Pertemuan 6
QUERY BANYAK TABEL

Dalam basis data selain fungsi, melakukan query dengan data yang diperoleh dari beberapa tabel merupakan salah satu dasar konsep SQL yang wajib dipahami.

Bukalah skema pada pertemuan 2 dan perhatikan struktur tabel MAHASISWA dan JURUSAN. Pada tabel mahasiswa terdapat sebuah field dengan nama **kode_jur**. Kalau diperhatikan, pada tabel JURUSAN juga terdapat field **kode_jur**. Perhatikan juga bahwa keduanya memiliki tipe data dan ukuran data yang sama.

Agar data dari kedua tabel tersebut dapat ditampilkan dengan benar di daam satu query, maka kedua tabel tersebut harus dihubungkan.

Istilah yang digunakan untuk menghubungkan dua atau lebih tabel yang memiliki keterkaitan primary key dan foreign key disebut sebagai JOIN.

Berdasarkan teori, ada beberapa jenis JOIN:

- **CROSS JOIN/Cartesian Product**

  Apabila sebuah query yang menggunakan lebih dari satu tabel namun tidak menyebutkan atau kurang syarat/kondisi join-nya, maka pada query tersebut terjadi suatu kondisi yang disebut cartesian product/cross join, dimana hasil query akan muncul (tanpa error) namun hasil yang diberikan adalah salah.

- **INNER JOIN**

  Merupakan join yang digunakan untuk mencari nilai yang terdapat di dalam tabel. Inner join terbagi atas beberapa macam, yaitu:

  - **EQUI JOIN**

    Equi join menggunakan sintaks berikut:

    ```
    SELECT tabel1.kolom, tabel2.kolom
    FROM tabel1, tabel2
    WHERE tabel1.kolom1 = tabel2.kolom2;
    ```

  - **EQUI JOIN dengan Klausa INNER JOIN**

    Sintaks yang digunakan adalah sebagai berikut:

    ```
    SELECT tabel1.kolom, tabel2.kolom
    FROM tabel1, tabel2
    WHERE tabel1.kolom1 = tabel2.kolom2;
    ```

  - **EQUI JOIN dengan Klausa JOIN**

    Sintaks yang digunakan adalah sebagai berikut:

    ```
    SELECT tabel1.kolom, tabel2.kolom
    FROM tabel1 JOIN tabel2 ON tabel1.kolom1 = tabel2.kolom2;
    ```

  - **NATURAL JOIN**

    Adalah sebuah join yang sama seperti *equi-join* namun tidak menyebutkan kondisi join-nya karena telah terdefinisikan pada tabel, yaitu ketika tabel di-*create references foreign key* telah disebutkan.

    Ada syarat yang harus dipenuhi untuk natural join, yaitu: baik nama field, tipe data, maupun ukuran tipe data pada kedua tabel harus sama.

    Sintaks yang digunakan dalam natural join adalah:

```
SELECT tabel1.kolomA, …, tabel2.kolomZ
FROM tabel1 NATURAL JOIN tabel2
```

- **OUTER JOIN**

  Merupakan jenis join yang selain mencari sebuah data yang terdapat di dalam tabel, namun juga untuk mencari nilai yang “orphan” (data yang tidak memiliki pasangan data dengan tabel lain).

  Bentuk sintaks dari outer join mengikuti bentuk Equi-Join dengan klausa JOIN, dengan perubahan minor seperti yang tampak berikut ini.

```
SELECT tabel1.kolom, tabel2.kolom
FROM tabel1 LEFT JOIN tabel2 ON tabel1.kolom1 = tabel2.kolom2;
```

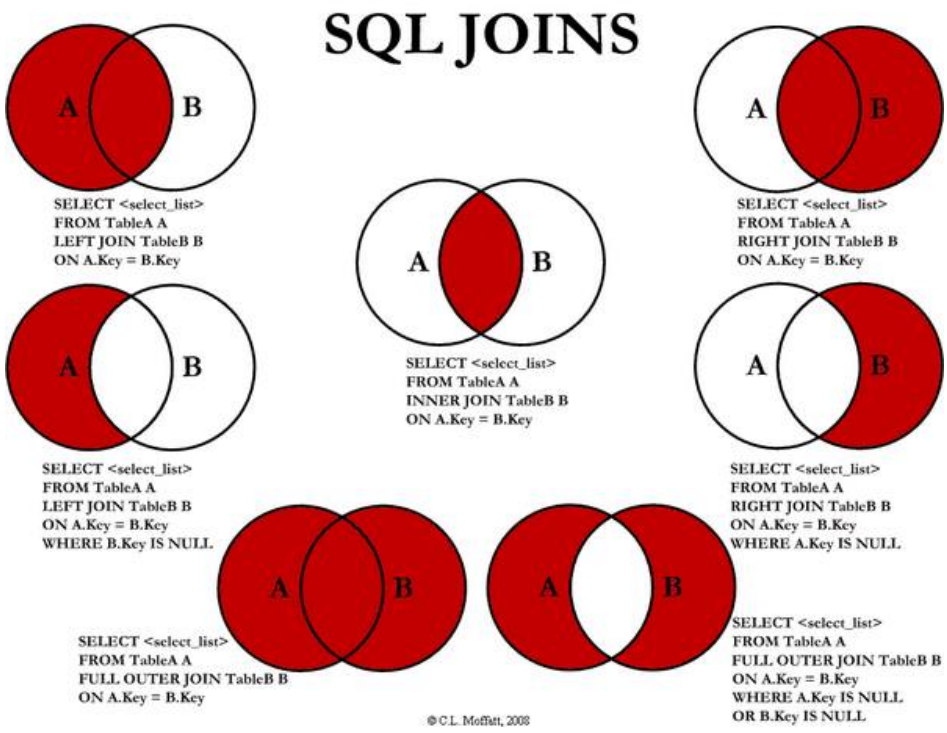
  Bentuk dari OUTER JOIN pada MySQL ada dua macam yaitu, LEFT OUTER JOIN atau RIGHT OUTER JOIN. Adapun perbedaan dari keduanya adalah hanya pada posisi tabel yang akan dimunculkan informasinya (tabel induk) di dalam query, apakah terletak di sebelah kiri atau sebelah kanan dari tabel anaknya.

- **SELF JOIN**

  Merupakan jenis JOIN yang memanggil tabel yang sama lebih dari sekali. Adapun bentuk ini mendukung model database yang membutuhkan struktur hirarki atau struktur organisasi. Penyebutan tabel yang sama dua kali dalam query tidak diperbolehkan, terkecuali dengan menggunakan alias (Lihat Alias Untuk Tabel di bawah.)

Perlu diperhatikan bahwa baik Equi-Join, Inner Join, Join, maupun natural join adalah sama-sama termasuk dalam inner join.

Jika diperhatikan secara teori, proses join sesungguhnya telah dipelajari pada tingkat pendidikan dasar. Perhatikan gambar berikut.



Aturan penting JOIN:
Kebanyakan pemula bingung apabila harus melakukan join yang lebih dari 2 tabel. Para pemula biasanya tidak dapat menentukan banyaknya syarat/aturan join yang wajib dituliskan agar query menghasilkan nilai yang benar.
Ada rumus yang mudah diingat yaitu:

Banyak JOIN = n - 1

dimana n adalah banyak tabel.
Jadi, jika sebuah query membutuhkan 10 tabel untuk digabungkan, maka diperlukan 10-1=9 syarat join agar hasil query tersebut benar.

Alias Untuk Tabel
Untuk query yang membutuhkan penyebutan nama tabel berulang-ulang, penggunaan alias sebagai pengganti sementara nama tabel dapat mempercepat penulisan perintah query dan dapat mengurangi kemungkinan terjadinya kesalahan.

Contoh:

```
SELECT nama_mahasiswa, alamat, no_HP, rs.kode_matakuliah, jadwal_matakuliah
FROM mahasiswa m, rencana_studi rs
WHERE m.kode_matakuliah = rs.kode_matakuliah
```

Perhatikan bagian yang dicetak tebal adalah bentuk alias yang digunakan pada tabel.

PRAKTIK

1. Tambahkan data-data berikut ini pada tabel MAHASISWA.

NIM	Nama_mhs	Tempat_lahir	Tgl_lahir	Jenis_kelamin	Kode_jur	Alamat	Kota	Asal_sekolah	Nomor_tlp	Tgl_daftar
155410001	Paidi	Klaten	1997-03-03	P	TI	Jl Bantul	Yogya	SMK 1 Yogya	089995558888	2015-04-04
155410002	Sutarmin	Solo	1996-07-01	P	TI	Jl Garuda	SMK 3 Yogya	Yogya	085656565656	2015-04-12
155410003	Djono	Ngawi	1997-12-05	P	TI	Jl Panembahan Senopati	Yogya	SMA 3	088989898989	2015-07-28

2. Tambahkan data-data berikut pada tabel KRS.

```
INSERT INTO krs VALUES('143110001','TK02','1020300004',2016,3,'B');
INSERT INTO krs VALUES('143210001','KA01','1020300007',2016,3,'A');
INSERT INTO krs VALUES('155410001','TI01','1020300006',2016,1,'C');
INSERT INTO krs VALUES('155410002','TI01','1020300006',2016,1,'D');
INSERT INTO krs VALUES('155410003','TI01','1020300006',2016,1,'E');
INSERT INTO krs VALUES('155410001','TI02','1020300003',2016,1,'B');
INSERT INTO krs VALUES('155410002','TI02','1020300003',2016,1,'C');
INSERT INTO krs VALUES('155410003','TI02','1020300003',2016,1,'E');
```

3. Tambahkan data-data berikut pada tabel TRANSKRIP

nim	kode_mkuliah	nilai
145410001	TI01	A
145410001	TI02	B
145410002	TI02	B
145610001	SI01	A
145610001	SI02	A
143310001	MI02	B
143110001	TK02	B
143210002	KA01	C
143210001	KA01	A
155410001	TI01	C
155410002	TI01	D
155410003	TI01	E
155410001	TI02	B
155410002	TI02	C
155410003	TI02	E

4. Buatlah query untuk menampilkan nama mahasiswa beserta nama jurusannya masing-masing.

```
SELECT nama_mhs, nama_jurusan
FROM mahasiswa m, jurusan j
WHERE m.kode_jur=j.kode_jur;
```

5. Buatlah query untuk menampilkan nama dosen dan kode mata kuliah yang pernah diampu olehnya.

```
SELECT nama_dosen, kode_mkul
FROM dosen d, krs
WHERE d.NID=krs.NID;
```

6. Buatlah query untuk menampilkan nama mahasiswa beserta kode mata kuliah apa saja yang pernah dia KRS-kan.

```
SELECT nama_mhs, kode_mkul
FROM mahasiswa m, krs
WHERE m.NIM=krs.NIM
```

7. Buatlah query untuk menampilkan nama mahasiswa beserta nama mata kuliah yang pernah diambilnya beserta nilai yang dia peroleh.

```
SELECT nama_mhs, nama_matakuliah, nilai
FROM mahasiswa m, matakuliah mk, nilai n
WHERE m.NIM=krs.NIM AND
      krs.kode_mkul=mk.kode_mkul
```

8. Buatlah query untuk menampilkan nama mata kuliah beserta nama mahasiswa yang pernah mengambil mata kuliah tersebut.

```
SELECT nama_matakuliah, nama_mhs
FROM mahasiswa m, matakuliah mk, krs
WHERE m.NIM=krs.NIM AND
      krs.kode_mkul=mk.kode_mkul;
```

9. Modifikasi query tersebut untuk menampilkan hanya nama mata kuliah beserta banyaknya mahasiswa yang mengambil masing-masing mata kuliah.

```
SELECT nama_matakuliah, COUNT(krs.nim)
FROM mahasiswa m, matakuliah mk, krs
WHERE m.NIM=krs.NIM AND
      krs.kode_mkul=mk.kode_mkul
GROUP BY nama_matakuliah;
```

10. Buatlah query untuk menampilkan nama mahasiswa yang mengambil mata kuliah beserta nama dosen yang mengampunya (hanya tampilkan nama mahasiswa dan nama dosennya).
11. Buatlah query untuk menampilkan nama mahasiswa yang masih mendapatkan nilai D dan E di transkrip-nya. Tampilkan pula nama mata kuliah beserta nilai-nya di transkrip tersebut.
12. Buatlah query untuk menampilkan nama jurusan dan banyaknya mahasiswa di tiap-tiap jurusan tersebut.
13. Buatlah query untuk menampilkan nama-nama dosen beserta banyaknya mata kuliah yang TELAH diampu.
14. Buatlah query untuk menampilkan nama mata kuliah, nilai mata kuliah. Urutkan hasilnya berdasarkan nama mata kuliah.

```
SELECT Nama_matakuliah, nilai, COUNT(nilai)
FROM matakuliah mk JOIN transkrip t ON
                t.kode_mkuliah=mk.kode_mkul
```

```
GROUP BY nama_matakuliah, t.nilai
ORDER BY nama_matakuliah;
```