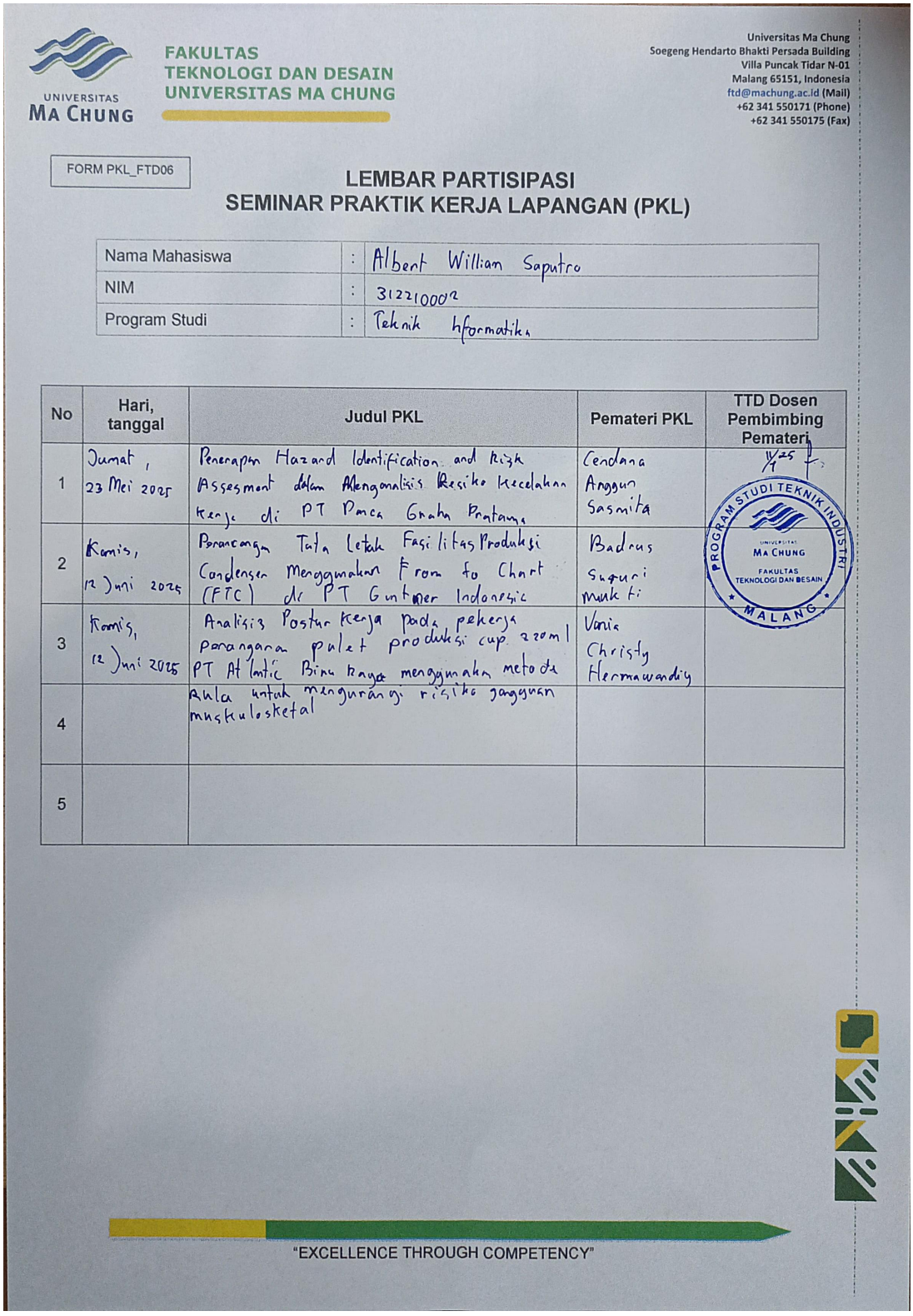
Microsoft. (2025). *SQL Server Management Studio (SSMS)*. <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms> [Diakses pada 23 Juli 2025]

Nevarez, B. (2022). *SQL server query tuning and optimization: Optimize Microsoft SQL Server 2022 queries and applications*. Packt.

Patil, S., Damare, P., Sonawane, J., & Maitre, N. (2015). *Study of Performance Tuning Techniques. Journal of Emerging Technologies and Innovative Research (JETIR)*, *2*(3), 499–502.

Lampiran

A. Lembar Partisipasi Seminar PKL



UNIVERSITAS MA CHUNG

FAKULTAS TEKNOLOGI DAN DESAIN UNIVERSITAS MA CHUNG

Universitas Ma Chung
Soegeng Hendarto Bhakti Persada Building
Villa Puncak Tidar N-01
Malang 65151, Indonesia
ftd@machung.ac.id (Mail)
+62 341 550171 (Phone)
+62 341 550175 (Fax)

FORM PKL_FTD06

LEMBAR PARTISIPASI
SEMINAR PRAKTIK KERJA LAPANGAN (PKL)

Nama Mahasiswa	:	Albert Willian Saputro
NIM	:	31221000?2
Program Studi	:	Teknik Informatika

No	Hari, tanggal	Judul PKL	Pemateri PKL	TTD Dosen Pembimbing Pemateri
1	Jumat, 23 Mei 2025	Penerapan Hazard Identification and Risk Assesment dalam Menganalisis Resiko Kecelakaan Kerja di PT Panca Graha Pratama	Candana Anggun Sasmita	
2	Kamis, 12 Juni 2025	Perancangan Tata Letak Fasilitas Produksi Condenser Menggunakan From to Chart (FTC) di PT Guntner Indonesia	Badaus Sugiri Mukti	
3	Kamis, 12 Juni 2025	Analisis Postur Kerja pada pekerja Perangakan pulet produksi cup 220ml PT Atlantic Bima Raya menggunakan metode	Vania Christy Hermawandiy	
4		Ahla untuk mengurangi risiko gangguan muskuloskeletal		
5				

PROGRAM STUDI TEKNIK INDUSTRI
UNIVERSITAS MA CHUNG
FAKULTAS TEKNOLOGI DAN BESAIN
MALANG

"EXCELLENCE THROUGH COMPETENCY"

B. Lembar Bimbingan PKL

UNIVERSITAS MA CHUNG
FAKULTAS TEKNOLOGI DAN DESAIN UNIVERSITAS MA CHUNG

Universitas Ma Chung
Soegeng Hendarto Bhakti Persada Building
Villa Puncak Tidar N-01
Malang 65151, Indonesia
ftd@machung.ac.id (Mail)
+62 341 550171 (Phone)
+62 341 550175 (Fax)

FORM PKL_FTD04

LEMBAR BIMBINGAN
PRAKTIK KERJA LAPANGAN (PKL)

Nama Mahasiswa	:	Albert William Saputra
NIM	:	312210002
Program Studi	:	Teknik Informatika
Judul Laporan PKL	:	ANALISIS DAN IMPLEMENTASI QUERY PERFORMANCE TUNING PADA MICROSOFT SQL SERVER MANAGEMENT STUDIO DI PT ADICIPTA INOVASI TEKNOLOGI (ADINS)

No	Hari, tanggal	Topik Bimbingan	TTD Dosen Pembimbing
1.	Rabu, 25-06-2025	Diskusi mengenai Metodologi Penelitian dan PKL	
2.	Selasa, 08-07-2025	Konsultasi mengenai Draf PKL	
3.	Senin, 21-07-2025	Konsultasi mengenai Draf PKL	
4.	Kamis, 24-07-2025	Konsultasi mengenai Draf PKL	
5.	Jumat, 25-07-2025	Konsultasi mengenai Draf PKL	

C. Lembar Jurnal Harian Pembimbing PKL

UNIVERSITAS MA CHUNG
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG

Universitas Ma Chung
Soegeng Hendarto Bhakti Persada Building
Villa Puncak Tidar N-01
Malang 65151, Indonesia
ftd@machung.ac.id (Mail)
+62 341 550171 (Phone)
+62 341 550175 (Fax)

FORM PKL_FTD05

JURNAL HARIAN PEMBIMBINGAN PRAKTIK KERJA LAPANGAN (PKL)

Nama Mahasiswa	:	Albert William Saputra
NIM	:	312210002
Program Studi	:	Teknik Informatika
Posisi PKL	:	Database Administrator (DBA)
Nama Instansi/Perusahaan	:	PT Adicipta Inovasi Tenologi (AdIns)
Alamat Instansi/Perusahaan	:	Jl. Lembah Dieng No.A1-7, Sumberjo, Kalisongo, Kec. Dau, Kabupaten Malang, Jawa Timur 65151
Nama Pembimbing Lapangan	:	Callista Angie

No	Hari, tanggal	Uraian Kegiatan PKL	TTD Dosen Pembimbing Lapangan
1.	Rabu, 26-02-2025	Training SQL Practice 1 & SQL Practice 2	
2.	Kamis, 27-02-2025	Review Training SQL Practice 1 & SQL Practice 2	
3.	Senin, 03- 03-2025	Training tuning index menggunakan execution plan	
4.	Kamis, 06-03-2025	Training Logshipping	
5.	Jumat, 07-03-2025	Check Logshipping Client & Training Logshipping Alert	
6.	Rabu, 12-03-2025	Training Mirroring	
7.	Rabu, 05-05-2025	Analisa Fragmentasi Index dan Membuat Rebuild Index	

Universitas Ma Chung
Soegeng Hendarto Bhakti Persada Building
Villa Puncak Tidar N-01
Malang 65151, Indonesia
ftd@machung.ac.id (Mail)
+62 341 550171 (Phone)
+62 341 550175 (Fax)

No	Hari, tanggal	Uraian Kegiatan PKL	TTD Dosen Pembimbing Lapangan
8.	Kamis, 16-05-2025	Tuning Kueri dengan Temp Table dan Membuat Index	
9.	Jumat, 17-05-2025	Housekeeping Table	
10.	Senin, 18-05-2025	Archiving Table	

D. Lembar Penilaian Eksternal PKL

UNIVERSITAS MA CHUNG

FAKULTAS
TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG

Universitas Ma Chung
Soegeng Hendarto Bhakti Persada Building
Villa Puncak Tidar N-01
Malang 65151, Indonesia
ftd@machung.ac.id (Mail)
+62 341 550171 (Phone)
+62 341 550175 (Fax)

FORM PKL_FTD03

LEMBAR PENILAIAN EKSTERNAL PRAKTIK KERJA LAPANGAN (PKL)

Nama Mahasiswa	:	Albert William Saputra
NIM	:	312210002
Program Studi	:	Teknik Informatika
Pembimbing Lapangan	:	Callista Angie
Tanggal Pelaksanaan	:	21 Februari 2025
Tempat PKL	:	PT. Adicipta Inovasi Teknologi (AdIns)
Topik PKL	:	Analisis dan implementasi *query performance tuning pada microsoft sql server management studio* di pt adicipta inovasi teknologi (adins)

Komponen Penilaian :

No	Komponen Penilaian	Nilai Angka (0-100)
1	Disiplin dalam kehadiran (presensi)	100
2	Kesungguhan dalam melakukan praktik kerja	100
3	Disiplin dalam pekerjaan	100
4	Kemampuan memecahkan masalah	97
5	Tanggung jawab	100
6	Kemauan untuk mengetahui hal-hal yang ada di tempat praktik kerja	100
7	Pengetahuan tentang ilmu yang dilaksanakan	95
8	Keterampilan	100
9	Kemampuan menyampaikan pendapat	100
10	Kemampuan dalam bekerja sama	100
	Rerata (Jumlah Nilai : 10 komponen)	99.2

Rentang Nilai:

MUTU	RENTANG	BOBOT NILAI
A	95,00 s/d 100	4,00
AB	90,00 s/d 94,99	3,70
BA	82,00 s/d 89,99	3,30
B	73,00 s/d 81,99	3,00
BC	65,00 s/d 72,99	2,70
CB	60,00 s/d 64,99	2,30
C	56,00 s/d 59,99	2,00
CD	50,00 s/d 55,99	1,50
D	40,00 s/d 49,99	1,00
E	00,00 s/d 39,99	0,00

Jakarta, 03 Juli 2025

Pembimbing Lapangan*

Callista Angie
Database Administrator
***dilengkapi dengan stempel perusahaan dan tanda tangan**

"EXCELLENCE THROUGH COMPETENCY"

E. Dokumentasi Selama Kegiatan PKL