Shape, rectangle

Description automatically generatedA blue and white background

Description automatically generated

**Team Teaching:**

Dwi Puspitasari, S.Kom., M.Kom.

Yan Watequlis Syaifudin, ST., MMT., PhD.

Annisa Puspa Kirana, S. Kom, M.Kom

Yoppy Yunhasnawa, S.ST., M.Sc.

Candra Bella Vista, S.Kom., MT

Elok Nur Hamdana, S.T., M.T

Annisa Taufika Firdausi, ST., MT

Muhammad Shulhan Khairy, S.Kom, M.Kom

Farid Angga Pribadi, S.Kom.,M.Kom

Retno Damayanti, S.Pd., M.T.

PERTEMUAN 13

SELECT Multi-Tabel

**JOBSHEET**

PRAKTIKUM BASIS DATA

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG



|  |  |
| --- | --- |
|  | Jurusan Teknologi Informasi Politeknik Negeri Malang  **Jobsheet-13: SELECT Multi-Tabel**  **Mata Kuliah Basis Data**  Pengampu: Tim Ajar Basis Data  *Mei 2024* |

**Topik**

SELECT pada lebih dari satu tabel

**Tujuan**

Mahasiswa diharapkan memahami dan mampu menggunakan statement:

1. SELECT - INNER JOIN.
2. SELECT - LEFT OUTER JOIN.
3. SELECT - RIGHT OUTER JOIN.
4. SELECT - CROSS JOIN
5. Implicit JOIN

**Pendahuluan**

**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** merupakan salah satu konstruksi dasar dari SQL dan basis data. Join dapat didefinisikansebagai 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: INNER dan OUTER.

1. **INNER JOIN**

Inner join pada dasarnya adalah menemukan persimpangan (intersection) antara dua buah tabel. Sintaks *inner join* diperlihatkan sebagai berikut:



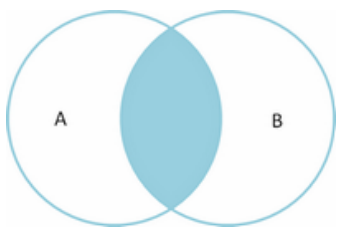
**Cara 1**  
SELECT A1, A2, ..., An   
FROM r1   
INNER JOIN r2   
ON r1.join\_key = r2.join\_key

*Inner join* juga dapat direpresentasikan dalam bentuk implisit.



**Cara 2**  
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.** Representasi INNER JOIN

1. **OUTER JOIN**

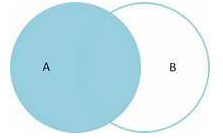
**LEFT OUTER JOIN**

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).



**Syntax**  
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 dalamdiagram Venn seperti Gambar 2.



**Gambar 2.** Representasi LEFT OUTER JOIN

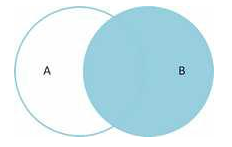
**RIGHT OUTER JOIN**

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:



**Syntax**  
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 dalamdiagram Venn seperti Gambar 3.



**Gambar 3.** Representasi RIGHT OUTER JOIN

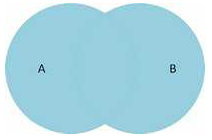
**FULL OUTER JOIN**

Full outer join (atau full join) pada hakekatnya merupakan kombinasi dari left dan right join. Sintaks *full outer join* diperlihatkan sebagai berikut:



**Syntax**  
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.** Representasi 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.

**Praktikum – Bagian 1: Menyiapkan Database untuk Percobaan**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| 1 | Bersama jobsheet ini disertakan sebuah file bernama **percobaan\_join.sql**, eksekusilah file tersebut pada server MySQL Anda dengan cara apapun yang Anda bisa.  Pada contoh di bawah ini, SQL dieksekusi dengan menggunakan perintah **SOURCE** melalui MySQL Shell. Anda juga dapat mengimpor SQL tersebut melalui PHPMyAdmin atau MySQL Workbench, atau tools yang lain yang Anda sukai. |
|  |  |
| 2 | Jika Anda telah berhasil mengimpor/mengeksekusi/menjalankan file **percobaan\_join.sql** tersebut, maka di server MySQL Anda akan dibuatkan database dengan struktur seperti berikut. |
|  |  |
| 3 | Pastikan pada database Anda terdapat tabel-tabel seperti di bawah ini dengan menjalankan SQL dibawah. Jika tabel-tabelnya sudah sesuai, lanjutkan ke **Praktikum – Bagian 2**. |
|  |  |

**Praktikum – Bagian 2: INNER JOIN**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| 1 | Apabila kita ingin menampilkan data yang kolom-kolomnya terdapat pada tabel yang berbeda, maka kita dapat menggunakan sintaksis INNER JOIN. Sintaksis ini akan menampilkan nilai kolom pada 2 atau lebih tabel **yang saling bersesuaian** dalam 1 baris.  Contoh: “Tampilkan nama kelas berikut nama prodinya!”  Solusi: Jalankan query berikut ini, dan akan ditampilkan **52 baris** nama kelas berikut nama prodinya yang bersesuaian (kolom kode\_prodi di tabel kelas sama nilainya dengan kolom kode\_prodi di tabel prodi). |
|  |
| 2 | Pada contoh sebelumnya, jika diperhatikan baik-baik, setiap nama kolom yang ingin ditampilkan harus disebutkan nama tabel asalnya dengan notasi dot/titik (nama\_tabel.nama\_kolom). Hal ini masuk akal karena hal tersebut memiliki tujuan untuk menghilangkan ambiguitas karena bisa saja 2 tabel yang berbeda memiliki kolom dengan nama yang sama.  Namun demikian akan sedikit merepotkan apabila kita secara berulang-ulang menuliskan nama tabel di sebelah nama kolom, lagi dan lagi. Apalagi jika nama tabelnya Panjang.  Solusinya adalah dengan menggunakan alias yaitu sintaksis **AS**. Dengan menggunakan sintaksis ini, query kita akan menjadi lebih singkat namun dengan hasil yang sama. Jalankan SQL berikut untuk mengetahui hasilnya! |
|  |
| 3 | Pada contoh sebelumnya kita telah menampilkan 2 kolom yang terletak pada 2 tabel yang berbeda namun penggunaan INNER JOIN tidaklah terbatas pada 2 tabel saja. Kita juga dapat menampilkan data yang lebih banyak dari beberapa tabel sekaligus.  Contoh: “Tampilkan nama dosen berikut kelas yang diajar dan harinya!”  Solusi: Jalankan SQL berikut. Jika benar akan ditampilkan **320 baris**. Pada query tersebut dilibatkan 4 tabel yaitu tabel jadwal, dosen, kelas, dan hari. |
|  |
| 4 | SELECT JOIN juga bisa difilter. Tentu saja dengan menggunakan klausa WHERE.  Contoh: “Tampilkan nama kelas berikut nama prodinya, hanya untuk kelas yang A saja!”  Solusi: Jalankan SQL berikut ini. Jika benar akan ditampilkan 7 baris yaitu semua kelas dari kelas 1-4 di masing-masing prodi yang namanya diakhiri huruf “A” |
|  |
| 5 | Lanjutkan ke **Praktikum – Bagian 3**. |

**Praktikum - Bagian 3: OUTER JOIN**

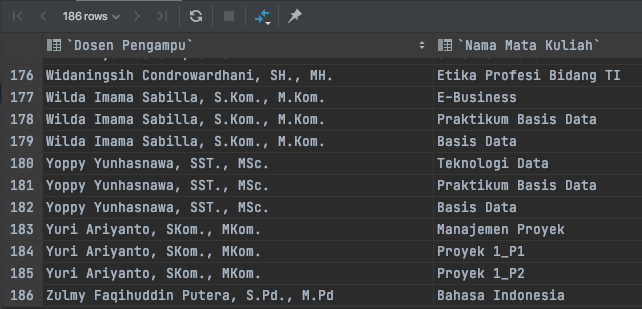
|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| 1 | Jika INNER JOIN hanya menampilan baris-baris pada 2 atau lebih tabel **yang saling bersesuaian**. Maka untuk menampilkan data pada tabel yang **saling bersesuaian ditambah yang tidak bersesuaian**, kita dapat menggunakan sintaksis OUTER JOIN.  OUTER JOIN dibagi menjadi 2:   * LEFT OUTER JOIN dan; * RIGHT OUTER JOIN   OUTER JOIN pada umumnya beguna untuk mengecek data yang tidak ada pasangannya di tabel yang di-JOIN-kan.  Contoh: “Tampilkan data semua kelas berikut nama prodinya, **beserta kelas yang tidak ada prodinya!**”  Solusi: Jalankan query berikut. Jika benar akan ditampilkan **57 baris** data dimana 52 baris adalah nama kelas yang ada prodinya, dan 5 baris sisanya adalah nama kelas yang tidak terdaftar di prodi manapun. |
|  |
| 2 | Apabila hanya ingin menampilkan data yang **tidak ada pasangannya** saja, maka kita bisa menggunakan filter melalui penambahan klausa WHERE.  Query berikut akan menampilkan data seperti sebelumnya, namun **hanya yang tidak ada pasangannya** saja. |
|  |
| 3 | Selain **LEFT** OUTER JOIN juga ada **RIGHT** OUTER JOIN. Keduanya sama-sama menampilkan data yang bersesuaian ditambah yang tidak bersesuaian. Bedanya dalah **letak data yang TIDAK NULL-nya di sebelah mana**. Tabel kiri atau tabel kanan.  Bagaimana menentukan tabel kiri dan tabel kanan?   * Tabel kiri adalah yang ditulis di sebelah **KIRI** (SEBELUM) kata-kata JOIN. * Tabel kanan adalah yang ditulis di sebelah **KANAN** (SESUDAJH)kata-kata JOIN.   LEFT OUTER JOIN 🡪 NULL-nya di tabel kanan, data yang lengkap di tabel **KIRI (LEFT)**  RIGHT OUTER JOIN 🡪 NULL-nya di tabel kiri, data yang lengkap di tabel **KANAN (RIGHT)**  Contoh: “Terdapat prodi baru yang belum ada kelasnya, tampilkan nama kelas berikut nama prodinya serta nama prodi-prodi baru yang belum ada kelasnya tersebut!”  Solusi: Jalankan query berikut. Apabila benar akan ditampilkan 55 baris dimana 52 baris adalah data yang bersesuaian (prodi dan nama kelasnya masing-masing), sedangkan 3 baris sisanya adalah prodi baru yang belum ada kelasnya. Perhatikan ***data yang lengkap ada di kolom nama\_prodi yang merupakan kolom dari tabel prodi yang ditulis di sebelah KANAN (RIGHT) dari kata JOIN*** pada sintaksis SQL-nya. |
|  |
| 4 | Jika kit ingin menampilkan semua data yang bersesuaian, ditambah dengan data yang tidak bersesuaian di tabel KANAN dan KIRI sekaligus, maka kita dapat menggunakan sintaksis FULL JOIN.  Pada beberapa DBMS tertentu, sintaksis eksplisit FULL JOIN telah didukung, namun pada MySQL, sintaksis ini belum didukung.  Untuk mengakalinya kita dapat menggunakan sintaksis UNION ALL yang akan menggabungkan 2 buah himpunan hasil SELECT yang berbeda.  Namun jangan lupa bahwa:   * pada sintaksis ini, kedua buah hasil SELECT harus memiliki jumlah kolom yang sama. Jika tidak, maka datanya tidak akan dapat ditampilkan. * Semikolon (;) harus diletakkan sekali saja di akhir statement SELECT yang paling belakang. * Jangan gunakan UNION saja karena data yang sama (duplikat) akan dihilangkan.   Jalankan query berikut ini untuk menampilkan hasil FULL join terhadap tabel kelas dan tabel prodi. Apabila benar, maka akan ditampilkan sebanyak 112 baris data dengan rincian:   * 104 data yang lengkap ada prodi dan kelasnya. * 5 baris data kelas yang tidak ada prodinya * 3 baris nama prodi yang tidak ada kelasnya |
| … |
| 5 | Lanjutkan ke **Praktikum – Bagian 5**. |

**Praktikum - Bagian 4: INNER JOIN Implisit dan CROSS JOIN**

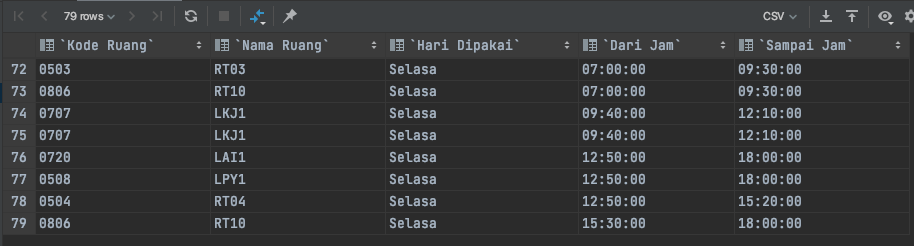
|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| 1 | Sintaksis JOIN yang kita pelajari sebelumnya merupakan sintaksis ANSI SQL yang lebih baru. Sekedar pengetahuan saja, bahwa sebelum distandarkannya format sintaksis tersebut, sebelumnya JOIN dilakukan dengan menggunakan sintaksis yang **tidak ada kata JOIN-nya**. Format ini disebut sebagai **IMPLICIT JOIN**. Sedangkan format sintaksis kita sebelumnya disebut sebagai **EXPLICIT JOIN**.  Kita dianjurkan untuk menggunakan format yang baru, yaitu yang **ada kata JOIN-nya** karena cenderung lebih jelas dan menghindari terjadinya kesalahan maksud pada SQL yang kita tulis. Dengan menuliskan kata-kata JOIN, berarti kita secara sadar memang ingin menampilkan data dari 2 tabel atau lebih.  Namun demikian untuk sekedar pengetahuan Anda, jalankan sintaksis SQL berikut, hasilnya akan sama dengan hasil pada Praktikum – Bagian 1 Langkah 1.  Perhatikan pada sintaksis ini tidak ada kata-kata “JOIN”-nya. |
|  |
| 2 | INNER JOIN Implisit juga bisa diberikan alias nama tabel agar tidak terlalu Panjang.  Alias nama tabel dituliskan pada klausa FROM, sama seperti pada Explicit JOIN.  Jalankan query berikut. Hasilnya akan sama dengan langkah sebelumnya, namun dengan penulisan SQL yang lebih singkat. |
|  |
| 3 | Selain INNER JOIN, OUTER JOIN, dan FULL JOIN, terdapat satu jenis JOIN lagi yaitu CROSS JOIN.  CROSS JOIN akan menampilkan **kombinasi satu-satu** dari setiap kolom pada semua tabel tanpa mempedulikan kesesuaian antar tabel atau tidak.  Jumlah baris yang dihasilkan adalah perkalian jumlah baris pada kedua buah tabel yang di CROSS-kan. |
|  |
| 4 | CROSS JOIN juga dapt dilakukan secara IMPLICIT yaitu dengan tanpa memberikan WHERE pada sintaksis JOIN IMPLICIT.  Jalankan query di bawah ini, hasilnya akan sama dengan langkah sebelumnya. |
|  |

**Tugas**

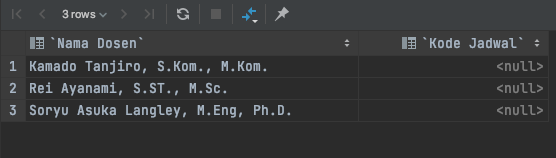
1. Jalankan semua SQL pada praktikum-praktikum di atas, pahami maksudnya dan *Screenshot*-lah hasilnya!
2. Tampilkan nama dosen berikut mata kuliah yang mereka ampu (186 baris) dengan ketentuan:
   1. Tidak ada data yang duplikat
   2. Urut berdasarkan nama dosen dari A-Z.
   3. Nama kolom yang ditampilkan harus sesuai dengan contoh di bawah.



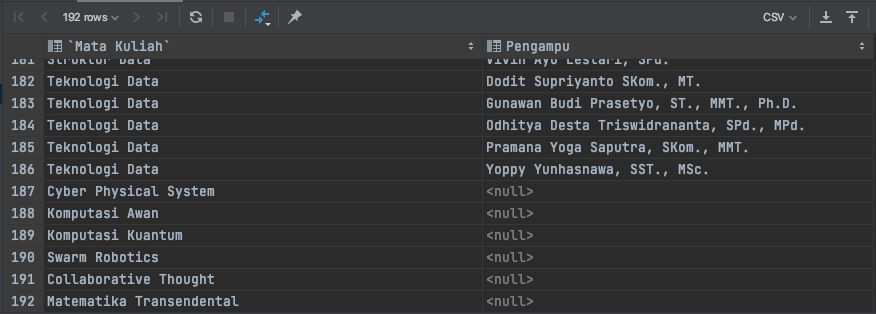
1. Tampilkan ruang yang digunakan untuk perkuliahan pada hari 'Selasa' berikut jam-nya (79 baris) dengan ketentuan:
   1. Harus ditampilkan ruangan tersebut dipakai mulai dari jam berapa sampai dengan jam berapa.
   2. Nama kolom yang ditampilkan harus sesuai contoh di bawah.



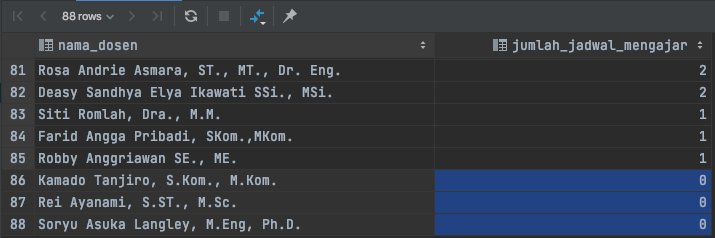
1. Tampilkan dosen yang tidak mendapatkan jadwal mengajar! Catatan: Nama kolom harus sesuai contoh.



1. Tampilkan nama mata kuliah berikut dosen pengampunya berikut (192 baris) yang tidak ada dosen pengampunya! Ketentuan:
   1. Nama Kolom harus sesuai contoh
   2. Anda bisa menggunakan RIGHT OUTER JOIN agar lebih mudah



1. Tampilkan nama-nama dosen (88 baris) berikut jumlah jadwal mereka! Ketentuan:
   1. Nama kolom harus sesuai contoh
   2. Apabila ada dosen yang tidak mendapatkan jadwal, jumlah\_jadwal\_mengajarnya haruslah = 0



***-- Selamat Mengerjakan --***