

Misi 2 kelompok 6. LPMPP

Pengujian dilakukan untuk membandingkan efisiensi query pada Data Warehouse setelah penerapan strategi Indexing dan Partitioning. Pengujian menggunakan parameter SET STATISTICS IO dan SET STATISTICS TIME.

```
-- 1. Aktifkan Mode Pencatatan Waktu & I/O
-- SET STATISTICS TIME ON; -- Buat liat durasi (ms)
SET STATISTICS IO ON; -- Buat liat seberapa berat dia baca disk
GO

PRINT '*** TEST 1: QUERY AGREGASI JOIN (Menguji Index) ***';
-- Query ini memaksa database pakai Index di Foreign Key (StatusKey & JenisIKKey)
-- Harusnya cepat banget karena udah di-index di Step 4.

SELECT
    j.Nama_JenisKI,
    s>Nama_Status,
    SUM(f.JumlahPengkajuan) AS Total_Pengkajuan,
    SUM(f.BiayaPendaftaran) AS Total_Biaya
FROM dbo.Fact_Pengkajuan f
JOIN dbo.Dim_JenisKI j ON f.JenisIKKey = j.JenisIKKey
JOIN dbo.Dim_Status s ON f.StatusKey = s.StatusKey
GROUP BY j.Nama_JenisKI, s>Nama_Status
ORDER BY Total_Pengkajuan DESC;
GO
```

91 %

Results Messages

```
RQL Server Execution Time:
CPU time = 0 ms. elapsed time = 0 ms.

RQL Server parse and compile time:
CPU time = 0 ms. elapsed time = 20 ms.

*** TEST 1: QUERY AGREGASI JOIN (Menguji Index) ***

RQL Server Execution Time:
CPU time = 0 ms. elapsed time = 0 ms.
```

1st row affected:

Table 'Merkatilis'	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 physical reads 0,	lob page server reads 0,	lob read-ahead reads 0,
Table 'Fact_Pengangkutan'	Scan count 7, <th>logical reads 51,</th> <th>physical reads 67,</th> <th>page server reads 67,</th> <th>read-ahead reads 67,</th> <th>page server read-ahead reads 67,</th> <th>lob logical reads 0,</th> <th>lob physical reads 0,</th> <th>lob page server reads 0,</th> <th>lob read-ahead reads 0,</th>	logical reads 51,	physical reads 67,	page server reads 67,	read-ahead reads 67,	page server read-ahead reads 67,	lob logical reads 0,	lob physical reads 0,	lob page server reads 0,	lob read-ahead reads 0,
Table 'Dim_JenisKendaraan'	Scan count 1, <th>logical reads 3,</th> <th>physical reads 1,</th> <th>page server reads 3,</th> <th>read-ahead reads 1,</th> <th>page server read-ahead reads 1,</th> <th>lob logical reads 0,</th> <th>lob physical reads 0,</th> <th>lob page server reads 0,</th> <th>lob read-ahead reads 0,</th>	logical reads 3,	physical reads 1,	page server reads 3,	read-ahead reads 1,	page server read-ahead reads 1,	lob logical reads 0,	lob physical reads 0,	lob page server reads 0,	lob read-ahead reads 0,
Table 'Dim_Status'	Scan count 1, <th>logical reads 3,</th> <th>physical reads 1,</th> <th>page server reads 3,</th> <th>read-ahead reads 1,</th> <th>page server read-ahead reads 1,</th> <th>lob logical reads 0,</th> <th>lob physical reads 0,</th> <th>lob page server reads 0,</th> <th>lob read-ahead reads 0,</th>	logical reads 3,	physical reads 1,	page server reads 3,	read-ahead reads 1,	page server read-ahead reads 1,	lob logical reads 0,	lob physical reads 0,	lob page server reads 0,	lob read-ahead reads 0,

RQL Server Execution Time:
CPU time = 18 ms. elapsed time = 24 ms.

Completion time: 2020-11-20T20:16:03.957Z+0400+07:00

Analisis:

C. Hasil Pengujian 2: Query Filter Partisi (Partition Pruning)

```
SQLQuery1.sql - C:\CAELAN\kn409 (57) - 4 X
SET STATISTICS IO ON; -- Buat lihat seberapa berat dia baca disk
GO

PRINT '=== TEST 2: QUERY FILTER PARTISI (Menguji Partition Pruning) ===';

SELECT
    DateKey_Daftar,
    COUNT(PengajuanKey) AS Jumlah_Harian
FROM dbo.Fact_PengajuanKl
WHERE DateKey_Daftar BETWEEN 20240101 AND 20241231 -- Filter 1 Tahun
GROUP BY DateKey_Daftar
ORDER BY DateKey_Daftar;
GO

-- Matikan Mode Statistik
SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
GO
```

91%

Results Messages

```

SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 10 ms.
=== TEST 2: QUERY FILTER PARTISI (Menguji Partition Pruning) ===

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

(60 rows affected)
Table 'Fact_PengajuanKl'. Scan count 1, logical reads 9, physical reads 1, page reads 0, read-ahead reads 0, page reads read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page reads 0, lob reads 0.

```

Analysis:

Query melakukan filter data khusus tahun 2024. Strategi partisi terbukti berhasil melakukan *Partition Pruning*, di mana SQL Server hanya membaca 1 partisi (Scan count = 1) dan mengabaikan partisi tahun lainnya. Hal ini menurunkan *Logical Reads* secara signifikan menjadi 9 page reads dengan waktu eksekusi 0 ms (instant).

D. Query Optimization Recommendations (Rekomendasi)

Rekomendasi Optimasi Lanjutan:

1. Maintenance Statistics Rutin:

Disarankan untuk menjadwalkan *Update Statistics* secara berkala (mingguan) agar Query Optimizer selalu memiliki metadata terbaru tentang distribusi data, memastikan rencana eksekusi (Execution Plan) tetap optimal seiring bertambahnya data.

2. Index Maintenance (Rebuild/Reorganize):

Seiring berjalannya waktu, proses ETL (Insert/Delete) dapat menyebabkan *fragmentasi index*. Disarankan membuat *Maintenance Plan* untuk melakukan *Rebuild Index* jika fragmentasi > 30% atau *Reorganize* jika fragmentasi antara 5-30%.

3. Penerapan Columnstore Index (Opsional):

Jika volume data tabel Fakta (Fact_PengajuanKI) mencapai jutaan baris di masa depan, disarankan untuk mengganti *Row-Store Index* dengan Clustered Columnstore Index (CCI). CCI dapat mengompresi data hingga 10x dan mempercepat query agregasi analitik secara signifikan.

4. Manajemen Partisi Tahunan:

Karena tabel menggunakan strategi partisi tahunan (PF_TahunPengajuan), administrator database harus menyiapkan *Filegroup* dan *Scheme* baru sebelum tahun berganti (contoh: menyiapkan partisi 2026 sebelum Januari 2026) menggunakan fungsi SPLIT RANGE.