



**JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA YOGYAKARTA**

**MODUL 7
BAHASA QUERY**

Menampilkan Data dari Beberapa Tabel (*Multiple Tabel*) Bagian II

A. TUJUAN

1. Mahasiswa dapat menulis perintah SELECT untuk mengakses data dari beberapa tabel menggunakan ***natural join***, ***USING clause***, ***ON clause***.
2. Memahami ***Cross join*** (Cartesian Product)

B. LANDASAN TEORI & LANGKAH PRAKTIKUM

Pada modul 6 yang lalu telah dibahas tentang mengakses data dari beberapa tabel menggunakan *equijoin*, *outer join*, dan *self join*. Pada modul ini akan dibahas mengakses data dari beberapa tabel menggunakan *natural join*, *USING clause*, *ON clause*, *cross join*, dan *non equijoin*. Tabel-tabel yang digunakan dalam praktikum ini dapat dilihat pada LAMPIRAN.

Sintaks umum untuk join lebih dari satu tabel :

```
SELECT  table1.column, table2.column
FROM    table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

Keterangan :

- **Table1.column** menunjukkan tabel dan kolom dimana data diambil.
- **NATURAL JOIN** merelasikan (*join*) dua tabel berdasar nama kolom yang sama dan tipe data yang sama.
- **JOIN table USING column_name** merelasikan (*equijoin*) dua tabel berdasar nama kolom yang sama namun tipe data dapat berbeda.
- **JOIN table ON table1.column_name** merelasikan (*equijoin*) berdasar pada kondisi dalam ON clause = **table2.column_name**.

- **LEFT/RIGHT/FULL OUTER** digunakan untuk melakukan *outer joins*.
- **CROSS JOIN** menghasilkan suatu Cartesian Product dari dua tabel.

Sintaks umum tersebut merupakan SQL:1999 standar yang dapat digunakan mulai Oracle 9i ke atas (Oracle 10g, Oracle 11g). Sebelum Oracle 9i (yaitu Oracle 8i ke bawah) bentuk sintaks *join* berbeda dari standard ANSI tersebut (lihat bentuk sintaks *join* pada modul 6 pokok bahasan B.2 halaman 2). Perlu diketahui bahwa sintaks *join* dalam SQL 1999 standard tidak menawarkan unjuk kerja yang lebih baik dibanding sintaks *join* pada versi sebelumnya (Oracle 8i ke bawah).

1. Natural Join

Dalam Oracle 10g dimungkinkan *join* dilakukan secara otomatis tanpa menspesifikasikan secara eksplisit kolom yang menghubungkan 2 tabel. **Keyword NATURAL JOIN** digunakan untuk melakukan *join* secara otomatis berdasar kolom dalam dua tabel yang memiliki nama yang sama dan tipe data yang sama. Bila nama kolom sama namun tipe data tidak sama, maka akan terjadi *error*.

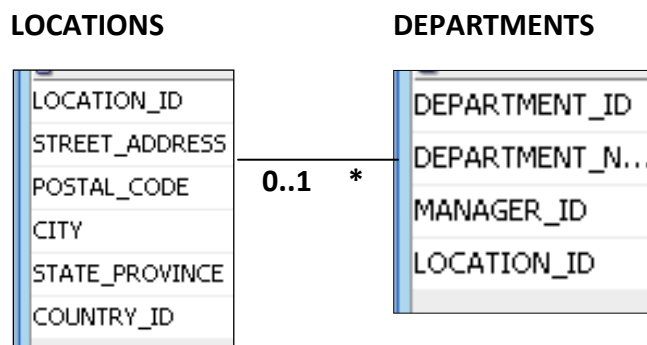
Contoh :

```
SELECT department_id, department_name, location_id, city
FROM departments
NATURAL JOIN locations;
```

Hasil :

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	30	Purchasing	1700	Seattle
5	90	Executive	1700	Seattle
6	100	Finance	1700	Seattle
7	110	Accounting	1700	Seattle
8	120	Treasury	1700	Seattle
9	130	Corporate Tax	1700	Seattle
10	140	Control And Credit	1700	Seattle
11	150	Shareholder Services	1700	Seattle
12	160	Benefits	1700	Seattle
13	170	Manufacturing	1700	Seattle
14	180	Construction	1700	Seattle
15	190	Contracting	1700	Seattle
16	200	Operations	1700	Seattle

Pada contoh di atas, tabel LOCATION direlasikan (*join*) dengan tabel DEPARTMENT menggunakan kolom **location_id** yang merupakan kolom dengan nama yang sama dari kedua tabel tersebut (lihat struktur tabel di bawah ini).



Untuk membatasi data yang ditampilkan, NATURAL JOIN dapat diimplementasikan dalam klausa WHERE.

Contoh :

```

SELECT department_id, department_name, location_id, city
FROM departments
NATURAL JOIN locations
WHERE department_id IN (20,50);
  
```

Results:				
	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	20	Marketing	1800	Toronto
2	50	Shipping	1500	South San Francisco

2. Membuat JOIN menggunakan klausa USING

Seperti telah disebutkan di atas bahwa NATURAL JOIN merelasikan menggunakan kolom dengan nama yang sama dan tipe data yang sama. Jika kolom (*field*) yang akan direlasikan memiliki nama yang sama namun memiliki tipe data yang berbeda, kita dapat menggunakan klausa USING sebagai ganti dari NATURAL JOIN.

Contoh :

```

SELECT employees.employee_id, employees.last_name,
       departments.location_id, department_id
FROM employees JOIN departments
USING (department_id);
  
```

Hasil :

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	114	Raphaely	1700	30
5	115	Khoo	1700	30
6	116	Baida	1700	30
7	117	Tobias	1700	30
8	118	Himuro	1700	30
9	119	Colmenares	1700	30
10	203	Mavris	2400	40

Ketentuan : Jangan menggunakan nama prefix tabel saat menggunakan klausa USING. Perhatikan dan cobalah kedua contoh di bawah ini !

Contoh benar :

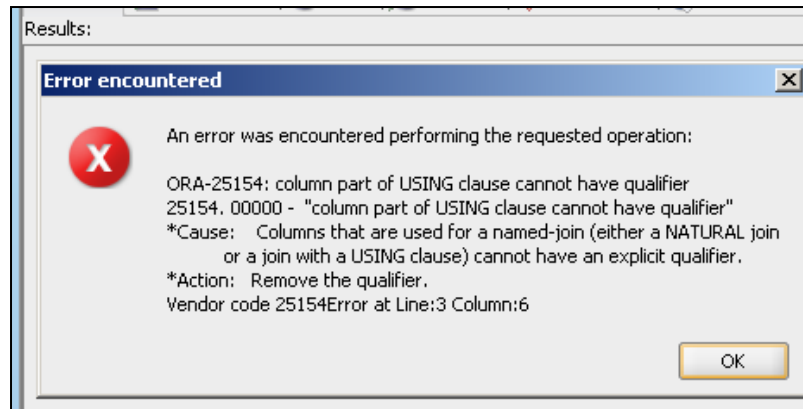
```
SELECT l.city, d.department_name
FROM locations l JOIN departments d USING (location_id)
WHERE location_id = 1400;
```

Results:	
CITY	DEPARTMENT_NAME
1 Southlake	IT

Contoh salah :

```
SELECT l.city, d.department_name
FROM locations l JOIN departments d USING (location_id)
WHERE d.location_id = 1400;
```

Statemen tersebut salah karena kolom `location_id` memiliki qualifier (nama prefix tabel), sehingga muncul pesan error : *column part of USING clause cannot have qualifier*.



Penggunaan nama alias tabel dilakukan dengan cara sebagai berikut :

```
SELECT e.employee_id, e.last_name,
       d.location_id, department_id
FROM   employees e JOIN departments d
USING (department_id);
```

Results:				
	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	114	Raphaely	1700	30
5	115	Khoo	1700	30
6	116	Baida	1700	30
7	117	Tobias	1700	30
8	118	Himuro	1700	30
9	119	Colmenares	1700	30
10	203	Mavris	2400	40

Perhatikan bahwa pada kolom department_id tidak boleh menggunakan nama prefix tabel maupun nama alias tabel.

Nama alias tabel disarankan untuk digunakan karena bermanfaat untuk:

- mempersingkat penulisan query
- meningkatkan unjuk kerja (*performance*) *server database* dalam menemukan tabel yang dimaksud.

Aturan penulisan nama alias tabel, lihat kembali modul 6.

3. Membuat JOIN menggunakan klausa ON

Join tabel dapat pula dilakukan menggunakan klausa ON. Keuntungannya adalah klausa ON membuat *coding (query)* lebih mudah dipahami.

Contoh :

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON (e.department_id = d.department_id);
```

Hasil :

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	114	Raphaely	30	30	1700
5	115	Khoo	30	30	1700
6	116	Baida	30	30	1700
7	117	Tobias	30	30	1700
8	118	Himuro	30	30	1700
9	119	Colmenares	30	30	1700
10	203	Mavris	40	40	2400

Pada contoh di atas, kolom **department_id** pada tabel EMPLOYEES dan tabel DEPARTMENTS direlasikan (*join*) menggunakan klausa ON. Struktur kedua tabel tersebut dapat dilihat pada LAMPIRAN.

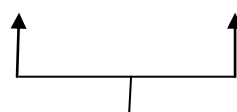
Klausa ON dapat pula digunakan untuk merelasikan kolom dengan nama yang berbeda. Sebagai contoh pada *self-joins* berikut ini :

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
105	Austin	103
106	Pataballa	103
107	Lorentz	103
108	Greenberg	101
109	Faviet	108
110	Chen	108

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
105	Austin
106	Pataballa
107	Lorentz
108	Greenberg
109	Faviet
110	Chen



MANAGER_ID di tabel WORKER sama dengan
EMPLOYEE_ID pada tabel MANAGER



Gambar 7.1. Self Join

Terdapat 2 kolom dengan nama yang berbeda yaitu **manager_id** dan **employee_id** namun kita tahu bahwa **manager_id** pada tabel **WORKER** sama dengan **employee_id** pada tabel **MANAGER**.

Contoh *self-join* menggunakan klausa ON :

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON (e.manager_id = m.employee_id);
```

Hasil :

	 EMP	 MGR
1	Smith	Cambrault
2	Ozer	Cambrault
3	Kumar	Cambrault
4	Fox	Cambrault
5	Bloom	Cambrault
6	Bates	Cambrault
7	Hunold	De Haan
8	Vishney	Errazuriz
9	Marvins	Errazuriz
10	Lee	Errazuriz

Pada penggunaan klausa ON ini, dimungkinkan pula untuk menambah kondisi guna membatasi data yang ditampilkan. Cara pertama adalah dengan menambahkan klausa AND.

Contoh :

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON (e.department_id = d.department_id)
AND e.manager_id = 149;
```

Hasil :

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	175	Hutton	80	80	2500
3	179	Johnson	80	80	2500
4	177	Livingston	80	80	2500
5	176	Taylor	80	80	2500

Pada contoh di atas, dibatasi hanya ditampilkan pegawai yang memiliki manager_id = 149.

Cara kedua dapat pula dilakukan dengan menggunakan klausa WHERE.

Contoh :

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON (e.department_id = d.department_id)
WHERE  e.manager_id = 149;
```

Hasil :

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	175	Hutton	80	80	2500
3	179	Johnson	80	80	2500
4	177	Livingston	80	80	2500
5	176	Taylor	80	80	2500

Rangkuman :

Sintak tanpa JOIN dan USING :

```
SELECT e.employee_id, e.last_name, d.location_id
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

Sintak JOIN dengan klausa USING :

```
SELECT e.employee_id, e.last_name, d.location_id
FROM employees e
JOIN departments d
USING (department_id);
```

Sintak JOIN dengan klausa ON :

```
SELECT e.employee_id, e.last_name, d.location_id
FROM employees e
JOIN departments d
ON (e.department_id = d.department_id);
```

Membuat *Three-Way Join* dengan menggunakan klausa ON :

Three-Way Join adalah suatu relasi (*join*) yang melibatkan 3 buah tabel. *Join* dilakukan dari kiri ke kanan. Kondisi join yang pertama memakai referensi kolom dari tabel pertama dan kedua, sedangkan kondisi join yang kedua dapat memakai referensi kolom dari semua tabel (ketiga tabel) tersebut.

Contoh :

```
SELECT employee_id, city, department_name
FROM employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

Hasil :

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	105	Southlake	IT
7	106	Southlake	IT
8	107	Southlake	IT

Pada contoh di atas, kondisi join yang pertama untuk melakukan *join* antara tabel EMPLOYEES dan tabel DEPARTMENTS sedangkan kondisi join yang kedua untuk melakukan join dengan tabel LOCATIONS. Struktur ketiga tabel tersebut dapat dilihat pada lampiran.

Catatan :

Bandingkan dengan sintak tanpa JOIN (pada modul 6) :

```
SELECT e.employee_id, l.city, d.department_name
FROM employees e, departments d, locations l
WHERE d.department_id = e.department_id
AND d.location_id = l.location_id;
```

4. Cartesian Product

Cartesian Product terjadi ketika :

- tidak ada kondisi join
- kondisi join ada namun salah
- semua baris pada tabel pertama direlasikan (*join*) dengan seluruh baris pada tabel kedua.

Untuk menghindari terjadinya *Cartesian Product*, gunakan selalu kondisi join secara benar.

Contoh *Cartesian Product* :

```
SELECT last_name, department_name
FROM employees, departments;
```

EMPLOYEES (20 record)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
5	104	Ernst	60
6	105	Austin	60
7	106	Pataballa	60
8	107	Lorentz	60
9	108	Greenberg	100
10	109	Faviet	100
11	110	Chen	100
12	111	Sciarra	100
13	112	Urman	100
14	113	Popp	100
15	114	Raphaely	30
16	115	Khoo	30

DEPARTMENTS (8 record)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	30	Purchasing	1700
4	40	Human Resources	2400
5	50	Shipping	1500
6	60	IT	1400
7	70	Public Relations	2700
8	80	Sales	2500

Cartesian Product :
20 x 8 = 160 record

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Ande	Administration
3	Atkinson	Administration
4	Austin	Administration
5	Baer	Administration
6	Baida	Administration
7	Banda	Administration
8	Bates	Administration
9	Bell	Administration
10	Bernstein	Administration
11	Bissot	Administration
12	Bloom	Administration
13	Bull	Administration
14	Cabrio	Administration
15	Cambrault	Administration

Gambar 7.2. Cartesian Product

Pada contoh di atas, ditampilkan last name dari tabel EMPLOYEES dan department_name dari tabel DEPARTMENTS. Karena tidak ada kondisi join yang diberikan, maka seluruh baris (20 baris) pada tabel EMPLOYEES direlasikan (*join*) dengan seluruh baris (8 baris) pada tabel DEPARTMENTS sehingga terbentuk output dengan jumlah 160 baris.

Sebuah *Cartesian Product* terjadi karena tidak adanya kondisi *join* atau kondisi *join* ada namun salah. Hasil *Cartesian Product* jarang sekali bermanfaat, kecuali digunakan sebagai *testing* (simulasi) untuk mengetahui berapa jumlah data yang *reasonable*. Gunakan kondisi join secara benar untuk menghindari terjadinya *Cartesian Product*.

Jika memang diperlukan membuat suatu *Cartesian Product* antara kedua tabel, dapat digunakan keyword *cross join*.

Contoh :

```
SELECT last_name, department_name
FROM employees
CROSS JOIN departments;
```

Hasil :

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Ande	Administration
3	Atkinson	Administration
4	Austin	Administration
5	Baer	Administration
6	Baida	Administration
7	Banda	Administration
8	Bates	Administration
9	Bell	Administration
10	Bernstein	Administration
11	Bissot	Administration
12	Bloom	Administration
13	Bull	Administration
14	Cabrio	Administration
15	Cambrault	Administration

6. Kesimpulan (Modul 6 dan Modul 7)

- Ada banyak cara untuk merelasikan (*join*) tabel-tabel, yaitu :
 - Equijoin
 - Outer join
 - Full (or two-sided) outer join
 - Self-join
 - Natural join
 - Klausula USING
 - Klausula ON
 - Cross join (Cartesian Product)
- *Cartesian Product* terjadi karena tidak adanya kondisi *join* yang valid. Bila ingin membuat suatu *Cartesian Product* dapat digunakan keyword CROSS JOIN.
- Nama alias tabel sebaiknya digunakan, karena bermanfaat untuk :
 - Meningkatkan unjuk kerja akses ke database
 - Mempersingkat penulisan query
 - Menghemat memory

C. TUGAS

1. Buatlah query menggunakan NATURAL JOIN untuk menampilkan alamat semua departemen. Gunakan table LOCATIONS dan COUNTRIES. Tampilkan location ID, street address, city, state province dan country.

LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
1500	2011 Interiors Blvd	South San Francisco	California	United States of America
1700	2004 Charade Rd	Seattle	Washington	United States of America
1800	460 Bloor St. W.	Toronto	Ontario	Canada
2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

2. Buatlah query untuk menampilkan last_name, department_id, dan department_name untuk semua pegawai.
 - a. Gunakan JOIN dengan klausa USING
 - b. Gunakan JOIN dengan klausa ON
3. Buatlah query untuk menampilkan last_name, job, department number, dan department name untuk seluruh karyawan yang bekerja di Toronto.

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

- a. Gunakan NATURAL JOIN
 - b. Gunakan JOIN dengan klausa USING
 - c. Gunakan JOIN dengan klausa ON
4. Buatlah query untuk menampilkan last name dan number setiap employee beserta dengan last name dan number manajernya, dengan format tampilan seperti di bawah ini! Gunakan label kolom EMPLOYEE, EMP#, Manager, Mgr#. Gunakan JOIN dengan klausa ON !

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100
Hartstein	201	King	100
Whalen	200	Kochhar	101
Higgins	205	Kochhar	101

5. Buatlah query untuk menampilkan last name, department name, location ID, dan city dari employee yang mendapatkan commission. Gunakan JOIN dengan klausa ON.

D. DAFTAR PUSTAKA

Oracle Database 10g : SQL Fundamental, Oracle Inc. 2004

😊 Even a journey of a thousand miles...
must begin with the first step 😊

LAMPIRAN STRUKTUR TABEL DAN RELASI TABEL

