MODUL PRAKTIKUM
BASIS DATA

Oleh :

Amelia Yusnita, S.Kom.,M.Kom

PROGRAM STUDI SISTEM INFORMASI

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

WIDYA CIPTA DHARMA

SAMARINDA

	STMIK WIDYA CIPTA DHARMA SAMARINDA	**S1- SISTEM INFORMASI**
Prak. BASIS DATA	**Penggabungan Data**	**LABSHEET 05**
Semester III		**Dosen :** **Amelia Yusnita,S.Kom.,M.Kom** **Email : amelia@wicida.ac.id**

1. TUJUAN

Mahasiswa mampu dan memahami cara melakukan query yang melibatkan lebih dari satu tabel

2. TEORI

Pada saat bekerja dengan database, anda akan banyak menjumpai proses seleksi data yang melibatkan lebih dari satu tabel. Syarat utama untuk melakukan seleksi data dari dua tabel atau lebih adalah adanya kolom relasi. Kolom relasi adalah kolom yang digunakan sebagai kunci untuk menghubungkan antar tabel yang satu dengan yang lain.

3. LANGKAH KERJA

A. Join dan Equijoin

Makna Join adalah penggabungan data yang berasal dari tabel. Umumnya operator yang digunakan untuk melaksanakan penggabungan berupa operator sama dengan (=). Penggabungan yang menggunakan operator inilah yang dikenal dengan sebutan *equality join* atau *equijoin*. Hasil penggabungannya berupa baris-baris gabungan tabel-tabel yang memiliki nilai sama untuk kolom-kolom yang disebutkan dalam kondisi penggabungan namun, dalam prakteknya kdangkala diperlukan penggabungan yang berbeda dengan yang telah diuraikan

Apabila operator yang digunakan untuk menghubungkan satu tabel dengan tabel lainnya berupa sama dengan (=) melainkan berupa operator seperti <> atau >= maka *join* akan disebut sebagai *non-equijoin*.

Untuk mempraktekkan berbagai operasi join, siapkanlah terlebih dahulu database bernawa "**Wilayah**" dan tabel bernama **Provinsi, Kota, dan Kecamatan**. Pernyataan yang diperlukan untuk menciptakan database adalah seperti berikut.

```
1 CREATE DATABASE wilayah;
```

Setelah database tercipta, buatlah tabel dengan pernyataan berikut :

```sql
CREATE TABLE provinsi (
kode_prov varchar (2) NOT NULL PRIMARY KEY,
nama_prov varchar (30) NOT NULL );
```

```sql
CREATE TABLE kota (
kode_prov varchar (2) NOT NULL,
kode_kota varchar (2) NOT NULL,
nama_kota varchar (25) NOT NULL,
PRIMARY KEY (kode_prov, kode_kota));
```

Setelah itu, masukkan data pada masing-masing tabel seperti dibawah ini

```sql
INSERT INTO Provinsi VALUES
('12','Sumatera Utara'),('20','DKI Jakarta'),
('21','Jawa Barat'),('22','Jawa Tengah'),
('23','DI Yogyakarta'),('24','Jawa Timur'),
('32','Kalimantan Utara');
```

Hasilnya :

kode_prov	nama_prov
12	Sumatera Utara
20	DKI Jakarta
21	Jawa Barat
22	Jawa Tengah
23	DI Yogyakarta
24	Jawa Timur
32	Kalimantan Utara

```sql
INSERT INTO kota VALUES
('12','01','Medan'),('21','01','Bandung'),
('21','02','Bogor'),('22','01','Semarang'),
('22','02','Kudus'),('23','01','Yogya'),
('23','02','Sleman'),('24','01','Surabaya'),
('31','01','Banjarmasin');
```

Hasilnya :

kode_prov	kode_kota	nama_kode
12	01	Medan
21	01	Bandung
21	02	Bogor
22	01	Semarang
22	02	Kudus
23	01	Yogya
23	02	Sleman
24	01	Surabaya
31	01	Banjarmasin

```sql
SELECT kota.nama_kota, provinsi.nama_prov
FROM kota, provinsi
WHERE kota.kode_prov = provinsi.kode_prov;
```

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur

Perhatikan bahwa kota banjarmasin yang tidak punya pasangan nilai kode_prov pada tabel provinsi tidak ikut ditampilkan. Provinsi kalimantan utara juga tidak muncul mengingat tidak ada kota di kalimantar utara yang disebutkan pada tabel kota. Dengan

perkataan lain, hanya yang memenuhi syarat kota.kode_prov = provinsi.kode_prov yang ditampilkan.

E. Natural Join

Natural Join adalah penggabungan data dari dua tabel yang didasarkan pada kolom dengan nama sama pada kedua tabel. Penggabungan ini mencerminkan hubungan antara kunci tamu dan kunci primer dalam dua tabel. Selain menggunakan cara seperti

```
SELECT kota.nama_kota, provinsi.nama_prov
FROM kota, provinsi
WHERE kota.kode_prov = provinsi.kode_prov;
```

MySQL memberikan cara alternatif dengan menggunakan kata kunci *NATURAL JOIN* pernyataan yang digunakan :

```
SELECT nama_kota,nama_prov
FROM kota NATURAL JOIN provinsi;
```

Pernyataan ini juga bisa ditulis menjadi :

```
SELECT nama_kota, nama_prov
FROM provinsi NATURAL JOIN kota;
```

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur

Seperti halnya pernyataan SELECT bisa juga ditambahkan klausa seperti ORDER BY untuk mengurutkan hasil. Contoh :

```
SELECT nama_kota, nama_prov
FROM provinsi NATURAL JOIN kota
ORDER BY nama_kota;
```

Hasilnya :

nama_kota ▲ 1	nama_prov
Bandung	Jawa Barat
Bogor	Jawa Barat
Kudus	Jawa Tengah
Medan	Sumatera Utara
Semarang	Jawa Tengah
Sleman	DI Yogyakarta
Surabaya	Jawa Timur
Yogya	DI Yogyakarta

F. Inner Join dan Outer Join

Equijoin sering kali dibedakan menjadi dua kategori yakni *inner equijoin* (atau disingkat *inner join*) dan *outer equijoin* (atau disingkat *outer join*). Untuk melihat perbedaannya kedua *Equijoin* ini, dapat dilihat berikut ini:

```
SELECT kota.nama_kota, provinsi.nama_prov
FROM kota, provinsi
WHERE kota.kode_prov = provinsi.kode_prov;
```

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur

Jika diperhatikan terdapat kota yang tidak terlihat pada hasil diatas yakni (Banjarmasin). Hasil seperti itulah, yakni hanya berupa baris yang memiliki data pada kedua tabel, yang disebut dengan *inner join*.

Kota yang tidak ditampilkan adalah Banjarmasin yang tidak memiliki entri pada tabel provinsi. Namun bagaimana seandainya dikehendaki agar kota yang tidak tercantum pada tabel provinsi ikut ditampilkan pada hasil query ?

Hasil yang diharapkan tersebut dinamakan dengan *outer join*. Untuk melakukannya bisa menggunakan NATURAL LEFT JOIN. Contoh :

```
SELECT nama_kota,nama_prov
FROM kota NATURAL LEFT JOIN provinsi;
```

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur
Banjarmasin	*NULL*

Hasil query terlihat bahwa Banjarmasin dimunculkan tetapi nama_prov diisi dengan NULL.

Alternatif lain yaitu dengan menggunakan LEFT JOIN. Dalam hal ini, kondisi dengan menggunakan ON atau USING seperti pada CROSS JOIN diperlukan. Contoh :

```
SELECT nama_kota, nama_prov
FROM kota LEFT JOIN provinsi
ON kota.kode_prov = provinsi.kode_prov;
```

Jika menggunakan USING, pernyataannya diperlukan berupa :

```
SELECT nama_kota, nama_prov
FROM kota LEFT JOIN provinsi
USING (kode_prov);
```

Istilah *outer join* pada contoh yang ada, inilah yang sering dinamakan LEFT JOIN atau LEFT OUTER JOIN. Pada LEFT JOIN, semua baris pada tabel sebelah kiri ditampilkan dan jika tidak ada pasangan pada tabel kanan, nilai NULL digunakan.

Gunakan kata LEFT menjadi RIGHT, contoh :

```
SELECT nama_kota, nama_prov
FROM kota RIGHT JOIN provinsi
USING (kode_prov);
```

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
NULL	DKI Jakarta
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur
NULL	Kalimantan Utara

Pada gambar diatas semua provinsi akan ikut ditampilkan sekalipun belum memiliki data kota. Misalnya Provinsi DKI Jakarta dan Kalimantan utara. Pernyataan tersebut juga dapat diganti dengan NATURAL RIGHT JOIN seperti berikut :

```
SELECT nama_kota, nama_prov
FROM kota NATURAL RIGHT JOIN provinsi
```

Istilah *outer join* pada contoh yang menggunakan RIGHT JOIN, inilah yang sering dinamakan RIGHT JOIN atau RIGHT OUTER JOIN. Pada RIGHT JOIN, semua baris pada tabel sebelah kanan ditampilkan dan jika tidak ada pasangan pada tabel kiri, nilai NULL digunakan.

G. Penggunaan Alias pada Join

Pada penyataan SELECT yang menggunakan JOIN, penerapan alias pada nama tabel tetap bisa dilakukan. Seperti :

```
SELECT nama_kota, nama_prov
FROM kota CROSS JOIN provinsi
ON kota.kode_prov = provinsi.kode_prov;
```

Dapat ditulis menjadi :

```sql
SELECT k.nama_kota, p.nama_prov
FROM kota k CROSS JOIN provinsi p
ON k.kode_prov = p.kode_prov;
```

H. Operator UNION

Operator UNION berguna untuk menggabungkan hasil dari dua query tanpa ada baris yang kembar, sebagai contoh

```sql
SELECT nama_kota, nama_prov
FROM kota NATURAL LEFT JOIN provinsi
UNION
SELECT nama_kota, nama_prov
FROM kota NATURAL RIGHT JOIN provinsi;
```

Menggabungkan hasil dari dua SELECT

Hasilnya :

nama_kota	nama_prov
Medan	Sumatera Utara
Bandung	Jawa Barat
Bogor	Jawa Barat
Semarang	Jawa Tengah
Kudus	Jawa Tengah
Yogya	DI Yogyakarta
Sleman	DI Yogyakarta
Surabaya	Jawa Timur
Banjarmasin	*NULL*
NULL	DKI Jakarta
NULL	Kalimantan Utara

Dengan cara seperti itulah dimungkinkan untuk memperoleh semua kota yang tidak punya pasangan nama provinsi, semua provinsi yang belum mempunyai pasangan dalam tabel kota, dan sekaligus semua kota yang memiliki pasangan nama provinsi dalam tabel provinsi.

4. LATIHAN