Query Optimization

Latar Belakang Masalah :
Data yang disimpan dalam database semakin besar ukuran dan volumenya, sehingga perlu didukung dengan kecepatan akses.

Keandalan sistem database (DBMS) dapat diketahui dari cara pengoptimalnya bekerja dalam memproses pernyataan SQL.

Query optimization adalah proses pemilihan rencana evaluasi-permintaan yang paling efisien dari banyak strategi untuk memproses kueri, terutama jika kueri rumit.

Query optimization bertujuan :
1. Mengurangi sebanyak mungkin tupel yang tidak dibutuhkan.
2. Memperbaiki strategi evaluasi query untuk membuat evaluasi lebih efektif.
3. Menemukan jalur akses yang paling efisien untuk meminimalkan total waktu proses kueri.

Query Optimization

Beberapa upaya sederhana untuk meningkatkan keefektifan DBMS:

1. **Creating Index**
   Indeks dapat meningkatkan kecepatan pada pencarian record. Dalam membuat indeks, field kunci indeks ditentukan berdasarkan bidang yang sering diakses oleh clauses : *Where, Join, Order By, Group By.*

   Create Index index_name on table_name (key_field)

   Contoh :
   Query : Menampilkan nama pegawai 'Smith'

   Sql : Select Ename from emp where ename='Smith';

   Membuat Index dengan key field 'ename' :

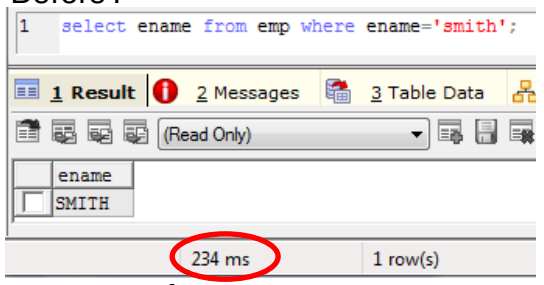
   Create index idx_ename on emp (ename);

   Selanjutnya, tuliskan SQL berikut dan lihat hasilnya.

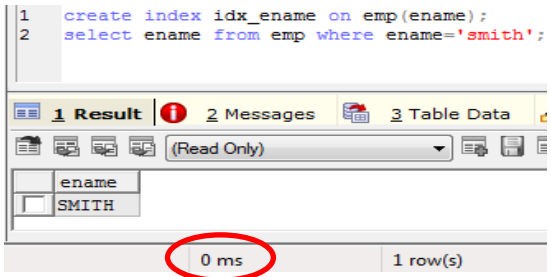
   Sql : Select Ename from emp where ename='Smith';

The execution result of the sql after creating the index can increase the speed. access time can be seen in the bottom line :

Before :



after :



Query Optimization

2. **Determining the Right Data Type**
   Misalnya menggunakan Tipe data **char** atau **varchar**. Kedua tipe data tersebut adalah tipe data karakter. **char** Menyediakan ukuran storage yang tetap (fixed-length), sedangkan **varchar** menyediakan ukuran storage sesuai dengan isi data (variable-length).

   The **char** data type is used for fields that have a consistent data length, such as postal code. While **varchar** is used for fields that have a variable length of data.

3. **Avoiding Fields of Null Value**
   Nilai null kadang-kadang membingungkan dalam interpretasi programmer dan dapat menyebabkan kesalahan logika pemrograman. Selain itu, field bernilai null mengkonsumsi lebih banyak memori.

Query Optimization

4. **Avoiding the SELECT * operations**
   Jika hanya ingin mengakses field tertentu, lebih baik untuk menulis field yang ingin diakses saja, sehingga query menjadi bidang SELECT field1, field2, field3, fieldn. Ini akan berdampak pada pengurangan beban lalu lintas jaringan dan penguncian tabel, terutama jika tabel memiliki beberapa bidang.

5. **Restrict the ORDER BY operation**
   Penggunaan operasi ORDER BY yang berfungsi untuk mengurutkan data. Ini memiliki konsekuensi untuk meningkatkan beban query, karena akan menambah satu proses lagi, yaitu proses penyortiran.

6. Using ***Subquery*** *or* ***JOIN***
   Disarankan untuk memprioritaskan penggunaan operasi GABUNG, karena dalam kasus umum akan menghasilkan kinerja yang lebih cepat. Tetapi jika dalam kasus-kasus tertentu hanya bisa diselesaikan dengan subquery, maka gunakan subquery.

Query Optimization

7. Using ***WHERE in SELECT***
WHERE clause is used as a condition to filter records so that it can minimize the load of data traffic, because not all records are accessed.

8. **Consider the operator's access speed**
Urutan operator yang diproses tercepat adalah =, >, >=, <, <=, Like, <>.

Contoh :
gunakan ekspresi logika WHERE 1 = 1 daripada WHERE 0 <> 1

Query Optimization

langkah-langkah berikut adalah proses optimasi query yang hanya mempertimbangkan cost paling minimum (jumlah tupel dan kolom paling minimum):

1. Menentukan Query Plan
   Query Plan ditulis dengan menggunakan aljabar relasional. Pembuatan Query Plan didasarkan pada pohon permintaan (pohon ekspresi) yang berisi urutan operasi relasional.

2. Transformasi query plan menggunakan equivalence rules
   Rules equivalence digunakan untuk mentransformasikan query plan kedalam optimize query :
   a. Membuat query tree (expression tree)
   b. Menentukan relational algebra
   c. Mengevaluasi output dan jumlah record and kolom yang terkait
   d. Mengulangi langkah sampai query yang optimal diperoleh

Query Optimization

Equivalence Rules

Equivalence Rules : dua bentuk ekspresi yang berbeda adalah setara. Ekspresi bentuk pertama dapat diganti dengan ekspresi bentuk kedua atau sebaliknya . Karena dua ekspresi menghasilkan hasil output yang sama.

1. Operasi seleksi konjungtif dapat didekonstruksi menjadi urutan seleksi individu.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Operasi SELEKSI adalah **commutative**

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Hanya operasi terakhir dalam urutan operasi proyeksi yang diperlukan yang lain dapat dihilangkan

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) = \Pi_{L_1}(E)$$

Query Optimization

4. Operasi SELEKSI dapat dikombinasi dengan Cartesian products dan Theta join
   a. $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$

      This expression is just the definition of the theta join.

   b. $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

5. Operasi Theta Join adalah **commutative**

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

6. Operasi Natural-join adalah **associative**

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

7. Operasi himpunan union dan intersection adalah **commutative**.

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_1$$

8. Operasi himpunan Set Difference adalah **Not commutative**.

Query Optimization

9. Operasi Himpunan union dan intersection adalah **associative**.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

10. Pada operasi Selection berlaku :

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

11. Pada projection berlaku :

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Query Optimization

Contoh :

Teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Course

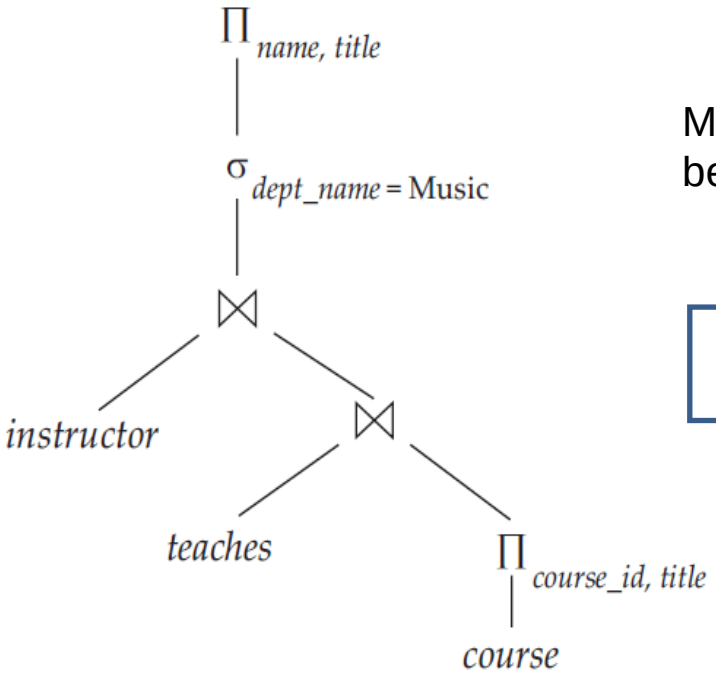
course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Query :

“Menampilkan **name** semua instruktur di departemen Musik bersama dengan **title** dari semua mata kuliah yang diajarkan instruktur tersebut.”

Query Optimization

Langkah 1 :
Buat query tree (expression tree) sebagai dasar menentukan query plan



Menentukan **query plan** menggunakan relational algebra berdasarkan expression tree

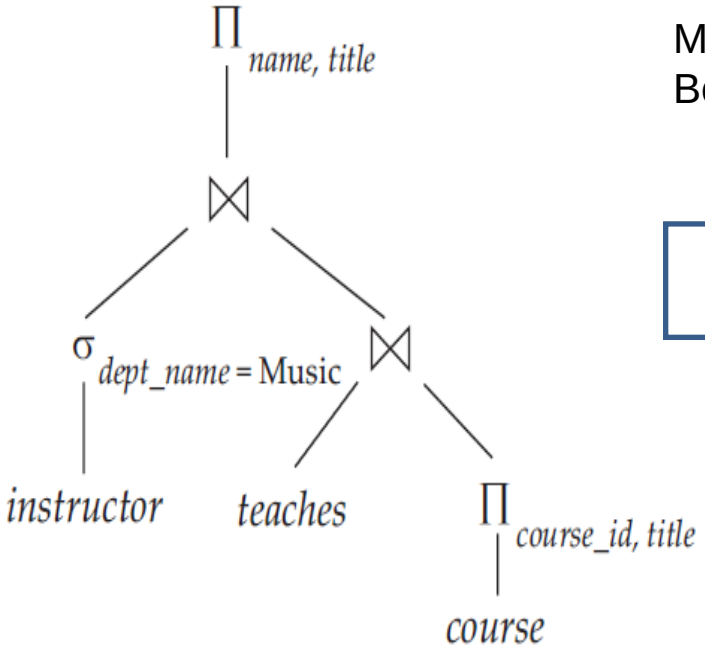
Relational Algebra :

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

expression tree awal

Query Optimization

Langkah 2 :
Transformasikan query plan kedalam optimize query dengan menggunakan Equivalence Rules.
Dalam kasus ini, query plan dapat dioptimasi menggunakan equivalence rules no. 6 (Natural-join operations are **associative**).



Transformed expression tree

Menentukan query plan menggunakan relational algebra
Berdasarkan expression tree

Relational Algebra :

$\pi_{name, title}((\sigma_{dept\_name = \text{“Music”}}(instructor)) \bowtie (teaches \bowtie \pi_{course\_id, title}(course)))$

Langkah Selanjutnya :
Analisis proses dan output dari query plan dan optimize query.

Query Optimization

Analisis Proses dan Output query plan dan optimize query

Query plan :

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

Output :

$\pi_{course\_id,title}(course)$

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

$teaches \bowtie \pi_{course\_id,title}(course)$

course_id	ID	sec_id	semester	year	title
CS-101	10101	1	Fall	2009	Intro to Computer Science
CS-101	45565	1	Spring	2010	Intro to Computer Science
CS-190	83821	1	Spring	2009	Game Design
CS-190	83821	2	Spring	2009	Game Design
CS-315	10101	1	Spring	2010	Robotics
CS-319	45565	1	Spring	2010	Image Processing
CS-319	83821	2	Spring	2010	Image Processing
CS-347	10101	1	Fall	2009	Database System Concepts
EE-181	98345	1	Spring	2009	Intro to Digital System
FIN-201	12121	1	Spring	2010	Invesment Banking
HIS-351	32343	1	Spring	2010	World History
MU-199	15151	1	Spring	2010	Music Video Production
PHY-101	22222	1	Fall	2009	Physical Principles

$(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
76766	Crick	Biology	72000.00	BIO-101	1	Summer	2009	Intro to Biology
76766	Crick	Biology	72000.00	BIO-301	1	Summer	2010	Genetics
10101	Srinivasan	Comp. Sci.	65000.00	CS-101	1	Fall	2009	Intro to Computer Science
45565	Katz	Comp. Sci.	75000.00	CS-101	1	Spring	2010	Intro to Computer Science
83821	Brandt	Comp. Sci.	92000.00	CS-190	1	Spring	2009	Game Design
83821	Brandt	Comp. Sci.	92000.00	CS-190	2	Spring	2009	Game Design
10101	Srinivasan	Comp. Sci.	65000.00	CS-315	1	Spring	2010	Robotics
45565	Katz	Comp. Sci.	75000.00	CS-319	1	Spring	2010	Image Processing
83821	Brandt	Comp. Sci.	92000.00	CS-319	2	Spring	2010	Image Processing
10101	Srinivasan	Comp. Sci.	65000.00	CS-347	1	Fall	2009	Database System Concepts
98345	Kim	Elec. Eng.	80000.00	EE-181	1	Spring	2009	Intro to Digital System
12121	Wu	Finance	90000.00	FIN-201	1	Spring	2010	Invesment Banking
32343	El Said	History	60000.00	HIS-351	1	Spring	2010	World History
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production
22222	Einstein	Physics	95000.00	PHY-101	1	Fall	2009	Physical Principles

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

name	title
Mozart	Music Video Production

Query Optimization

Analisis Proses dan Output query plan dan optimize query

The Optimize Query :

π *name, title* ((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ π *course_id,title* (*course*)))

Output :

π *course_id,title* (*course*)

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

σ *dept_name =“Music” (instructor)*

	ID	name	dept_name	salary
☐	15151	Mozart	Music	40000.00

((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ π *course_id,title* (*course*)))

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

teaches ⋈ π *course_id,title* (*course*)

course_id	ID	sec_id	semester	year	title
CS-101	10101	1	Fall	2009	Intro to Computer Science
CS-101	45565	1	Spring	2010	Intro to Computer Science
CS-190	83821	1	Spring	2009	Game Design
CS-190	83821	2	Spring	2009	Game Design
CS-315	10101	1	Spring	2010	Robotics
CS-319	45565	1	Spring	2010	Image Processing
CS-319	83821	2	Spring	2010	Image Processing
CS-347	10101	1	Fall	2009	Database System Concepts
EE-181	98345	1	Spring	2009	Intro to Digital System
FIN-201	12121	1	Spring	2010	Invesment Banking
HIS-351	32343	1	Spring	2010	World History
MU-199	15151	1	Spring	2010	Music Video Production
PHY-101	22222	1	Fall	2009	Physical Principles

π*name, title* ((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ π *course_id,title* (*course*)))

name	title
Mozart	Music Video Production

Query Optimization

Hasil analisis :

analisis :

Query plan dan optimize query menghasilkan output yang sama

Analisis Proses :

Operasi natural join dari optimize query menghasilkan jumlah record yang lebih kecil daripada operasi natural join operation dari query plan.

Query plan

$(instructor \bowtie (teaches \bowtie \pi_{course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
76766	Crick	Biology	72000.00	BIO-101	1	Summer	2009	Intro to Biology
76766	Crick	Biology	72000.00	BIO-301	1	Summer	2010	Genetics
10101	Srinivasan	Comp. Sci.	65000.00	CS-101	1	Fall	2009	Intro to Computer Science
45565	Katz	Comp. Sci.	75000.00	CS-101	1	Spring	2010	Intro to Computer Science
83821	Brandt	Comp. Sci.	92000.00	CS-190	1	Spring	2009	Game Design
83821	Brandt	Comp. Sci.	92000.00	CS-190	2	Spring	2009	Game Design
10101	Srinivasan	Comp. Sci.	65000.00	CS-315	1	Spring	2010	Robotics
45565	Katz	Comp. Sci.	75000.00	CS-319	1	Spring	2010	Image Processing
83821	Brandt	Comp. Sci.	92000.00	CS-319	2	Spring	2010	Image Processing
10101	Srinivasan	Comp. Sci.	65000.00	CS-347	1	Fall	2009	Database System Concepts
98345	Kim	Elec. Eng.	80000.00	EE-181	1	Spring	2009	Intro to Digital System
12121	Wu	Finance	90000.00	FIN-201	1	Spring	2010	Invesment Banking
32343	El Said	History	60000.00	HIS-351	1	Spring	2010	World History
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production
22222	Einstein	Physics	95000.00	PHY-101	1	Fall	2009	Physical Principles

Optimize query

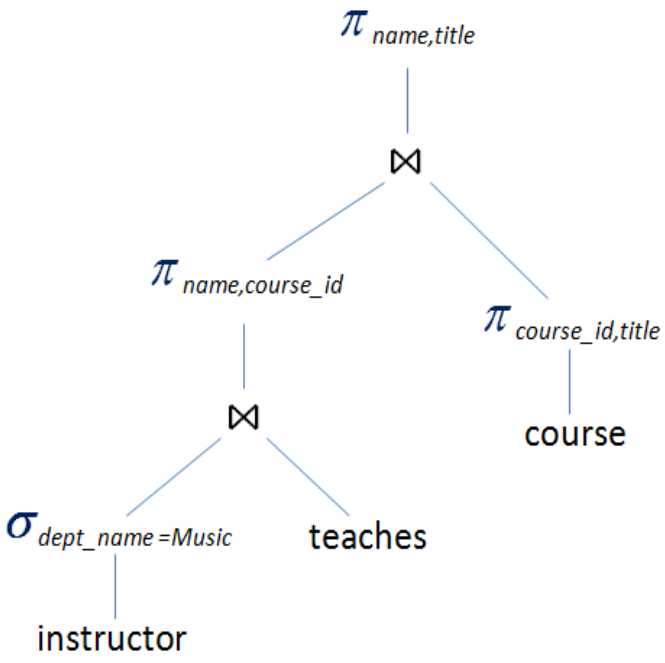
$((\sigma\ dept\_name = \text{“Music”}\ (instructor)) \bowtie (teaches \bowtie \pi_{course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

Hasil query di atas membutuhkan proses optimasi dengan mengurangi kolom tidak yang diperlukan (dept_name, gaji, dll).

Query Optimization

Langkah selanjutnya :
Proses optimasi query digunakan untuk mengurangi kolomyang tidak diperlukan. Oleh karena itu, expression tree diformulasikan menjadi :



expression tree yang ditransformasikan

Menentukan optimize query menggunakan relational algebra berdasarkan expression tree

Relational Algebra :

$\pi_{name, title}((\pi_{name,course\_id}((\sigma_{dept\_name = \text{“Music”}}(instructor)) \bowtie teaches) \bowtie \pi_{course\_id,title}(course))$

Langkah selanjutnya :
Analisis proses dan output dari optimize query sebelumnya dan optimize query.

Query Optimization

Analisis Proses dan Output optimize query

π name, title ((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi \text{ course\_id,title}}$ (course))

Output :

π course_id,title (course)

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

σ dept_name ="Music" (instructor)

	ID	name	dept_name	salary
☐	15151	Mozart	Music	40000.00

π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches)

name	course_id
Mozart	MU-199

((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi \text{ course\_id,title}}$ (course))

course_id	name	title
MU-199	Mozart	Music Video Production

π name, title ((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi \text{ course\_id,title}}$ (course))

name	title
Mozart	Music Video Production

Query Optimization

Hasil Analisis :

Analisis Output :

Optimize query sebelumnya dan optimize query menghasilkan output yang sama

Analisis proses:

Operasi natural join dari optimize query menghasilkan jumlah kolom yang lebih kecil daripada operasi natural join dari optimize query sebelumnya.

Hasil dari optimize query sebelumnya

$((\sigma dept\_name = \text{“Music”} (instructor)) \bowtie (teaches \bowtie \pi_{course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

Kolom-kolom yang tidak diperlukan

Hasil dari optimize query

$((\pi name,course\_id((\sigma dept\_name = \text{“Music”} (instructor)) \bowtie teaches) \bowtie \pi_{course\_id,title} (course))$

course_id	name	title
MU-199	Mozart	Music Video Production

Kolom yang tidak diperlukan

Query Optimization

Latihan (menggunakan skema database university) :

Tulislah query yang paling optimal menggunakan langkah-langkah proses query optimization :

1. Realtional Algebra :

   $\pi_{name}(\sigma_{title = \text{“Game Design”}}(Student \bowtie takes \bowtie course))$

2. Menampilkan **name** semua instruktur pada department ‘Music’ bersama dengan **course title** semua matakuliah yang instrukturnya mengajar pada tahun 2009.
3. Menampilkan **name** semua student dan **course title** pada department ‘Comp. Sci.’.
4. Menampilkan **name** semua student, **course title** dan **name** building pada Department ‘Comp. Sci.’.