JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA YOGYAKARTA

MODUL 6
BAHASA QUERY

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

A. TUJUAN

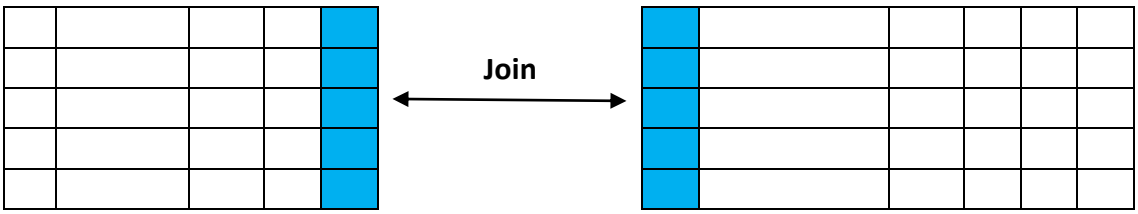
1. Mahasiswa dapat menulis perintah SELECT untuk mengakses data dari beberapa tabel (lebih dari 1 tabel) menggunakan *equijoin.*
2. Menampilkan data yang tidak memenuhi kondisi *join* dengan menggunakan *outer joins*.
3. Membuat *join* sebuah tabel dengan dirinya sendiri menggunakan *self join***.**

B. LANDASAN TEORI & LANGKAH PRAKTIKUM

Tabel-tabel yang digunakan dalam praktikum ini dapat dilihat di LAMPIRAN.

1. Memperoleh Data dari Beberapa Tabel

Kadang-kadang kita perlu menggunakan data yang berasal lebih dari 1 tabel. Untuk mengatasi hal ini, kita dapat melakukan *join* sejumlah tabel untuk memperoleh data dari beberapa tabel. Suatu *join* digunakan untuk menampilkan informasi dari beberapa tabel (lebih dari 1 tabel).



Gambar 6.1. Gunakan *Join* untuk merelasikan 2 atau lebih tabel

Macam-macam Join antara lain :

- Equijoin
- Outer join
- Full (or two-sided) outer join
- Self-join
- Natural join
- Using clause
- Arbitari join condition for outer join
- Cross join (Cartesian Product)
- Non equijoin

2. Sintak umum untuk join tabel-tabel :

```
SELECT      table1.column, table2.column
FROM        table1, table2
WHERE       table1.column1 = table2.column2;
```

Keterangan :

- **table.column**: menunjukkan tabel dan kolom di mana data diambil.
- **table1.column1 = table2.column2** : menunjukkan kondisi join antara 2 buah tabel.
- Tuliskan kondisi join pada klausa WHERE.
- Untuk join n buah tabel diperlukan paling sedikit n-1 kondisi join.

3. Equijoin

Laporan di bawah ini menampilkan data yang berasal dari 2 buah tabel yang berbeda yaitu :

- Employee_ID berasal dari tabel EMPLOYEES
- Department_ID ada di tabel EMPLOYEES maupun tabel DEPARTMENTS.
- Department_name berasal dari tabel DEPARTMENTS

EMPLOYEES

	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

	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
9	90	Executive	1700
10	100	Finance	1700
11	110	Accounting	1700
12	120	Treasury	1700
13	130	Corporate Tax	1700
14	140	Control And Credit	1700
15	150	Shareholder Services	1700
16	160	Benefits	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	114	30	Purchasing
5	115	30	Purchasing
6	116	30	Purchasing
7	117	30	Purchasing
8	118	30	Purchasing
9	119	30	Purchasing
10	203	40	Human Resources

Gambar 6.2. Memperoleh Data dari Beberapa Tabel

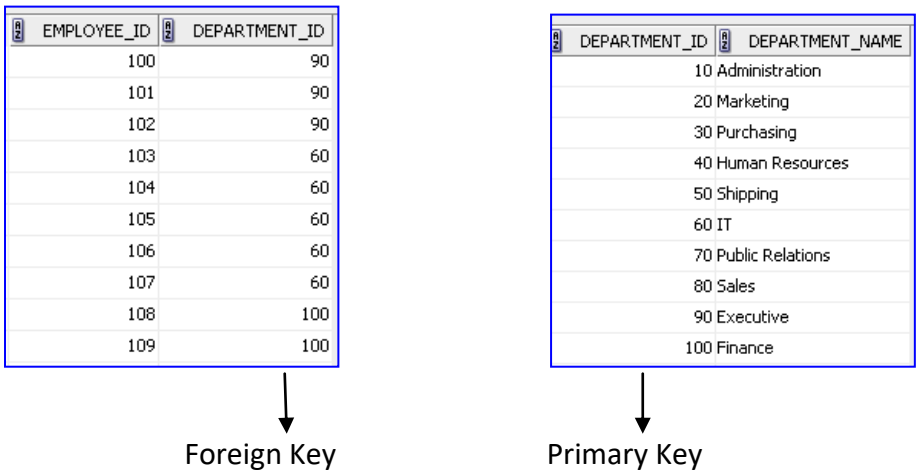
Untuk menghasilkan laporan seperti tersebut di atas, kita perlu merelasikan tabel EMPLOYEES dan DEPARTMENTS agar dapat mengakses data dari kedua tabel tersebut, dengan query sebagai berikut :

```
SELECT employees.employee_id, employees.department_id,
       departments.department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

Keterangan :

- Klausa SELECT menspesifikasikan nama kolom yang diambil, yaitu :
  - employee_id, department_id dari tabel EMPLOYEES
  - department_name dari tabel DEPARTMENTS
- Klausa FROM menspesifikasikan tabel yang diakses :
  - Tabel EMPLOYEES
  - Tabel DEPARTMENTS
- Klausa WHERE menspesifikasikan bagaimana kedua tabel direlasikan(*join*):
  - employees.department_id = departments.department_id;

Baris dari satu tabel dapat dihubungkan dengan tabel lain menggunakan kolom Primary Key dan Foreign Key. Pada contoh di atas, department_id pada tabel EMPLOYEES berfungsi sebagai Foreign Key sedangkan department_id pada table DEPARTMENTS berfungsi sebagai Primary Key.



Gambar 6.3. Equijoin / Simple Join/ Inner Join

Untuk memperoleh nama department dari setiap pegawai, dibandingkan nilai kolom **department_id** *(foreign key)* pada **tabel EMPLOYEES** dengan nilai nilai kolom **department_id** *(primary key)* pada **tabel DEPARTMENTS**. Hubungan antara tabel EMPLOYEES dan tabel DEPARTMENTS adalah sebuah ***equijoin*** yaitu nilai department_id di kedua tabel harus sama. *Equijoin* juga sering disebut *simple join* atau *inner join*.

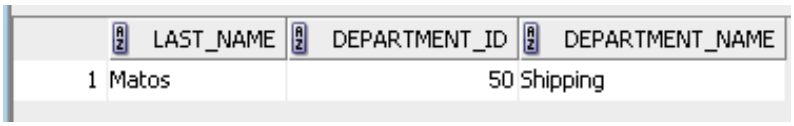
4. Penambahan Kondisi Pencarian Menggunakan Operator AND

Pada join tabel, anda dapat menambah kriteria pencarian pada klausa WHERE untuk membatasi baris pada tabel dengan menggunakan operator AND.

Contoh :
Untuk menampilkan nomor departemen dan nama departemen dari pegawai bernama Matos, maka diperlukan penambahan kondisi pada klausa WHERE sebagai berikut :

```
SELECT employees.last_name, employees.department_id,
        departments.department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND employees.last_name = 'Matos';
```

Hasil :



	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

5. Menggunakan Alias Tabel

Menuliskan nama kolom dengan awalan (prefix) nama tabel dapat sangat menghabiskan waktu, khususnya jika nama tabelnya panjang. Gunakan nama alias tabel untuk mempersingkat penulisan query. Gunakan prefix tabel untuk meningkatkan unjuk kerja (performance) karena membantu server basisdata untuk menemukan secara persis kolom yang dimaksud.

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

Panduan penulisan :

- Nama alias tabel ditulis pada klausa FROM. Nama tabel dituliskan secara lengkap diikuti spasi dan kemudian nama alias tabel.
- Nama alias dapat terdiri 30 character, namun nama alias yang lebih pendek lebih baik dibandingkan nama yang panjang.
- Nama alias tabel sebaiknya memiliki makna/arti yang dekat dengan tabel aslinya.
- Nama alias tabel hanya valid untuk satu statemen SELECT, untuk statemen SELECT yang lain harus didefinisikan kembali.

Contoh :

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM employees e, departments d
WHERE 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
11	120	Weiss	50	50	1500
12	121	Fripp	50	50	1500

```
SELECT e.last_name, e.department_id,
       d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND e.last_name = 'Matos';
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

6. Menggabungkan Lebih dari Dua Tabel

Kadang-kadang diperlukan untuk join lebih dari 2 tabel. Sebagai contoh, untuk menampilkan last_name, department_name, dan city dari setiap pegawai, maka harus menggabungkan tabel EMPLOYEES, DEPARTMENTS, dan LOCATIONS.

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Austin	60
Pataballa	60
Lorentz	60
Greenberg	100
Faviet	100
Chen	100
Sciarra	100
Urman	100
Popp	100
Raphaely	30
Khoo	30

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
30	1700
40	2400
50	1500
60	1400
70	2700
80	2500
90	1700
100	1700
110	1700
120	1700
130	1700
140	1700
150	1700
160	1700

LOCATIONS

LOCATION_ID	CITY
1000	Roma
1100	Venice
1200	Tokyo
1300	Hiroshima
1400	Southlake
1500	South San Francisco
1600	South Brunswick
1700	Seattle
1800	Toronto
1900	Whitehorse
2000	Beijing
2100	Bombay
2200	Sydney
2300	Singapore
2400	London
2500	Oxford

Untuk melalukan join n tabel, diperlukan minimum n-1 kondisi join. Misalnya, untuk join 3 buah tabel, dibutuhkan minimal 2 join. Gunakan AND untuk menggabungkan lebih dari 1 kondisi join.
Contoh :

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

Hasil :

	LAST_NAME	DEPARTMENT_NAME	CITY
1	Abel	Sales	Oxford
2	Ande	Sales	Oxford
3	Atkinson	Shipping	South San Francisco
4	Austin	IT	Southlake
5	Baer	Public Relations	Munich
6	Baida	Purchasing	Seattle
7	Banda	Sales	Oxford
8	Bates	Sales	Oxford
9	Bell	Shipping	South San Francisco
10	Bernstein	Sales	Oxford
11	Bissot	Shipping	South San Francisco
12	Bloom	Sales	Oxford
13	Bull	Shipping	South San Francisco
14	Cabrio	Shipping	South San Francisco
15	Cambrault	Sales	Oxford
16	Cambrault	Sales	Oxford

7. Outer Join

Pada *equijoin (inner join)*, jika suatu baris (row) tidak memenuhi kondisi *join* maka baris tersebut tidak akan ditampilkan di hasil query. Pada contoh *equijoin (inner join)* di bawah ini, meskipun tabel EMPLOYEES berisi 107 employee, tetapi hasil query hanya menampilkan 106 karena disebabkan employee bernama Grant tidak mempunyai department_id di tabel EMPLOYEES.

Sintak *inner join* :

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

atau dapat pula ditulis dengan cara :

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e INNER JOIN departments d
ON e.department_id = d.department_id;
```

Amatilah hasil querynya !

Baris yang tidak ditampilkan tersebut, dapat ditampilkan menggunakan *outer join*. Terdapat 3 macam *outer join* yaitu :

a. *Left outer join* : menampilkan semua baris tabel sebelah kiri termasuk yang tidak mempunyai baris yang cocok dengan tabel sebelah kanan.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

m

Results:

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Colmenares	30	Purchasing
5	Himuro	30	Purchasing
6	Tobias	30	Purchasing

...

100	Urman	100	Finance
101	Sciarra	100	Finance
102	Chen	100	Finance
103	Faviet	100	Finance
104	Greenberg	100	Finance
105	Gietz	110	Accounting
106	Higgins	110	Accounting
107	Grant	(null)	(null)

Tampak bahwa semua baris pada tabel EMPLOYEES (tabel sebelah kiri) ditampilkan semua, termasuk employee Grant yang tidak memiliki department_id.

b. *Right outer join* : menampilkan semua baris tabel di sebelah kanan termasuk yang tidak mempunyai baris yang cocok dengan tabel sebelah kiri.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Tobias	30	Purchasing
5	Colmenares	30	Purchasing
6	Baida	30	Purchasing
7	Raphaely	30	Purchasing

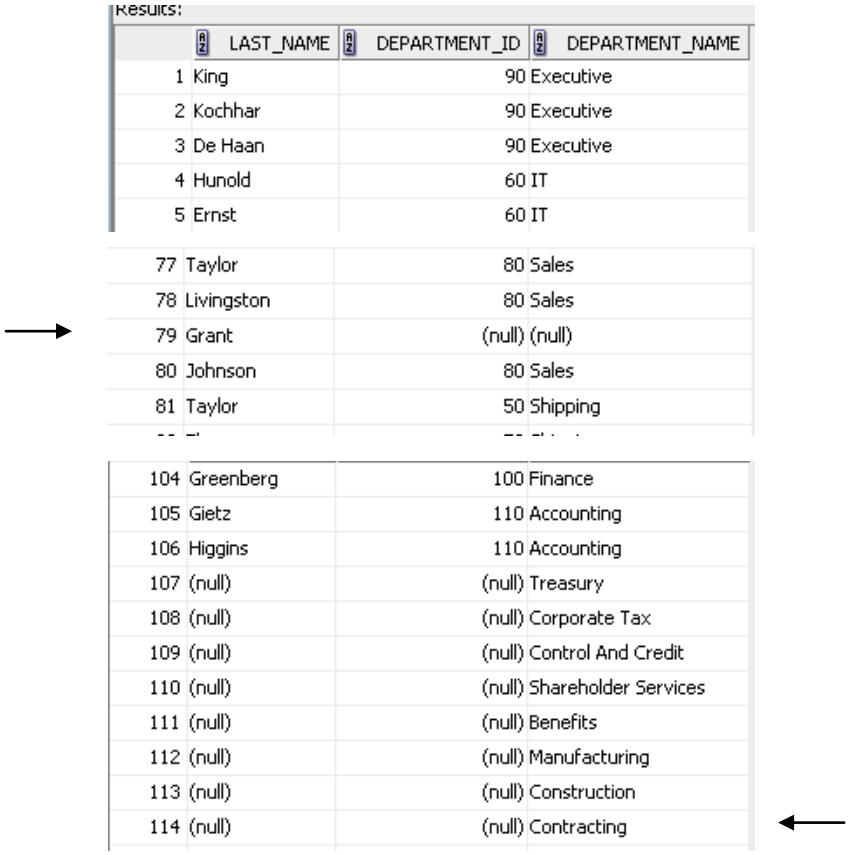
...

104	Greenberg	100	Finance
105	Gietz	110	Accounting
106	Higgins	110	Accounting
107	(null)	(null)	Treasury
108	(null)	(null)	Corporate Tax
109	(null)	(null)	Control And Credit
110	(null)	(null)	Shareholder Services
111	(null)	(null)	Benefits
112	(null)	(null)	Manufacturing
113	(null)	(null)	Construction
114	(null)	(null)	Contracting

Tampak bahwa semua baris pada tabel DEPARTMENTS (tabel sebelah kanan) ditampilkan semua, termasuk department Contracting yang tidak memiliki pegawai.

c. *Full outer join* : menampilkan semua baris tabel di sebelah kanan termasuk yang tidak mempunyai baris yang cocok dengan tabel sebelah kiri. Demikian pula sebaliknya ditampilkan semua baris tabel di sebelah kiri termasuk yang tidak mempunyai baris yang cocok dengan tabel sebelah kanan.

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```



Results:

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	King	90	Executive
2	Kochhar	90	Executive
3	De Haan	90	Executive
4	Hunold	60	IT
5	Ernst	60	IT
77	Taylor	80	Sales
78	Livingston	80	Sales
79	Grant	(null)	(null)
80	Johnson	80	Sales
81	Taylor	50	Shipping
104	Greenberg	100	Finance
105	Gietz	110	Accounting
106	Higgins	110	Accounting
107	(null)	(null)	Treasury
108	(null)	(null)	Corporate Tax
109	(null)	(null)	Control And Credit
110	(null)	(null)	Shareholder Services
111	(null)	(null)	Benefits
112	(null)	(null)	Manufacturing
113	(null)	(null)	Construction
114	(null)	(null)	Contracting

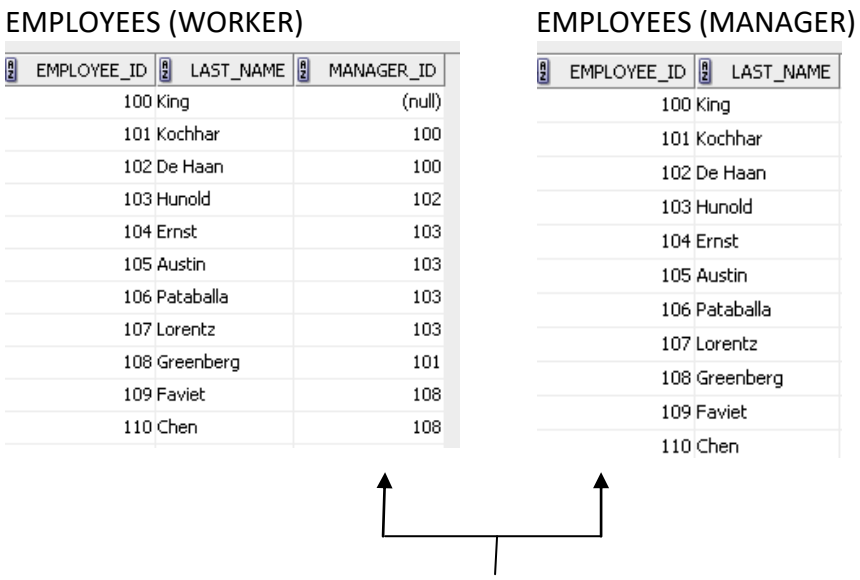
Tampak bahwa semua baris pada tabel EMPLOYEES (tabel sebelah kiri) ditampilkan semua, termasuk employee Grant yang tidak memiliki department_id. Demikian pula sebaliknya semua baris pada tabel DEPARTMENTS (tabel sebelah kanan) ditampilkan semua, termasuk department Contracting yang tidak memiliki pegawai.

Catatan : Perbandingan INNER vs OUTER join

- Join 2 tabel hanya menampilkan baris yang cocok (berelasi), hal ini disebut suatu *inner join (equijoin).*
- Join antara 2 tabel yang selain menampilkan hasil inner join juga menampilkan baris yang tidak cocok di sebelah kiri (atau kanan) disebut left outer join atau right outer join.
- Join antara 2 tabel yang selain menampilkan hasil inner join juga menampilkan baris yang tidak cocok di sebelah kiri dan kanan disebut full outer join.

8. Self Join

Kadang-kadang diperlukan untuk melakukan join tabel dengan dirinya sendiri. Sebagai contoh, untuk menampilkan nama manager dari setiap pegawai, diperlukan join tabel EMPLOYEES dengan dirinya sendiri. Join seperti ini disebut *self join*.



Gambar 6.3. Self Join

Untuk mencari manager dari seorang pegawai misalnya Lorentz, maka diperlukan langkah-langkah sebagai berikut :

- Temukan Lonrentz pada tabel EMPLOYEES dengan melihat kolom LAST_NAME.
- Temukan manager dari Lorentz dengan melihat pada kolom MANAGER_ID. Nomor manager_id Lorentz adalah 103.
- Temukan nama manager yang memiliki EMPLOYEE_ID 103. Tampak bahwa emplyee_id tersebut milik Hunold. Jadi manager Lorentz adalah Hunold.

Dalam proses ini, dilakukan 2 kali melihat table EMPLOYEES. Pertama, melihat table untuk memperoleh Lorentz pada kolom LAST_NAME dan memperoleh nilai 103 pada MANAGER_ID. Kedua, melihat EMPLOYEE_ID dengan nilai 103 dan memperloleh LAST_NAME Hunold.

```
SELECT worker.last_name || ' memiliki manager '
        || manager.last_name
FROM employees worker, employees manager
WHERE worker.manager_id = manager.employee_id;
```

Results:

	WORKER.LAST_NAME||'MEMILIKIMANAGER'||MANAGER.LAST_NAME
1	Smith memiliki manager Cambrault
2	Ozer memiliki manager Cambrault
3	Kumar memiliki manager Cambrault
4	Fox memiliki manager Cambrault
5	Bloom memiliki manager Cambrault
6	Bates memiliki manager Cambrault
7	Hunold memiliki manager De Haan
8	Vishney memiliki manager Errazuriz
9	Marvins memiliki manager Errazuriz
10	Lee memiliki manager Errazuriz
11	Greene memiliki manager Errazuriz
12	Banda memiliki manager Errazuriz
13	Ande memiliki manager Errazuriz
14	Sarchand memiliki manager Fripp
15	Olson memiliki manager Fripp
16	Marlow memiliki manager Fripp
17	Dellinger memiliki manager Fripp

Untuk melakukan self join, gunakan dua buah nama alias tabel untuk tabel yang sama. Contoh : tabel EMPLOYEES memiliki nama alias sebagai WORKER dan juga sebagai MANAGER. Join dilakukan pada klausa WHERE dimana manager_id setiap karyawan berelasi dengan employee_id setiap manager.

C. TUGAS

1. Buatlah query 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 pegawai yang bekerja di departemen Shipping.

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

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#.

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.

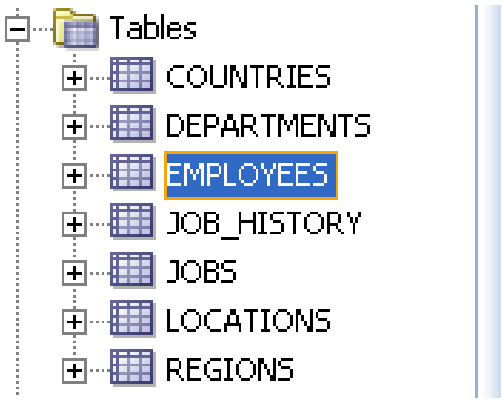
D. DAFTAR PUSTAKA

1. 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

Terdapat 7 buah tabel yang digunakan, antara lain :



LAMPIRAN
STRUKTUR TABEL DAN RELASI TABEL