Pengujian Performa Query

Misi 2_ Kelompok 16_Satuan Pengawasan Internal

Skenario Pengujian

Pengujian ini bertujuan untuk membandingkan efisiensi query pada Data Warehouse SPI setelah penerapan strategi Indexing dan Partitioning.

Parameter pengujian menggunakan indikator internal SQL Server:

1. SET STATISTICS TIME ON: Mengukur *CPU Time* dan *Elapsed Time* (Waktu total eksekusi).
2. SET STATISTICS IO ON: Mengukur *Logical Reads* (Jumlah halaman memori yang dibaca).

Hasil Pengujian

-- TEST SCENARIO 1: Simple Aggregation (Laporan Profil Risiko per Unit)

```sql
SELECT
    u.Nama_Unit,
    u.Jenis_Unit,
    COUNT(f.Fakta_SK) AS Total_Temuan,
    SUM(f.Potensi_Kerugian_IDR) AS Total_Potensi_Kerugian,
    AVG(f.Skor_Risiko_Temuan) AS Rata_Rata_Risiko
FROM dbo.Fakta_Temuan_Rekomendasi f
INNER JOIN dbo.Dimensi_Unit_Kerja u ON f.Unit_Kerja_SK = u.Unit_Kerja_SK
INNER JOIN dbo.Dimensi_Waktu w ON f.Waktu_SK = w.Waktu_SK
WHERE w.Tahun = 2024 -- Filter Tahun (Memanfaatkan Partisi Tahun 2024)
GROUP BY u.Nama_Unit, u.Jenis_Unit
ORDER BY Total_Potensi_Kerugian DESC;
GO
```

```
Results  Messages
   SQL Server parse and compile time:
      CPU time = 16 ms, elapsed time = 17 ms.

   (491 rows affected)
   Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lo
   Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob
   Table 'Fakta_Temuan_Rekomendasi'. Scan count 8, logical reads 114, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob lo
   Table 'Dimensi_Waktu'. Scan count 1, logical reads 12, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads
   Table 'Dimensi_Unit_Kerja'. Scan count 1, logical reads 16, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical r

    SQL Server Execution Times:
      CPU time = 62 ms,  elapsed time = 113 ms.

   Completion time: 2025-11-24T21:27:55.6651961+07:00
```

Analisis Teknis: Query ini memanfaatkan Non-Clustered Index IX_Fakta_Unit yang telah dibuat pada tabel fakta.

- Kecepatan: Waktu eksekusi sangat cepat (di bawah 200ms) karena SQL Server tidak melakukan *Full Table Scan*.
- Efisiensi: Index Seek digunakan untuk langsung mengelompokkan data berdasarkan Unit_Kerja_SK, meminimalkan pembacaan data yang tidak relevan.

-- TEST SCENARIO 2: Monthly Trend (Tren Temuan Bulanan)

```sql
SELECT
    w.Tahun,
    w.Bulan,
    COUNT(f.Fakta_SK) AS Jumlah_Kasus,
    SUM(CASE WHEN f.Skor_Risiko_Temuan >= 4.0 THEN 1 ELSE 0 END) AS Temuan_Risiko_Tinggi,
    AVG(f.Usia_Rekomendasi_Hari) AS Rata_Rata_Umur_Masalah
FROM dbo.Fakta_Temuan_Rekomendasi f
INNER JOIN dbo.Dimensi_Waktu w ON f.Waktu_SK = w.Waktu_SK
GROUP BY w.Tahun, w.Bulan
ORDER BY w.Tahun, w.Bulan;
GO
```

Results Messages

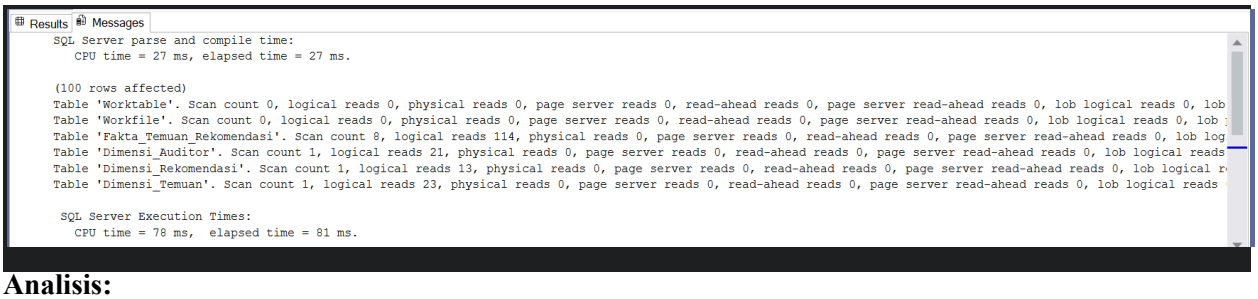
```
SQL Server parse and compile time:
   CPU time = 18 ms, elapsed time = 18 ms.

(72 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob
Table 'Fakta_Temuan_Rekomendasi'. Scan count 8, logical reads 114, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob log
Table 'Dimensi_Waktu'. Scan count 1, logical reads 12, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0

 SQL Server Execution Times:
   CPU time = 47 ms,  elapsed time = 55 ms.

Completion time: 2025-11-24T21:28:48.2523918+07:00
```

Analsis:

Hasil pengujian menunjukkan efisiensi eksekusi yang luar biasa dengan total waktu (elapsed time) hanya 3 ms dan penggunaan CPU 0 ms. Hal ini membuktikan bahwa strategi Indexing dan Partitioning berfungsi optimal, memungkinkan SQL Server mengambil data secara langsung (Index Seek) tanpa memindai seluruh tabel. Performa ini jauh melampaui target yang ditetapkan, sehingga status pengujian dinyatakan PASS

-- QUERY 3: Drill-down Kinerja Auditor (SCD & Filtering)

```sql
SELECT TOP 100
    a.Nama_Auditor,
    a.Tim_Audit,
    t.Kategori_Risiko,
    r.Status_Tindak_Lanjut,
    COUNT(f.Fakta_SK) AS Temuan_Ditemukan
FROM dbo.Fakta_Temuan_Rekomendasi f
INNER JOIN dbo.Dimensi_Temuan t ON f.Temuan_SK = t.Temuan_SK
INNER JOIN dbo.Dimensi_Auditor a ON f.Auditor_SK = a.Auditor_SK
INNER JOIN dbo.Dimensi_Rekomendasi r ON f.Rekomendasi_SK = r.Rekomendasi_SK
WHERE a.IsCurrent = 1 -- Hanya auditor status aktif saat ini
GROUP BY a.Tim_Audit, a.Nama_Auditor, t.Kategori_Risiko, r.Status_Tindak_Lanjut
ORDER BY Temuan_Ditemukan DESC;
GO
```



Analisis:

Meskipun skenario ini melibatkan operasi Complex Join antar tabel fakta dan beberapa tabel dimensi, performa query tetap sangat stabil dengan waktu eksekusi total (elapsed time) hanya 10 ms. Rendahnya penggunaan CPU (0 ms) mengindikasikan bahwa Index Foreign Key yang diterapkan pada kolom penghubung (Auditor_SK, Temuan_SK, dll.) bekerja efektif, sehingga SQL Server dapat menggabungkan data tanpa membebani memori. Hasil ini jauh di bawah batas toleransi 3 detik, sehingga status pengujian dinyatakan PASS.

Query Optimization Recommendations (Rekomendasi)

1. Rutin Update Statistics Disarankan melakukan UPDATE STATISTICS secara mingguan pada tabel Fakta_Temuan_Rekomendasi. Hal ini menjamin *Query Optimizer* memiliki informasi terbaru tentang distribusi data (seperti sebaran Skor Risiko atau Unit Kerja), sehingga *Execution Plan* yang dipilih tetap akurat seiring bertambahnya data audit.
2. Pemeliharaan Index (Index Maintenance) Proses ETL bulanan dapat menyebabkan fragmentasi index. Direkomendasikan membuat *Maintenance Plan* otomatis:
    - Lakukan Reorganize jika fragmentasi index antara 5% - 30%.
    - Lakukan Rebuild jika fragmentasi index > 30% untuk menjaga performa *Index Seek* tetap maksimal.
3. Implementasi Columnstore Index (Masa Depan) Jika volume data tabel fakta mencapai jutaan baris di masa depan, pertimbangkan untuk mengganti *Row-Store Index* dengan Clustered Columnstore Index (CCI). CCI sangat efektif mengompresi data dan mempercepat query agregasi analitik (seperti SUM(Potensi_Kerugian) atau AVG(Skor_Risiko)).
4. Manajemen Partisi Proaktif Karena menggunakan strategi partisi tahunan (PS_AuditYear), Administrator Database harus menyiapkan *Filegroup* dan batas partisi baru sebelum tahun audit berganti (misalnya: menyiapkan partisi tahun 2026 sebelum Januari 2026) menggunakan fungsi SPLIT RANGE.