OPTIMASI SQL

Imam kuswardayan, S.Kom, M.T

Alasan Optimasi

- Waktu adalah uang dan pengguna tak suka menunggu sehingga program harus berjalan cepat.
- Pada online atau aplikasi client – server internet, semakin banyak user yang sedang online laluintas data sangat tinggi.
- Sekalipun menggunakan server yang lebih cepat, terbukti hal ini hanya faktor kecil dibandingkan dengan pemilihan algoritma yang digunakan.

Skema Database

- Desain buruk dan tidak ternormalisasi memberikan efek besar dalam kecepatan.
- Skema database sebelum normalisasi 3NF menyebabkan pembacaan data yang lebih besar.
- 2NF dan 3NF meminimalkan redundansi data dengan ketergantungan fungsional (Functional Dependency)

Query cukup hanya yang diperlukan saja

- Gunakan filter sebanyak mungkin sebelum proses
- Klausa WHERE adalah bagian penting untuk optimasi
- Pilih hanya kolom-kolom yang diperlukan
  - Jangan pernah gunakan SELECT *, pilih kolom yang dipakai saja. Hal ini akan mempercepat dan mengurangi bandwith

Hati-hati dalam menggabungkan tabel

- Penggabungan tabel membutuhkan waktu lama, gunakan seperlunya, jangan gabungkan tabel-tabel yang tidak diperlukan
- Gunakan primary key jika join dengan tabel, serta join terhadap fields yang telah diindex.
- Gunakan klausa JOIN daripada WHERE untuk penggabungan

Klausa NOT IN

- Ubah klausa NOT IN menjadi NOT EXISTS

```
SELECT * FROM tabel1 a
WHERE a.kolom1 NOT IN
( SELECT b.kolom1 FROM tabel2 b)
```

```
SELECT * FROM tabel1 a
WHERE a.kolom1 NOT EXISTS
( SELECT b.kolom1 FROM tabel2 b)
```

Apa beda dua *statement* SQL ini? Klausa IN dan NOT IN memeriksa apakah sebuah nilai terdapat dalam list (bisa *array*, bisa *correlated subquery* seperti contoh di atas)

sedangkan klausa EXISTS dan NOT EXISTS hanya memeriksa keberadaan ada atau tidaknya row pada suatu list. Secara *performance*, tentu jauh lebih cepat EXISTS daripada IN dalam hal *correlated subquery*.

Optimasi Query dan Stored Procedure

- Pandangan umum, jika query berjalan lebih dari 1 detik, nampaknya perlu dilakukan optimasi.
- Mulailah pada query yang paling sering digunakan dan membutuhkan paling banyak waktu eksekusi.

Manajemen Index

- Setiap primary key memerlukan index karena menyebabkan penggabungan jadi lebih cepat.
- Setiap tabel memerlukan primary key agar lebih cepat.
- Gunakan indeks pada fields yang sering digunakan untuk filter pada klausa WHERE
- Hati-hati dalam menambahkan index pada field yang sering diupdate karena DBMS perlu melakukan reindex setelah update.

Stored Procedure

- Pindahkan Query anda ke Stored Procedure (SP). SP terkompilasi, sehingga lebih cepat dieksekusi daripada SQL code.
- SP tidak memerlukan banyak bandwith sebab anda bisa melakukan banyak query di dalam sebuah SP dan tetap di server sampai hasil akhir query dikembalikan.

Stored Procedure

- SP dijalankan diserver sehingga lebih cepat
- Kalkulasi dalam bentuk code (VB, Java, C++, ...) tidak secepat SP pada sebagian besar kasus.
- Dengan SP, code untuk akses DB terpisah dengan *presentation layer* (menggunakan 3-tiers model), sehingga mudah untuk maintain.

Buang View yang tidak dipake

- Views adalah bentuk khusus query. View tersimpan secara logical. Sehingga setiap kali menjalankan **SELECT * FROM myView**, anda menjalankan 2 query:
  - query yang dipakai untuk menampilkan myView
  - dan query isi **myView** nya.
- Jika anda selalu membutuhkan info yang sama, itu baik
- Jika anda perlu filter dalam view, maka lebih lambat karena menjalankan query pada query.

Tune DB Setting

- Anda dapat menggunakan DB Tunning dalam banyak cara: update statistik yang digunakan oleh optimizer, menjalankan opsi optimization, membuat DB read only dsb.

Gunakan Query Analyzer

- Pada banyak DBMS, ada tool untuk menjalankan dan mengoptimalkan query.
- Misal: SQL Server memiliki tool Query Analyzer:
  - Tulis query dan eksekusi
  - Melihat rencana eksekusi (execution plan)

PRAKTEK dan LATIHAN

Imam kuswardayan, S.Kom, M.T

Kasus 1

- KASUS:
  - Tampilkan Nama dan skor pemain yang berjenis kelamin laki-laki
- **JAWAB:**
  - **Query:** SELECT * FROM Pemain
  - **App User Control:** menambahkan property Filter: jeniskelamin = ‘L’

Optimal

JAWAB BENAR:

SELECT Nama, Skor FROM Pemain WHERE jeniskelamin=‘L’

data yang melalui jaringan lebih sedikit dengan memilih kolom seperlunya dan filter

Kasus 2

- **ORIGINAL:**
  - **Query:**

SELECT Nama, Skor FROM Pemain WHERE jeniskelamin='L'

Optimal

Anda membutuhkan klausa ORDE BY ? Sering. Jika tidak, buang.

Jika iya, jadikan dalam query semuanya

  - **App User Control:**

Sort: Skor

KASUS 3

- **ORIGINAL:**
  - For (i=0;i<2000;i++)
    - **CALL query:**
    - SELECT Skor FROM Pemain WHERE UID = parameter(i)

Optimal

Query asli tersebut membutuhkan banyak bandwith jaringan dan menyebabkan seluruh sistem melambat

SELECT Skor FROM Pemain WHERE UID>0 AND UID<2000

KASUS 4

- **ASK:** nama dan lokasi pemain saat ini

- **ORIGINAL:**

```
SELECT Nama, NamaLokasi
FROM Pemain, Lokasi
WHERE IDLokasiSaatIni=IDLokasi
```


```
SELECT Nama, NamaLokasi
FROM Pemain
INNER JOIN Lokasi ON IDLokasiSaatIni = IDLokasi
```

KASUS 5

- **ASK:** 1000 kota dan deskripsi berbahasa Indonesia
- **ORIGINAL:**
  - SELECT NamaLokasi, Deskripsi
    FROM lokasi JOIN Lokasi_deskripsi
    ON lokasi.idLokasi = Lokasi_deskripsi.idLokasi
    WHERE idBahasa=“id”

Filter dulu baru join
SELECT NamaLokasi, Deskripsi
FROM lokasi JOIN (SELECT deskripsi FROM Lokasi_deskripsi
WHERE idBahasa=“id”) LD ON lokasi.idLokasi = ld.idLokasi LIMIT 1000