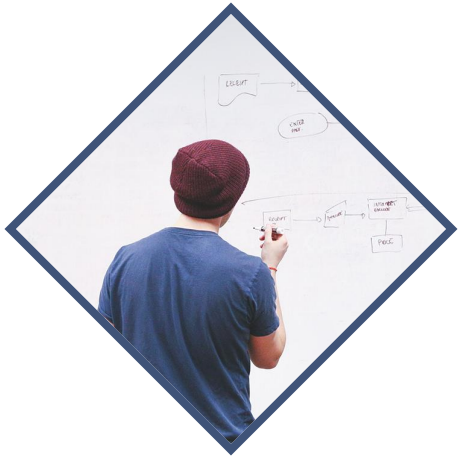
STMIK
Amik Riau

MATAKULIAH
BASIS DATA TERDISTRIBUSI

Rahmaddeni, M.Kom
NIDN : 1007128301

Pertemuan 4 – Optimasi Query Terdistribusi,



Visi Program Studi

Menjadi Program Studi Teknik Informatika yang unggul dalam bidang mobile computing di Sumatera tahun 2030

STMIK Amik Riau

#01

PENGENALAN OPTIMASI QUERY (Query Optimizer)

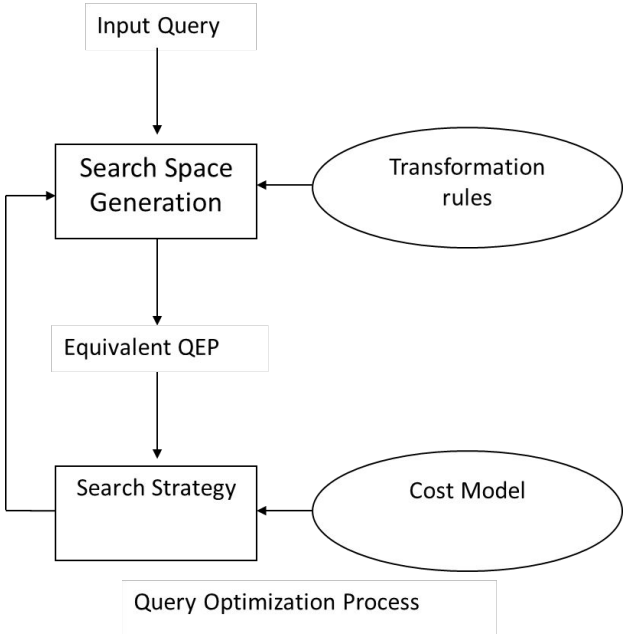
OPTIMASI QUERY

Optimasi Query adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi tersebut menjadi lebih efektif.

Dengan kata lain, Optimasi Query mencari urutan optimal dari operasi query.
Strategi yang terpilih diharapkan dapat meminimalisasi biaya yang harus dikeluarkan dalam pengeksekusian query.

OPTIMASI QUERY

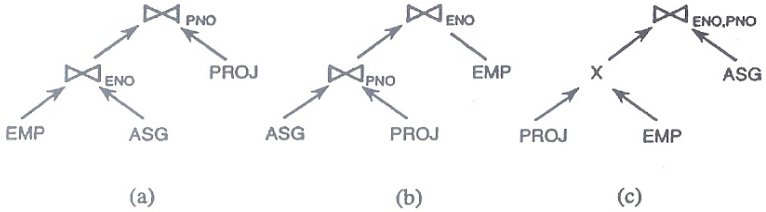
Query Optimizer, sebuah modul software yang mendukung optimasi query terdiri dari tiga komponen, yaitu :



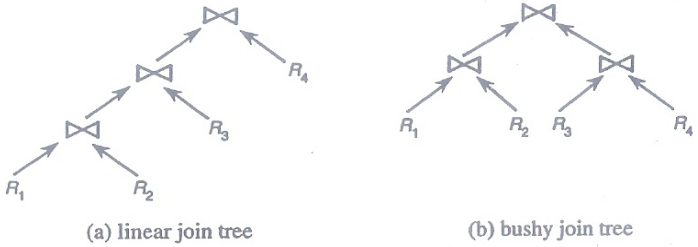
Query Optimization Process

SEARCH SPACE

- Merupakan sekumpulan alternatif perencanaan eksekusi yang merepresentasikan query.
- Seluruh alternatif adalah ekuivalen, yaitu memberikan hasil yang sama dengan urutan pengoperasian, pengimplementasian, dan performance yang berbeda.
- Representasi dari query tersebut dibentuk dalam suatu *operator trees* yang dapat dibuat dengan aturan transformasi.



Equivalent Join trees



Bentuk Umum Join trees

COST MODEL

Cost Model memprediksi biaya yang harus dikeluarkan dari suatu query yang akan dieksekusi.

Terdiri dari :

- Cost function
- Statistic database
- Formula.

SEARCH STRATEGY

Search strategy mengeksplorasi search space dan memilih strategi terbaik dari alternatif yang ada menggunakan cost model.

Search Strategy yang dapat digunakan adalah **Dynamic Programming** yang bersifat determenistik, di mana menghasilkan solusi terbaik, dan **Randomized Strategies** (seperti Iterative Improvement dan Simulated Annealing) yang berkonsentrasi pada pencarian solusi optimal dengan menghindari biaya tinggi dalam optimasi.

STMIK
Amik Riau

#02

OPTIMASI QUERY TERPUSAT

DEFENISI

Agar operasi *query* tidak lambat diperlukan metode yang dapat mengoptimalkan operasi *join query* tersebut. Suatu metode optimasi *query* mencoba untuk menentukan cara yang paling optimal untuk mengeksekusi *query* yang diberikan dengan mempertimbangkan rencana *query* yang mungkin.

Terdapat beberapa metode optimasi *query* untuk basis data terdistribusi diantaranya *Dynamic Approach*, *Static Approach*, *Semi Join Based Approach* dan *Hybrid Approach*.

Optimasi *query* pada materi ini akan menggunakan metode *Dynamic Approach* dimana metode ini memanfaatkan pemecahan *query* menggunakan Algoritma *INGRES*.

ALGORITMA INGRES

Algoritma INGRES menggunakan algoritma optimasi dinamis yang memecah query kalkulus ke dalam bagian yang lebih kecil secara rekursif.

Mengombinasikan dua tahapan yaitu dekomposisi dan optimasi kalkulus-aljabar.

TAHAPAN ALGORITMA INGRES

- Query diurai menjadi query yang berurutan dan memiliki relasi unik.
- Apabila ketika menguraikan dapat menghasilkan monorelasi query maka monorelasi query tersebut dijadikan indeks awal untuk acuan pecahan query selanjutnya.
- Monorelasi query sendiri adalah query yang bersifat operasi unary yang hasilnya didapat dari satu tabel.
- Apabila tidak terdapat monorelasi query, maka pemecahan query dilakukan secara berurutan tanpa acuan.

INGRES QUERY PROCESSOR :

q : SELECT $R_2.A_2, R_3.A_3, \ldots, R_n.A_n$
FROM $R_1, R_2, \ldots R_n$
WHERE $P_1(R_1.A'_1)$
AND
$P_2(R_1.A_1, R_2.A_2, \ldots, R_n.A_n)$

q' : SELECT $R_1.A_1$ INTO R'_1
FROM R_1
WHERE $P_1(R_1.A'_1)$

q'': SELECT $R_2.A_2, \ldots, R_n.A_n$
FROM $R'_1, R_2, \ldots, R_n$
WHERE $P_2(V_1.A_1, \ldots, V_n.A_n)$

Menampilkan nama dari karyawan yang bekerja pada proyek CAD/CAM

q_1 : SELECT EMP.ENAME
FROM EMP,ASG,PROJ
WHERE EMP.ENO=ASG.ENO
AND ASG.PNO=PROJ.PNO
AND PNAME="CAD/CAM"

q_{11} : SELECT PROJ.PNO INTO JVAR
FROM PROJ
WHERE PNAME="CAD/CAM"

q': SELECT EMP.ENAME
FROM EMP,ASG, JVAR
WHERE EMP.ENO=ASG.ENO
AND ASG.PNO=JVAR.PNO

q_{12} : SELECT ASG.ENO INTO GVAR
FROM ASG, JVAR
WHERE ASG.PNO=JVAR.PNO

q_{13} : SELECT EMP.ENAME
FROM EMP,GVAR
WHERE EMP.ENO=GVAR.ENO

ALGORITMA SISTEM R

Sistem R merupakan optimasi query statis yang didasarkan pada exhaustive search dari ruang solusi yang ada. Input untuk optimizer dengan sistem R adalah pohon relasi aljabar yang dihasilkan dari dekomposisi dari sebuah query SQL. Kemudian output dari sistem merupakan rencana eksekusi yang mengimplementasikan pohon relasi aljabar yang optimal.

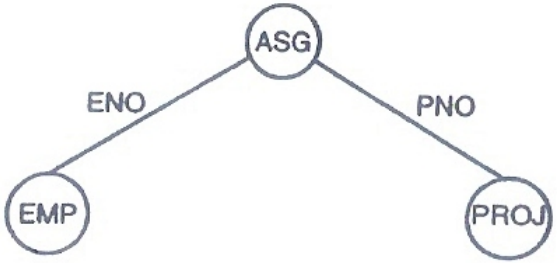
Optimizer menetapkan biaya (dalam hal waktu) dari masing-masing pohon kandidat dan mempertahankan pohon dengan biaya yang terkecil.

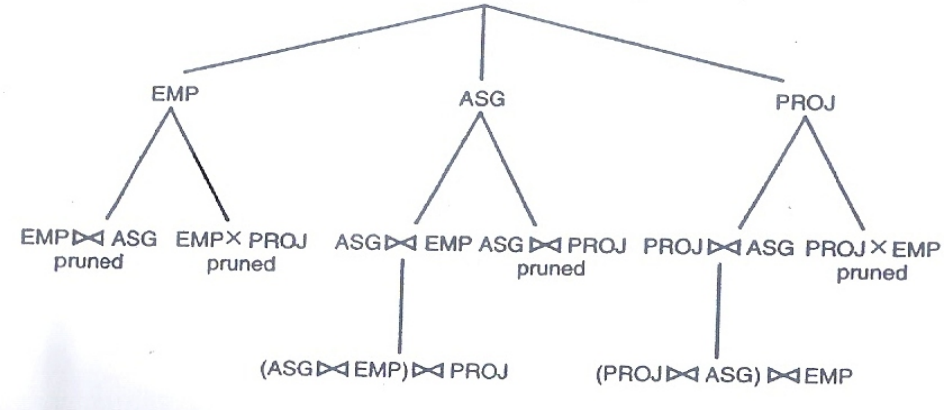
ALGORITMA SISTEM R

STMIK Amik Riau

. Contoh :

```
SELECT   EMP.ENAME
FROM     EMP,ASG,PROJ
WHERE    EMP.ENO=ASG.ENO
AND      ASG.PNO=PROJ.PNO
AND      PNAME=''CAD/CAM''
```





Alternative Join Orders

JOIN ORDERING

- Join ordering merupakan aspek yang penting dalam suatu optimasi query, terlebih pada query terdistribusi. Join ordering menjadi lebih penting ketika query merupakan query terdistribusi karena akan mempengaruhi communication time.

- Join ordering dapat dilakukan secara langsung, maupun dengan mengombinasikan dengan semijoins di mana akan meminimalisasi communication cost.

STMIK
Amik Riau

#03

OPTIMASI QUERY TERDISTRIBUSI

ALGORITMA INGRES TERDISTRIBUSI

Fungsi objektif dari algoritma ini adalah untuk meminimalisasi kombinasi baik communication time maupun response time. Algoritma ini juga mendapat keuntungan pada fragmentasi, tetapi hanya fragmentasi horizontal.

Input untuk algoritma pemrosesan query ini dinyatakan dalam calculus relational-tupel dan skema informasi (tipe network, lokasi dan ukuran masing-masing site).

ALGORITMA INGRES TERDISTRIBUSI

STMIK Amik Riau

Contoh :

Site 1	Site 2
EMP_1 ASG	EMP_2 PROJ

Beberapa strategi yang memungkinkan :

- Eksekusi keseluruhan query (EMP ASG PROJ) dengan memindah EMP_1 dan ASG ke Site 2
- Eksekusi (EMP ASG) PROJ dengan memindah (EMP_1 ASG) dan ASG ke Site 2, dll

Pemilihan di antara kemungkinan strategi-strategi trsebut membutuhkan estimasi ukuran dari hasil intermediate. Misal, jika ukuran (EMP_1 ASG) > ukuran (EMP_1), maka strategi 1 lebih dipilih daripada strategi 2.

ALGORITMA R*

Algoritma R* merupakan substansial ekstensi dari teknik yang dikembangkan untuk optimizer sistem R. Algoritma ini menggunakan pendekatan kompilasi di mana suatu exhaustive search dari seluruh alternative strategi dilakukan untuk memilih salah satu dengan biaya terendah.

Algoritma Sistem R* tidak mendukung pengimplementasian baik dalam fragmentasi maupun replikasi. Input untuk algoritma ini adalah query yang sudah dilokalisasi yang direpresentasikan dalam pohon relasi aljabar(query tree), lokasi relasi, dan statistiknya.

ALGORITMA R*

Contoh :

Query yang terdiri dari penggabungan dari relasi PROJ, eksternal relasi, dan ASG, internal relasi pada PNO. Kita asumsikan bahwa PROJ dan ASG disimpan dalam site yang berbeda dan terdapat index pada atribut PNO untuk relasi ASG. Strategi eksekusi yang memungkinkan untuk query tersebut :

1. Mengirimkan keseluruhan PROJ ke site dari ASG
2. Mengirimkan keseluruhan ASG ke site dari PROJ
3. Mengambil tupel ASG yang dibutuhkan untuk masing-masing tupel PROJ
4. Memindah ASG dan PROJ ke site ketiga

Algoritma R* memprediksi total waktu dari masing-masing strategi dan memilih yang paling sedikit.

ALGORITMA SDD-1

Algoritma SDD-1 berasal dari metode yang disebut sebagai algoritma ‘hill-climbing’, yang memiliki keistimewaan sebagai algoritma pemrosesan query terdistribusi yang pertama. Dalam algoritma ini, perbaikan dari solusi layak awal dihitung secara rekursif sampai tidak ada lagi perbaikan biaya yang dapat dilakukan. Input untuk algoritma ini termasuk query graph, lokasi relasi, dan statistik dari relasi.

ALGORITMA SDD-1

Contoh :

Asumsikan bahwa T_{MSG} = 0 dan T_{TR} = 1.

Relation	Size	Site
EMP	8	1
PAY	4	2
PROJ	1	3
ASG	10	4

Solusi layak awal :

ES_0 : EMP □ site 4
PAY □ site 4
PROJ □ site 4
Total.cost(ES_0) = 4+8+1=13

Alternatif Splitting :

ES1 : PAY □ site 1
ES2 : (PAY EMP) □ site 4
ES3 : PROJ □ site 4
Total.cost(ES') = 4+8+1 = 13

PERBEDAAN DARI KETIGA ALGORITMA TERSEBUT :

Algoritma	Optimization Timing	Objective Function	Optimization Factors	Network Topology	Semijoins	Stats	Fragments
Distributed INGRES	Dynamic	Response time or total cost	Msg. size, proc. cost	General or broadcast	No	1	Horizontal
R*	Static	Total cost	#Msg, Msg size, IO, CPU	General or local	No	1,2	No
SDD-1	Static	Total cost	Msg size	General	Yes	1,3,4,5	No

1 = relation cardinality
2 = number of unique values per attribute
3 = join selectivity factor
4 = size of projection on each join attribute
5 = attribute size and tuple size

STMIK Amik Riau

THANKS!

Ada Pertanyaan?
Boleh juga ke
WA
FB
IG