05

Optimasi Query

by: Ahmad Syauqi Ahsan

Optimasi Query

- Misalkan anda diberi kesempatan untuk mengunjungi 15 kota yang berbeda di Eropa. Satu-satunya batasan yang ada adalah "Waktu".
- Apakah anda mempunya perencanaan (plan) untuk mengunjungi kota-kota tersebut dengan urutan tertentu ?

Optimasi Query (2)

- Perencanaan:
  - Letakkan 15 kota tersebut kedalam beberapa grup berdasarkan kedekatan antara satu dengan yang lain.
  - Mulai kunjungi kota-kota dalam satu grup, baru kemudian pindah ke grup berikutnya
- Hal penting yang dapat diambil:

  Anda dapat mengunjungi semua kota secara lebih terorganisir, sehingga penggunaan waktu akan lebih efisien

Optimasi Query (3)

- Optimasi Query bekerja mirip seperti kasus diatas.
- Ada banyak cara untuk menghasilkan jawaban yang sama dari suatu Query.
- DBMS berusaha keras untuk memproses query dengan cara yang paling efisien (dalam hal penggunaan waktu) untuk menghasilkan jawaban.
- Biaya (Cost) = Waktu yang dibutuhkan untuk mendapatkan semua jawaban
- Hampir semua DBMS menggunakan **cost-based optimizer** untuk mengoptimalkan query.

Cost-Based Query Optimization

- Tahapan-tahapan pada Cost-Based Query Optimization:
  1. Parsing
  2. Transformasi (Transformation)
  3. Implementasi (Implementation)
  4. Perencanan pemilihan skenario query berdasarkan estimasi biaya (Plan selection based on cost estimates)

Diagram Alir Query

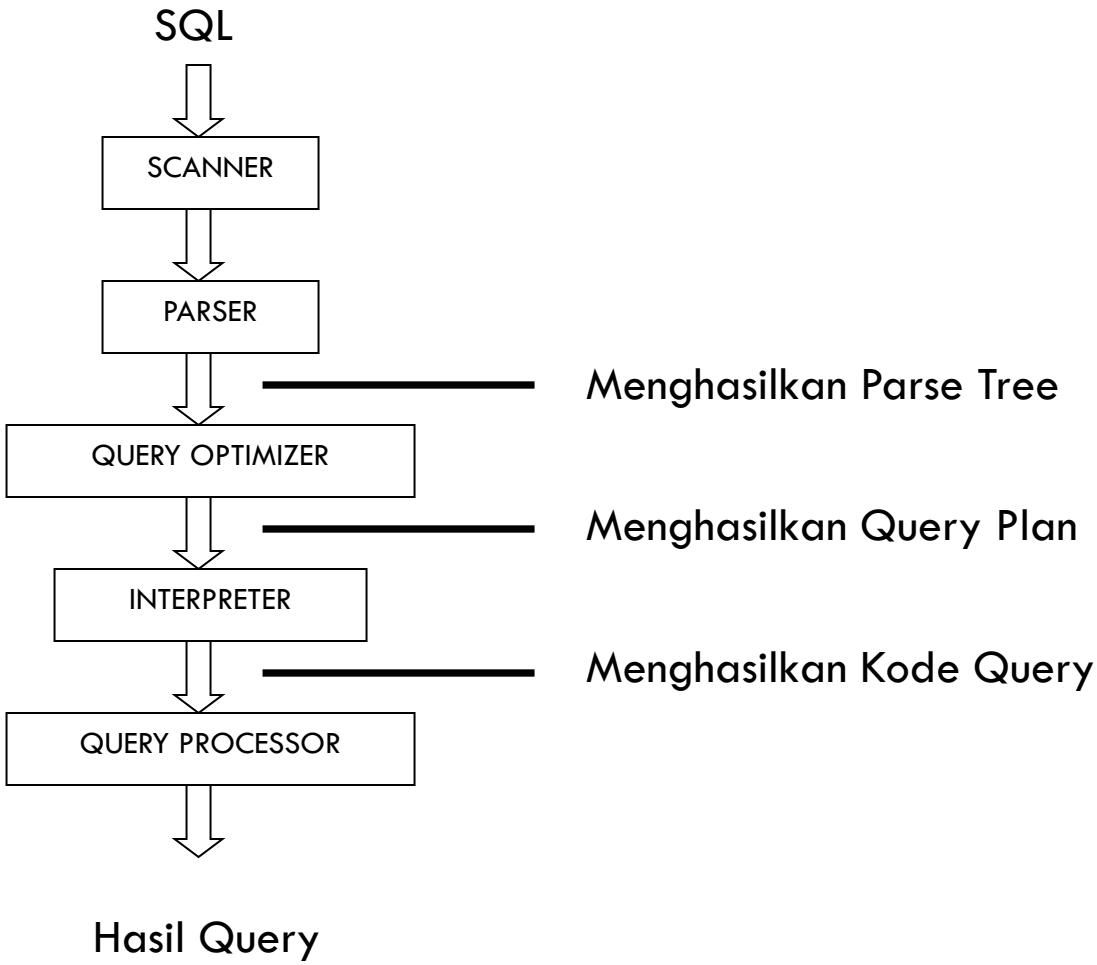


Diagram Alir Query (2)

- **Query Parser** – Melakukan verifikasi terhadap validitas dari statemen SQL, kemudian menterjemahkan Query kedalam struktur internal menggunakan kalkulus relasional.
- **Query Optimizer** – Mencari ekspresi terbaik dari beberapa ekspresi aljabar yang berbeda. Kriteria yang digunakan adalah "Murahnya Biaya"
- **Code Generator/Interpreter** – Menterjemahkan Query Plan yang dihasilkan oleh Query Optimizer menjadi Query Code untuk kemudian dikirimkan ke Query Processor
- **Query Processor** – Mengeksekusi Query Code yang didapat dari Code Generator.

Physical Plan

- Yang termasuk biaya dari Physical Plan adalah waktu penggunaan prosesor serta waktu komunikasi.
- Faktor yang paling penting untuk dipertimbangkan adalah proses baca-tulis (I/O) pada storage penyimpanan (harddisk), karena hal ini adalah aksi yang paling banyak mengkonsumsi waktu.
- Beberapa biaya yang lain yang masih berhubungan:
  - Operasi-operasi (join, union, dan intersection)
  - Urutan operasi
- Penggunaan join, union, atau intersection harus dibatasi dan diminimalkan untuk menghasilkan Physical Plan terbaik.

Projection

- Projection menghasillkan tuple untuk setiap argument tuple.
- Perubahan yang terjadi pada ukuran output adalah panjang dari tuple-tuple yang ada.

- Misalkan, untuk relasi/tabel 'R'
  - Relasi (20.000 tuple): R(a, b, c)
  - Setiap Tuple (190 bytes): header = 24 bytes, a = 8 bytes, b = 8 bytes, c = 150 bytes
  - Setiap Block (1024): header = 24 bytes

Projection (2)

- Kita dapat memasukkan 5 Tuple kedalam 1 Block.
  - 5 Tuples * 190 bytes = 950 bytes → dapat menggunakan 1 Block
  - Untuk 20.000 Tuple, kita membutuhkan 4.000 Block (20.000 / 5 = 4000)
- Menggunakan Projection menghasilkan eliminasi dari kolom **c** (150 bytes), kita dapat mengestimasi bahwa setiap Tuple akan berkurang menjadi 40 bytes (190 – 150 = 40)

Projection (3)

- Sekarang, estimasinya adalah 25 Tuple dalam 1 Block
- 25 Tuple * 40 bytes/tuple = 1000 bytes → dapat dimasukkan kedalam 1 blok
- Dengan 20.000 Tuple, estimasi yang baru adalah 800 Block (20.000 Tuple / 25 Tuple per Block = 800 Block)

Interaksi Query dalam DBMS

- Bagaimana Query berinteraksi dengan DBMS ?
  - Interactive users
  - Query yang tertanam (embedded query) didalam program yang ditulis dalam C, C++, Java, dan lain-lain
- Apa perbedaan diantara keduanya ?

Interactive Users

- Interactive Users adalah user/pengguna yang mengakses/meng-query database dengan mengirimkan perintah SQL langsung ke DBMS.
  (Untuk Oracle biasanya menggunakan SQL Plus)
- Ketika ada query dari interactive users, query tersebut akan langsung melewati Query Parser, Query Optimizer, Code Generator, dan Query Processor setiap kali dieksekusi.

Embedded Query

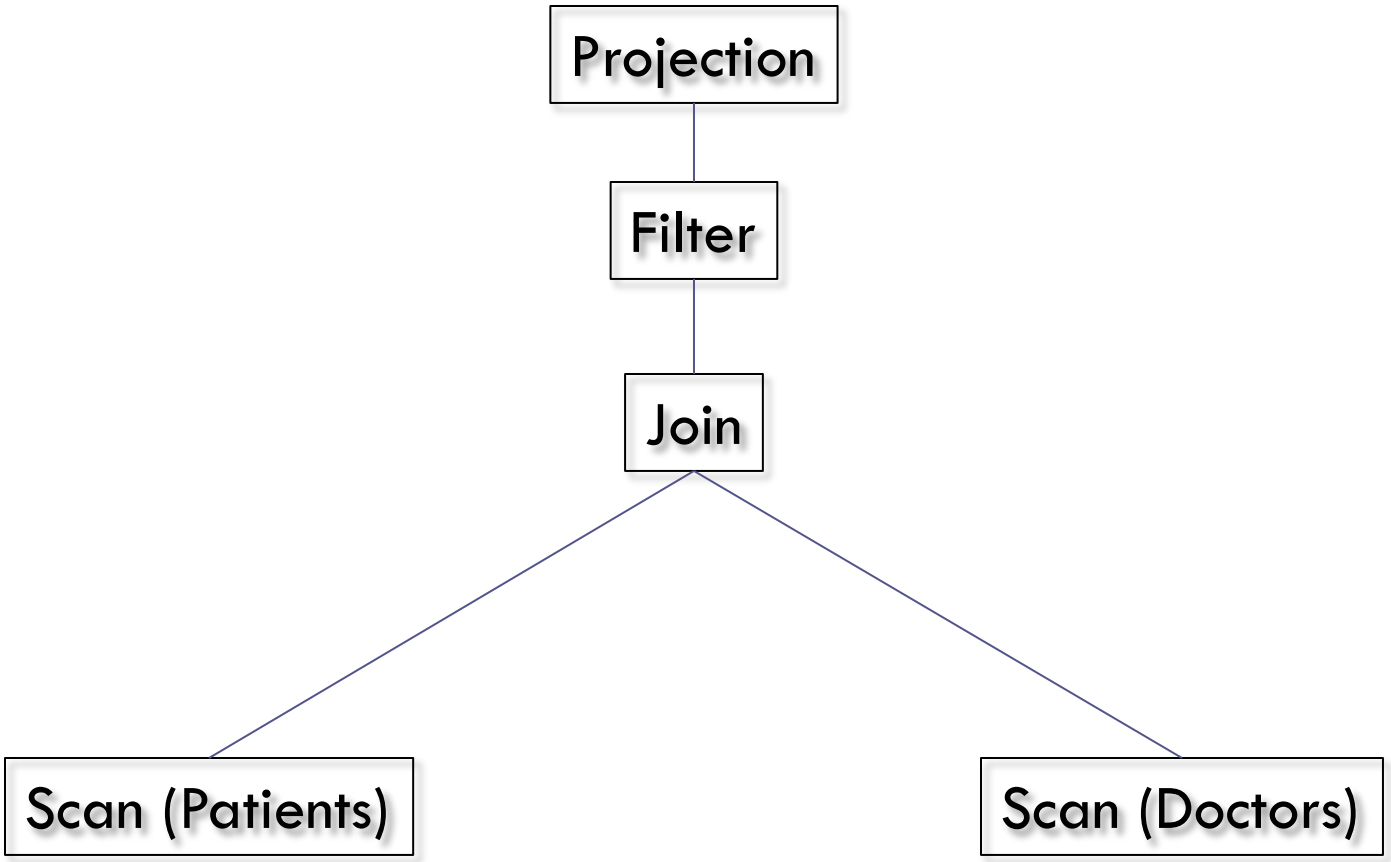
- Ketika ada embedded query, query tersebut tidak harus melewati Query Parser, Query Optimizer, Code Generator, dan Query Processor setiap kali dieksekusi.
- Pada Embedded Query, permintaan-permintaan (**calls**) yang dibangkitkan oleh Code Generator disimpan didalam basis data. Setiap kali query yang berada didalam program dijalankan saat run-time, Query Processor memanggil **calls** yang telah tersimpan didalam database
- Proses optimasi tidak memiliki ketergantungan dalam embedded query.

Cost-based query Optimization: Algebraic Expressions

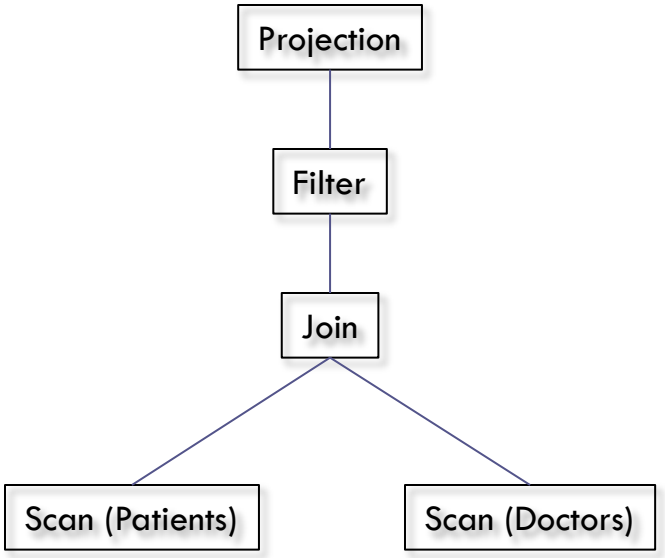
- Jika kita mempunyai query seperti berikut:

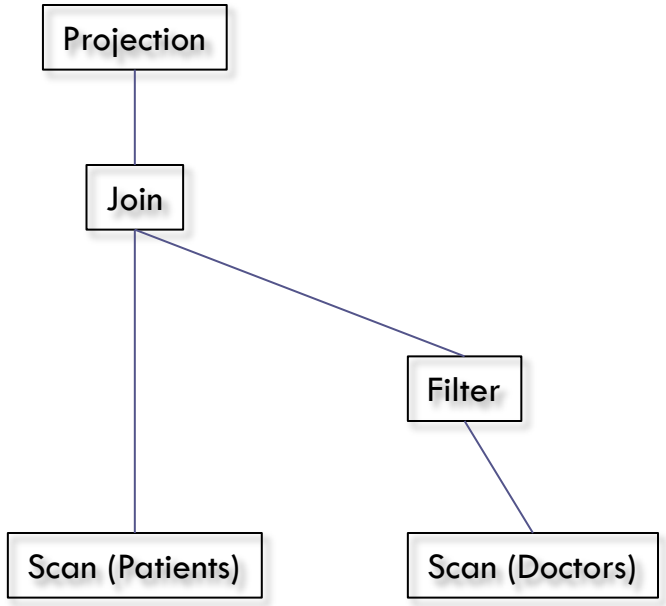
```
SELECT   p.pname, d.dname
FROM     Patients p, Doctors d
WHERE    p.doctor = d.dname
         AND d.gender = 'M'
```

Algebraic Expression

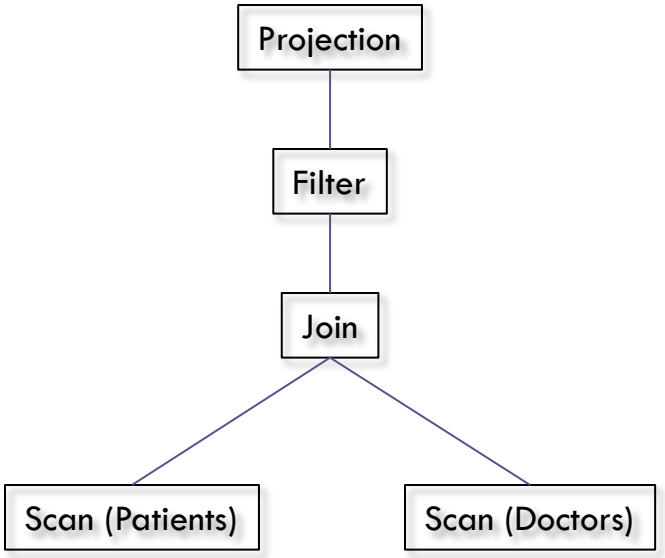


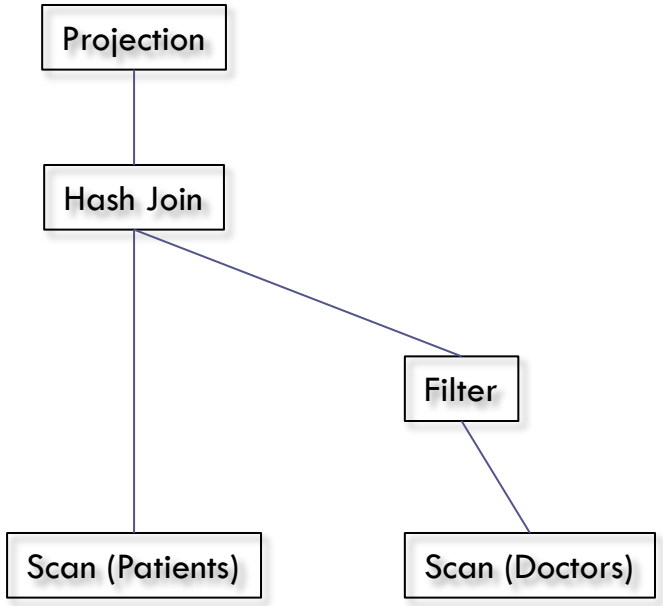
Cost-based query Optimization : Transformation



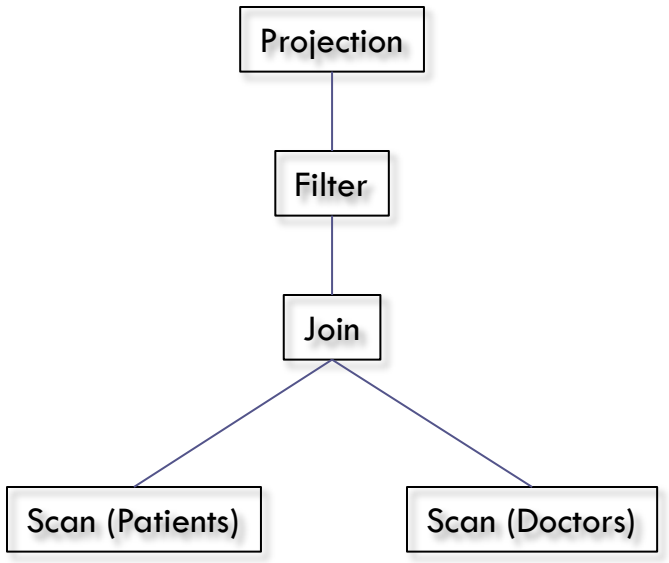


Cost-based query Optimization : Implementation

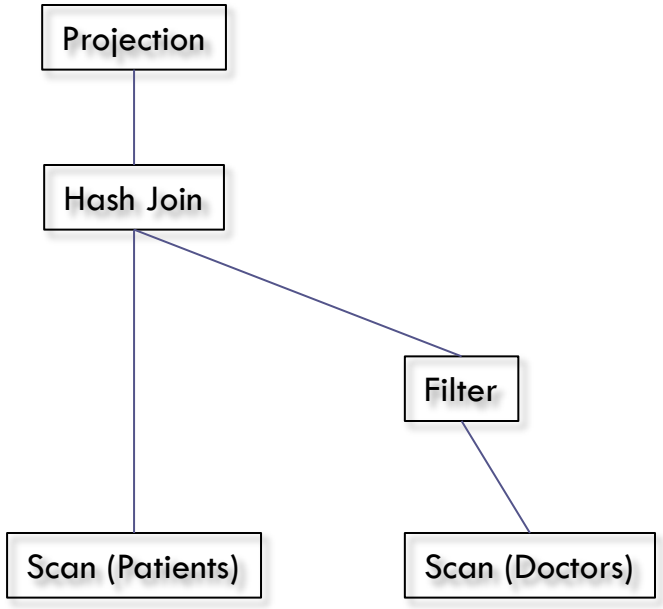




Cost-based query Optimization: Plan selection based on costs



Estimated Cost = 100ms



Estimated Cost = 50ms

Tanya Jawab

Terima Kasih