MODUL 5

RELASI & JOIN

A. TUJUAN

- Memahami keterhubungan entitas di dalam basis data.
- Memahami jenis-jenis operasi pengambilan data di beberapa entitas.
- Mampu menyelesaikan kasus retrieval yang melibatkan lebih dari satu entitas.
- Memahami fungsi Union pada mysql.

B. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas

C. DASAR TEORI

1. Relationship

Relationship adalah suatu hubungan antara beberapa entitas. Konsep ini sangat penting sekali di dalam basis data, di mana memungkinkan entitas-entitas untuk saling berhubungan satu sama lain.

Didalam sebuah relationship, primary key memiliki peran penting untuk mengaitkan entitas. Selain itu, primary key juga digunakan untuk mendefinisikan batasan keterhubungan.

2. Join

Join merupakan salah satu kontruksi dasar dari SQL dan basis data. *Join* dapat didefinisikan sebagai kombinasi record dari dua atau lebih table di dalam basis data relasional dan menghasilkan sebuah tabel (*temporary*) baru yang disebut sebagai *joined tabel*.

Join dapat diklasifikasikan ke dalam dua jenis, yaitu ***inner join*** dan ***outer join***. **a. Inner Join**

Inner join pada dasarnya adalah menemukan persimpangan (*intersection*) antara dua buah tabel.

Sintaks inner join diperlihatkan sebagai berikut :

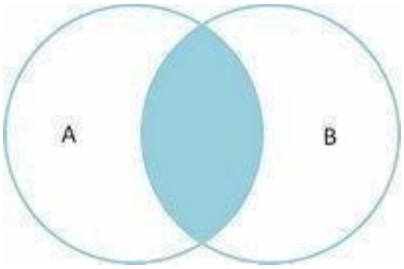
```
SELECT A1, A2, . . ., An

FROM r1
INNER JOIN r2
ON r1.join_key = r2.join_key
```

Inner join juga dapat direpresentasikan dalam bentuk implisit sebagai berikut :

```
SELECT A1, A2, . . ., An
FROM r1, r2
WHERE r1.join_key = r2.join_key
```

Misalkan terdapat table A dan B, maka hasil inner join dapat diperlihatkan sebagai bidang terasir dalam diagram Venn seperti Gambar 1.



Gambar 1. Inner Join

b. Outer Join

Outer join dibagi ke dalam tiga jenis, yaitu *left outer join*, *right outer join* dan *full outer join.*

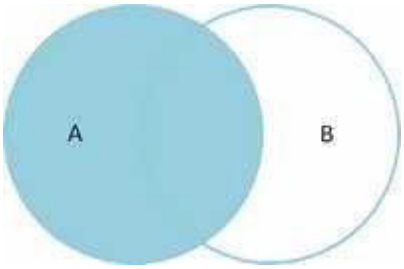
Left Outer Join

Left outer join atau *left join* mengembalikan semua nilai dari table kiri ditambah dengan nilai dari tabel kanan yang sesuai atau **NULL** jika tidak ada nilai yang sesuai.

Sintaks *left outer join* diperlihatkan sebagai berikut :

```
SELECT A1, A2, . . ., An
FROM r1
LEFT OUTER JOIN r2
ON r1.join_key = r2.join_key
```

Left outer join antar tabel A dan B dapat diilustrasikan dalam diagram Venn seperti Gambar 2.



Gambar 2. Left Outer Join

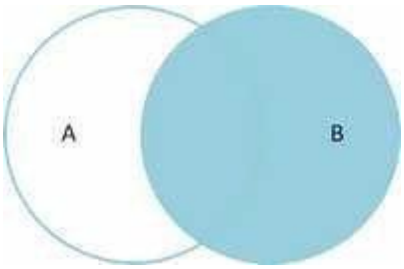
Right Outer Join

Right outer join atau *right join* pada dasarnya sama seperti left join, namun dalam bentuk terbalik, kanan dan kiri.

Sintaks *right outer join* diperlihatkan sebagai berikut :

```
SELECT A1, A2, . . ., An
FROM r1
RIGHT OUTERR JOIN r2  ON r1.join_key
= r2.join_key
```

Right outer join antara table A dan B dapat diilustrasikan dalam diagram Venn seperti Gambar 3.



Gambar 3. Right Outer Join

Full Outer Join

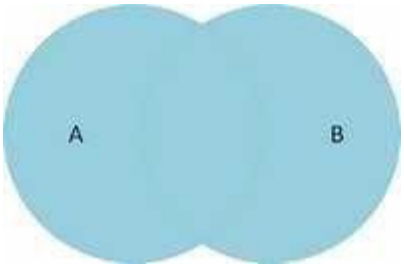
Full outer join atau *full join* pada hakikatnya merupakan kombinasi dari left dan right join.

Sintaks *full outer join* diperlihatkan sebagai berikut :

```
SELECT A1, A2, . . ., An

FROM r1
FULL OUTER JOIN r2
ON r1.join_key = r2.join_key
```

Bentuk visual dari full outer join dapat diperlihatkan menggunakan diagram Venn seperti Gambar 4.



Gambar 4. Full Outer Join

Selain empat jenis join yang utama di atas, masih ada beberapa variasi join lainnya, seperti **CROSS JOIN** (*cartesian product*), **NATURAL JOIN** dan sebagainya.

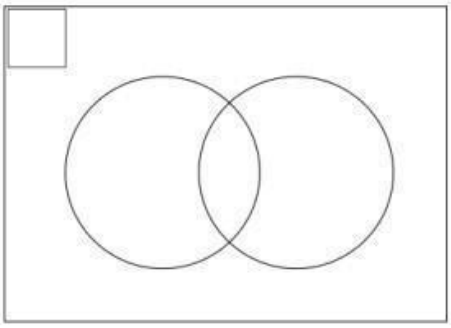
Perlu juga diperhatikan, join bisa diimplementasikan dalam bentuk bersarang (*nested join*). Jadi, di dalam sebuah operasi join bisa terdapat operasi join lainnya.

3. Union

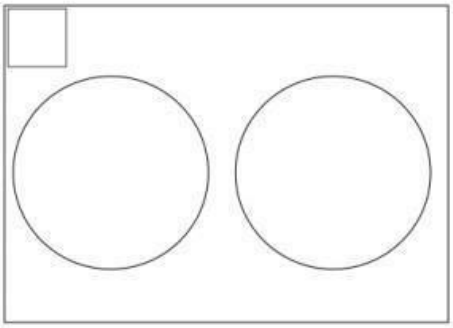
MySQL Union adalah statemaen yang mengkombinasikan dua buah atau lebih resulset dari beberapa table dengan statemen SELECT sehingga menjadi satu buah resulset. Union Statemen memiliki beberapa ketentuan sebagai berikut :

a. Jumlah kolom/field dari setiap statemen SELECT harus sama.
b. Tipe data kolom/field dari setiap statemen SELECT harus kompatibel.

Secara default statemen UNION akan menghapus semua record duplikat dari resulset. Apabila Anda ingin record duplikat tetap di tampilkan maka pada resulset tuliskan secara explisit UNION ALL. Perbedaan Union dan Union All dapat dijelaskan pada gambar diagram Venn 5 dan 6.



Gambar 5. Union



Gambar 6. Union All

Fungsi Union sendiri dapat dijalankan dengan sintaks sebagai berikut :

```
SELECT A1, A2, . . .An

FROM r1 UNION
SELECT A1, A2, . . .An
```

```
FROM r2;


SELECT A1, A2, . . .An

FROM r1 UNION ALL
SELECT A1, A2, . . .An
FROM r2;
```

D. LATIHAN

1. Relationship

Dalam latihan ini digunakan dua buah table bernama **karyawan** dan **departemen** dengan relationship **bekerja pada**. Struktur tabelnya diperlihatkan sebgai berikut :

```
CREATE TABLE karyawan( Nama varchar(30) NOT NULL, id_dep
int(5) NOT NULL
)ENGINE = MyISAM;


CREATE TABLE departemen( id_dep int(5) NOT NULL, nama_dep
varchar(30) NOT NULL, PRIMARY KEY(id_dep)
 )ENGINE = MyISAM;
```

Data yang digunakan adalah sebagai berikut :

Tabel Karyawan

nama	id_dep
Agus	10
Budi	16
Citra	12
Dani	17

Tabel Departemen

id_dep	nama_dep
10	Penelitian
11	Pemasaran
12	SDM
13	Keuangan

2. Inner Join

Sebagaimana dijelaskan, inner join akan mengembalikan data di tabel A dan B yang sesuai. Sebagai contoh, untuk mendapatkan data karyawan yang memiliki departemen, eksekusi pernyataan atau perintah SQL berikut :

```
SELECT *
FROM karyawan INNER JOIN departemen ON karyawan.id_dep =
departemen.id_dep;
```

nama	id_dep	id_dep	nama_dep
Agus	10	10	Penelitian
Citra	12	12	SDM

Selain itu, dapat pula menggunakan bentuk implisit dari *inner join* di atas, yaitu sebagai berikut :

```
SELECT * FROM karyawan, departemen  WHERE karyawan.id_dep
= departemen.id_dep;
```

Bandingkan hasil eksekusi perintah SQL yang menggunakan **INNER JOIN** dengan yang menggunakan bentuk implisitnya !

Dalam pengambilan data ini, kita juga bisa menspesifikasikan *field* terkait. Sebagai contoh, untuk mengambil nama karyawan dan nama departemen yang ditempatinya saja, eksekusi perintah SQL berikut :

```
SELECT karyawan.nama, departemen.nama_dep FROM karyawan
INNER JOIN departemen  ON karyawan.id_dep =
departemen.id_dep
```

nama	nama_dep
Agus	Penelitian
Citra	SDM

Perhatikan bahwa untuk menampilkan field tertentu saja, maka nama field tersebut harus disebutkan secara eksplisit beserta nama tabel tempat field.

Nama tabel dapat pula tidak disebutkan apabila nama *field* yang terkait bersifat unik (berbeda atau tidak ada yang menyamainya). Akan tetapi jika ada 2 atau lebih *field* yang memiliki nama yang sama dan berada di tabel yang berbeda, maka nama tabel wajib disebutkan secara eksplisit untuk menghindari keambiguan.

Agar penulisan SQL lebih efisien, kita dapat memanfaatkan fitur "derived table" (atau alias). Contohnya adalah sebagai berikut :

```
SELECT k.nama, d.nama_dep  FROM karyawan k INNER JOIN
departemen d ON k.id_dep =  d.id_dep
```

Pada pernyataan SQL di atas, tabel karyawan dinotasikan dengan huruf k dan tabel departemen menggunakan huruf d. perhatikan hasil eksekusi perintah SQL tersebut, apakah sama dengan hasil eksekusi perintah SQL sebelumnya (yang tidak menggunakan fitur (derived table) ?

Penggunaan derived table akan semakin efisien manakala kita berurusan dengan banyak field dan banyak tabel. Selain itu, juga menjadikan pernyataan SQL mudah dipahami.

3. Outer Join

Left Outer Join

Contoh penggunaan **LEFT OUTER JOIN** adalah sebagai berikut :

```
SELECT *
FROM karyawan k LEFT OUTER JOIN departemen d ON  k.id_dep
= d.id_dep;
```

nama	id_dep	id_dep	nama_dep
Agus	10	10	Penelitian
Budi	16	*NULL*	*NULL*
Citra	12	12	SDM
Dani	17	*NULL*	*NULL*

Perhatikan baris kedua dan keempat pada hasil eksekusi di atas, apa yang menyebabkan timbulnya **NULL** *value* ?

Apabila diperlukan, kita juga dapat menggunakan klausa **WHERE** di dalam join. Sebagai contoh, untuk mendapatkan data karyawan yang tidak memiliki departemen, eksekusi perintah SQL berikut :

```
SELECT *
FROM karyawan k LEFT OUTER JOIN departemen d ON  k.id_dep
= d.id_dep WHERE d.id_dep IS NULL;
```

nama	id_dep	id_dep	nama_dep
Budi	16	*NULL*	*NULL*
Dani	17	*NULL*	*NULL*

Dari hasil eksekusi di atas, dapat kita ketahui bahwa karyawan yang bernama **Budi** dan **Dani** tidak memiliki departemen (nama departemennya tidak tercatat di dalam tabel departemen).

Right Outer Join

Contoh penggunaan **RIGHT OUTER JOIN** adalah sebagai berikut :

```
SELECT *
FROM karyawan k RIGHT OUTER JOIN departemen d ON k.id_dep
= d.id_dep;
```

nama	id_dep	id_dep	nama_dep
Agus	10	10	Penelitian
Citra	12	12	SDM
NULL	*NULL*	11	Pemasaran
NULL	*NULL*	13	Keuangan

Perhatikan kembali baris kedua dan keempat pada hasil eksekusi di atas, apa yang menyebabkan timbulnya **NULL** *value* ?

Full Outer Join

Beberapa DBMS tidak mendukung fungsionalitas *full outer join*. Meski demikian, join ini dapat disimulasikan dengan memanfaatkan **UNION**. Tekniknya ialah dengan menggabung *left join* dan *right join* seperti perintah SQL berikut :

```
SELECT *
FROM karyawan k LEFT OUTER JOIN departemen d ON  k.id_dep
= d.id_dep
UNION SELECT *
FROM karyawan k RIGHT OUTER JOIN departemen d ON
k.id_dep = d.id_dep;
```

nama	id_dep	id_dep	nama_dep
Agus	10	10	Penelitian
Budi	16	*NULL*	*NULL*
Citra	12	12	SDM
Dani	17	*NULL*	*NULL*
NULL	*NULL*	11	Pemasaran
NULL	*NULL*	13	Keuangan

Cross Join

Cross join pada hakikatnya merupakan *inner join* dimana kondisi join selalu dievaluasi ***true***. Secara matematis, jika A dan B merupakan dua himpunan, maka cross join-nya sama dengan **X**.

Contoh penggunaan **CROSS JOIN** adalah sebagai berikut :

```
SELECT * FROM karyawan CROSS JOIN departemen;
```

Atau dalam bentuk implisitnya :

```
SELECT * FROM karyawan, departemen;
```

nama	id_dep	id_dep	nama_dep
Agus	10	10	Penelitian
Budi	16	10	Penelitian
Citra	12	10	Penelitian
Dani	17	10	Penelitian
Agus	10	11	Pemasaran
Budi	16	11	Pemasaran
Citra	12	11	Pemasaran
Dani	17	11	Pemasaran
Agus	10	12	SDM
Budi	16	12	SDM
Citra	12	12	SDM
Dani	17	12	SDM
Agus	10	13	Keuangan
Budi	16	13	Keuangan
Citra	12	13	Keuangan
Dani	17	13	Keuangan

4. Union

Buatlah tabel baru bernama karyawan2 pada database yang sama. Data tabelnya adalah sebagai berikut :

nama	id_dep
Dani	17
Anisa	18
Bagus	12

Setelah itu coba lakukan penggabungan dengan perintah :

```
SELECT nama, id_dep  FROM karyawan

UNION
SELECT nama, id_dep FROM karyawan2;
```

nama	id_dep
Agus	10
Budi	16
Citra	12
Dani	17
Anisa	18
Bagus	12

Lakukan perintah yang sama namun menggunakan **UNION ALL**. Jelaskan apa perbedaan fungsi **UNION** dan **UNION ALL**.

E. TUGAS PRAKTIKUM

Perhatikan, dalam mengerjakan tugas praktikum ini, sebaiknya pernyataan SQL disimpan di file untuk kemudian dieksekusi.

Tugas praktikum ini menggunakan tabel-tabel yang sudah dibuat sebelumnya. Berikut adalah data-data tabel yang akan digunakan (sesuaikan nilainya agar sama persis).

Tabel Mahasiswa

Nim	nama	jenis_kelamin	alamat
101	Arif	L	Jl. Kenangan
102	Budi	L	Jl. Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Malang

Tabel ambil_mk

Nim	kode_mk
101	PTI447
103	TIK333
104	PTI333
104	PTI777
111	PTI123
123	PTI999

Tabel Matakuliah

kode_mk	nama_mk	sks	semester
PTI447	Praktikum Basis Data	1	3
TIK342	Praktikum Basis Data	1	3
PTI333	Basis Data Terdistribusi	3	5
TIK123	Jaringan Komputer	2	5
TIK333	Sistem Operasi	3	5
PTI123	Grafika Multimedia	3	5
PTI777	Sistem Informasi	2	3

1. Dapatkan banyak mahasiswa yang **tidak mengambil matakuliah**. Selesaikan dengan pendekatan join eksplisit dan implisit.

2. Kelompokan data mahasiswa yang **mengambil matakuliah** berdasarkan jenis kelaminnya, kemudian hitung banyaknya.

3. Dapatkan nama mahasiswa yang **mengambil** matakuliah beserta kode_mk dan nama_mk yang diambilnya. Selesaikan dengan pendekatan join eksplisit dan implisit.

4. Dapatkan nim, nama, dan total sks yang diambil oleh mahasiswa, Dimana total sksnya lebih dari 3 dan kurang dari 6.

5. Dapatkan total sks matakuliah yang diambil oleh mahasiswa terdaftar.

F. TUGAS RUMAH

1. Buatlah database baru dengan nama Universitas. Lalu didalamnya terdapat tabel-tabel berikut : Tabel Instruktur

nip	Nama	jurusan	asal_kota
1	Junrico	Ilmu Komputer	Malang
2	Fahmi Hidayat	Ilmu Komputer	Malang
3	Danio Juan	Ilmu Sejarah	Klaten
4	Bobi Kristian	Sastra Indonesia	Magelang

Tabel matakuliah

kd_mk	nama_mk	sks
PTI101	Algoritma dan Pemograman	3
PTI102	Basis Data	3
PTI103	Visual Basic	3
IS101	Sejarah Indonesia	3
PTI007	Proyek Perangkat Lunak	3
SI102	Sastra Indonesia	3

Tabel ambil_mk

nip	kd_mk	ruangan	jml_mhs
1	PTI102	H5211	40
2	PTI102	H5212	45
2	PTI103	H5206	40

3	IS101	I7312	30
4	IS102	I7322	40

a. Tampilkan kd_mk dan mata kuliah yang jumlah mahasiswanya 40.
b. Tampilkan data Matakuliah yang diajar oleh instruktur Sastra Indonesia.
c. Tampilkan data Instruktur yang tidak mengajar Ilmu Komputer.
d. Tampilkan data Instruktur yang tidak mengajar.

2. Buatlah tabel seperti di bawah ini.

Tabel Customer

customer_id	customer_name	customer_addres
CS001	Aan	Pasuruan
CS002	Hanif	Banyuwangi
CS003	Mirza	Malang
CS004	Tanti	Tegal
CS005	Budie	Kediri

Tabel Orders

order_id	order_date	customer_id	qty	amount
CS001	10-12-2016	CS001	1	40000
CS002	11-01-2017	CS002	2	50000
CS003	12-01-2017	CS005	3	35000

a. Gabungkan kedua tabel tersebut dengan JOIN dan UNION
b. Tampilkan data customer yang memiliki amount order diatas 35000
c. Tampilkan field order_date dan amount dengan syarat tahun order setelah 2016 dan amount order kurang dari 40000

Selamat Mengerjakan