**MODUL 6 JOIN DAN UNION**

# TUJUAN

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

# 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.

-

-

-

-

1. **DASAR TEORI**
2. **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.

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

# Join

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

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

# 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***

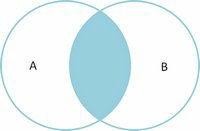
*Inner join* juga dapat direpresentasikan dalam bentuk implisit sebagai berikut :

**SELECT *A1, A2, ..., An***

**FROM *r1, r2***

**WHERE *r1.key = r2.key***

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



*Gambar 1. Inner Join*

# 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 tabel kiri ditambah dengan nilai dari tabel kanan yang sesuai (atau **NULL** jika tidak ada nilai yang sesuai).

Sintaks *left outer join* diperlihatkan sebagai berikut:

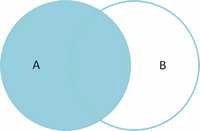
**SELECT *A1*, *A2*, ..., *An***

**FROM *r1***

**LEFT OUTER JOIN *r2***

**ON *r1.join\_key* = *r2.join\_key***

*Left outer join* antara 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:

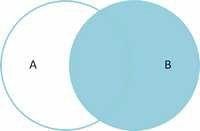
**SELECT *A1*, *A2*, ..., *An***

**FROM *r1***

**RIGHT OUTER JOIN *r2***

**ON *r1.join\_key* = *r2.join\_key***

*Right outer join* antara tabel 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 hakekatnya merupakan kombinasi dari

*left* dan *right* join.

Sintaks *full outer join* diperlihatkan sebagai berikut:

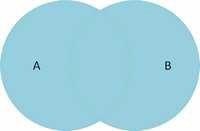
**SELECT *A1*, *A2*, ..., *An***

**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.

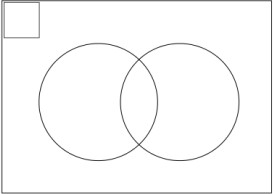
Perlu juga diperhatikan, join bisa diimplementasikan dalam bentuk bersarang (*nested join*). Jadi, di dalam sebuah operasi join bisa terdapat operasi join lainnya.

# Union

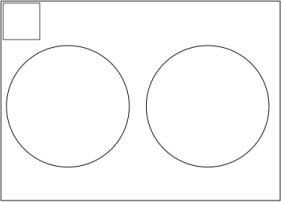
MySQL Union adalah statemen yang mengkombinasikan dua buah atau lebih *resultset* dari beberapa table dengan statemen SELECT sehingga menjadi satu buah *resulset*. Union Statemen memiliki beberapa ketentuan sebagai berikut.

* 1. Jumlah kolom/field dari setiap statemen SELECT harus sama.
  2. Tipe data kolom/field dari setiap statemen SELECT harus kompatibel.

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



*Gambar 5. Union*



*Ganbar 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;**

# LATIHAN

1. **Relationship**

Dalam latihan ini digunakan dua buah tabel bernama **karyawan** dan **departemen** dengan relationship ***bekerja pada***. Struktur tabelnya diperlihatkan sebagai 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 Tabel Departemen

|  |  |
| --- | --- |
| **nama** | **id\_dep** |
| Agus | 10 |
| Budi | 16 |
| Citra | 12 |
| Dani | 17 |

|  |  |
| --- | --- |
| **id\_dep** | **nama\_dep** |
| 10 | Penelitian |
| 11 | Pemasaran |
| 12 | SDM |
| 13 | Keuangan |

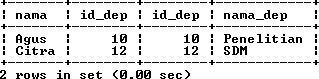
# Inner Join

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;**



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 :

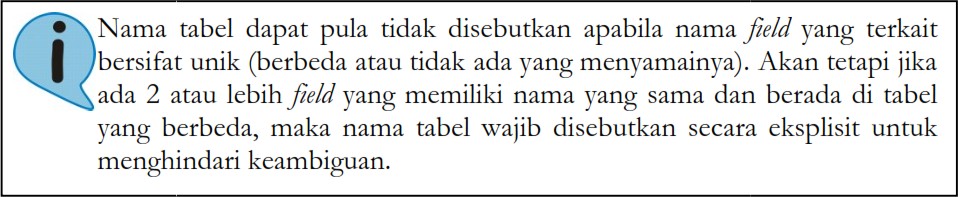


**SELECT karyawan.nama, departemen.nama\_dep FROM karyawan INNER JOIN departemen**

**ON karyawan.id\_dep = departemen.id\_dep;**

Perhatikan bahwa untuk menampilkan *field* tertentu saja, maka nama *field*

tersebut harus disebutkan secara eksplisit beserta nama tabel tempat field.

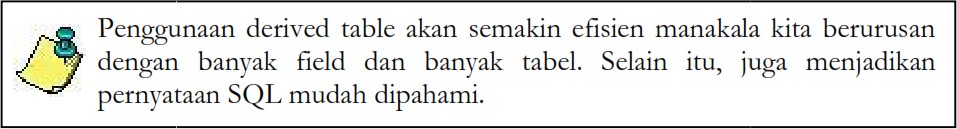


Agar penulisan SQL lebih efisien, kita dapat memanfaatkan fitur “*derived table*” (atau alias). Contohnya adalah sebagai berikut :

**SELECT k.nama, d.nama\_dep**

**FROM karyawan k INNER JOIN departemen d ON k.id\_dep = d.id\_dep;**

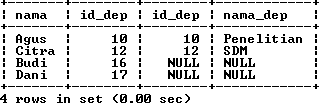
Pada pernyataan SQL di atas, tabel karyawan dinotasikan dengan huruf **k** dan tabel departemen menggunakan huruf **d**. Perhatikan hasil eksekusi perintah SQL tersebut, apakah sama dengan hasil eksekusi perintah SQL sebelumnya (yang tidak menggunakan fitur (*derived table*) ?.



# 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;**

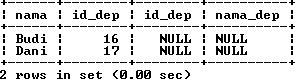
Perhatikan baris kedua dan keempat pada hasil eksekusi di atas, apa yang menyebabkan timbulnya NULL *value*?

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**

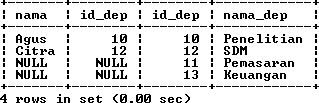
**WHERE d.id\_dep IS NULL;**



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 Joint

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



**SELECT \***

**FROM karyawan k RIGHT OUTER JOIN departemen d ON k.id\_dep = d.id\_dep;**

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

# Full Outer Join

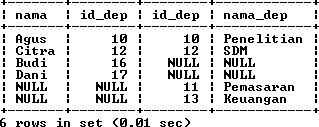
Beberapa DBMS tidak mendukung fungsionalitas *full outer join*. Meski demikian, *join* ini dapat disimulasikan dengan memanfaatkan **UNION**. Tekniknya ialah dengan menggabung *left join* dan *right join* seperti perintah SQL berikut :

**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;**



# Cross Join

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

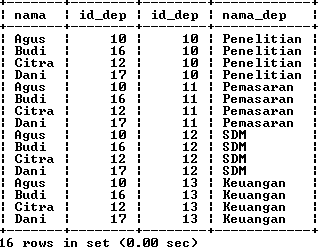
Contoh penggunaan **CROSS JOIN** adalah sebagai berikut :

**SELECT \***

**FROM karyawan CROSS JOIN departemen;**

Atau dalam bentuk implisitnya :

# SELECT \* FROM karyawan, departemen;



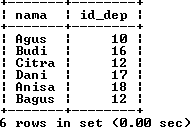
1. **UNION**

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;**

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

# TUGAS PRATIKUM

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 |

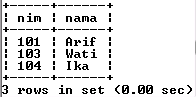
Tabel ambil\_mk

|  |  |
| --- | --- |
| **Nim** | **kode\_mk** |
| 101 | PTI447 |
| 103 | TIK333 |
| 104 | PTI333 |
| 104 | PTI777 |
| 111 | PTI123 |
| 123 | PTI999 |

Tabel Matakuliah

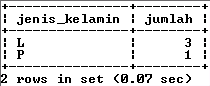
|  |  |  |  |
| --- | --- | --- | --- |
| **kode\_mk** | **nama\_mk** | **sks** | **semester** |
| PTI447 | Praktikum Basis Data | 1 | 3 |
| TIK342 | Praktikum Basis Data | 1 | 3 |
| PTI333 | Basis Data Terdistribusi | 3 | 5 |
| TIK123 | Jaringan Komputer | 2 | 5 |
| TIK333 | Sistem Operasi | 3 | 5 |
| PTI123 | Grafika Multimedia | 3 | 5 |
| PTI777 | Sistem Informasi | 2 | 3 |

1. Dapatkan data mahasiswa yang **mengambil matakuliah**. Selesaikan dengan pendekatan join eksplisit dan implisit. **Hindari duplikasi data**.

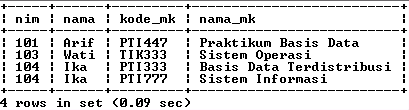


1. Kelompokkan data mahasiswa yang **tidak mengambil matakuliah**

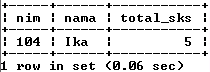
berdasarkan jenis kelaminnya, kemudian hitung banyaknya.



1. Dapatkan nim dan nama mahasiswa yang **mengambil** matakuliah beserta kode\_mk dan nama\_mk yang diambilnya. Selesaikan dengan pendekatan join eksplisit dan implisit.

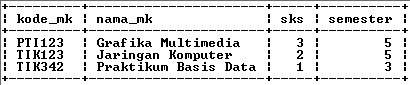


1. Dapatkan nim, nama, dan total sks yang diambil oleh mahasiswa, di mana total sksnya lebih dari 4 dan kurang dari 10.



1. Dapatkan data matakuliah yang **tidak diambil** oleh mahasiswa **terdaftar**

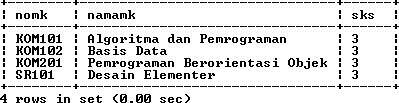
(mahasiswa yang terdaftar adalah mahasiswa yang tercatat di tabel mahasiswa).

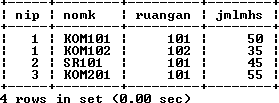


# TUGAS RUMAH

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

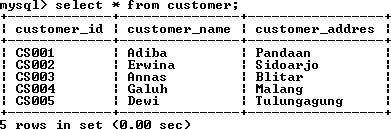
Tabel Instruktur

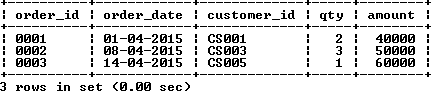
Tabel Matakuliah

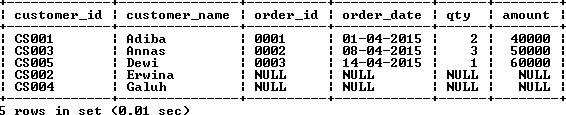
Table Kuliah

* 1. Tampilkan Nomor dan mata kuliah yang pesertanya lebih dari 40 orang.
  2. Tampilkan nip dan nama instruktur yang mengajar mata kuliah 'Basis Data'
  3. Tampilkan Total jumlah total mahasiswa yang Diajar oleh 'Steve Wozniak' menggunakan JOIN.

1. Buatlah Tabel seperti di bawah ini. Tabel Customer



Tabel Orders

Gabungkan kedua tabel itu menggunakan JOIN dan UNION sehingga menjadi seperti di bawah ini.

# Selamat Mengerjakan!