TUTORIAL Join dan SubQuery

Oleh: Yan Watequlis Syaifudin, Ph.D.

1. KONSEPTUAL

T-SQL (Transact-SQL) mendukung beberapa jenis query join untuk menggabungkan data dari dua atau lebih tabel. Berikut adalah beberapa jenis query join beserta penjelasan singkatnya:

1. **INNER JOIN**:
    - Mengembalikan baris ketika ada kecocokan di kedua tabel. Data yang tidak memiliki kecocokan di salah satu tabel tidak akan ditampilkan.
    - **Contoh**:
      ```
      SELECT a.*, b.*  FROM TableA a
      INNER JOIN TableB b ON a.ID = b.A_ID;
      ```
    - Dalam contoh ini, hanya baris yang memiliki kecocokan pada ID dari TableA dan A_ID dari TableB yang akan ditampilkan.

2. **LEFT JOIN (atau LEFT OUTER JOIN)**:
    - Mengembalikan semua baris dari tabel kiri (TableA), dan baris yang cocok dari tabel kanan (TableB). Jika tidak ada kecocokan, baris dari tabel kanan akan menampilkan NULL.
    - **Contoh**:
      ```
      SELECT a.*, b.*  FROM TableA a
      LEFT JOIN TableB b ON a.ID = b.A_ID;
      ```
    - Ini akan menampilkan semua baris dari TableA, dengan data dari TableB jika ada kecocokan; jika tidak ada, akan muncul NULL.

3. **RIGHT JOIN (atau RIGHT OUTER JOIN)**:
    - Mengembalikan semua baris dari tabel kanan (TableB), dan baris yang cocok dari tabel kiri (TableA). Jika tidak ada kecocokan, baris dari tabel kiri akan menampilkan NULL.
    - **Contoh**:
      ```
      SELECT a.*, b.*   FROM TableA a
      RIGHT JOIN TableB b ON a.ID = b.A_ID;
      ```
    - Ini akan menampilkan semua baris dari TableB, dengan data dari TableA jika ada kecocokan; jika tidak ada, akan muncul NULL.

4. **FULL JOIN (atau FULL OUTER JOIN)**:
    - Mengembalikan semua baris ketika ada kecocokan di salah satu tabel. Jika tidak ada kecocokan, akan menampilkan NULL di sisi yang tidak memiliki kecocokan.
    - **Contoh**:
      ```
      SELECT a.*, b.*  FROM TableA a
      FULL JOIN TableB b ON a.ID = b.A_ID;
      ```
    - Di sini, semua baris dari TableA dan TableB akan ditampilkan, baik dengan kecocokan maupun tanpa kecocokan.

5. **CROSS JOIN**:
    - Menghasilkan produk kartesian dari dua tabel, artinya setiap baris dari tabel pertama akan digabungkan dengan setiap baris dari tabel kedua.

- **Contoh**:

```
SELECT a.*, b.* FROM TableA a
CROSS JOIN TableB b;
```

- Hasil dari query ini akan menciptakan kombinasi dari semua baris yang ada di kedua tabel.

6. **SELF JOIN**:
   - Menggabungkan tabel dengan dirinya sendiri. Ini sering digunakan untuk membandingkan baris dalam tabel yang sama.
   - **Contoh**:

```
SELECT a.*, b.*
FROM TableA a
INNER JOIN TableA b ON a.ID = b.RelatedID;
```

   - Di sini, TableA dijoin dengan dirinya sendiri, menggunakan alias untuk membedakan antara dua instance dari tabel yang sama.

2. TUTORIAL

1. Membuat Tabel Mahasiswa

```
CREATE TABLE Mahasiswa (
    NIM VARCHAR(10) PRIMARY KEY,
    Nama VARCHAR(100),
    Umur INT,
    TglLahir DATE,
    Alamat VARCHAR(200)
);
```

Mengisi Tabel Mahasiswa

```
INSERT INTO Mahasiswa (NIM, Nama, Umur, TglLahir, Alamat) VALUES
('001', 'Andi', 21, '2002-03-15', 'Jakarta'),
('002', 'Budi', 22, '2001-02-20', 'Bandung'),
('003', 'Cici', 20, '2003-10-25', 'Surabaya'),
('004', 'Doni', 23, '2000-01-30', 'Medan'),
('005', 'Eko', 21, '2002-05-10', 'Yogyakarta'),
('006', 'Fani', 19, '2004-04-05', 'Semarang'),
('007', 'Gina', 22, '2001-08-15', 'Malang');
```

2. Membuat Tabel Dosen

```
CREATE TABLE Dosen (
    IdDosen INT PRIMARY KEY,
    NamaDosen VARCHAR(100),
    Bidang VARCHAR(100),
    Alamat VARCHAR(200),
    Status VARCHAR(50)
);
```

Mengisi Tabel Dosen

```
INSERT INTO Dosen (IdDosen, NamaDosen, Bidang, Alamat, Status) VALUES
(1, 'Dr. Ahmad', 'Matematika', 'Jakarta', 'Aktif'),
(2, 'Dr. Clara', 'Fisika', 'Bandung', 'Aktif'),
(3, 'Prof. Budi', 'Kimia', 'Surabaya', 'Tidak Aktif'),
(4, 'Dra. Siti', 'Biologi', 'Yogyakarta', 'Aktif'),
(5, 'Dr. Rudi', 'Informatika', 'Medan', 'Aktif');
```

3. Membuat Tabel MataKuliah

```
CREATE TABLE MataKuliah (
    KodeMK VARCHAR(10) PRIMARY KEY,
    NamaMK VARCHAR(100),
    SKS INT,
    Jam TIME
);
```

Mengisi Tabel MataKuliah

```
INSERT INTO MataKuliah (KodeMK, NamaMK, SKS, Jam) VALUES
('MK001', 'Matematika Dasar', 3, '08:00'),
```

```
('MK002', 'Fisika Dasar', 3, '10:00'),
('MK003', 'Kimia Dasar', 3, '12:00'),
('MK004', 'Biologi Dasar', 2, '14:00'),
('MK005', 'Pemrograman I', 4, '16:00');
```

4. Membuat Tabel Nilai

```
CREATE TABLE Nilai (
    NIM VARCHAR(10),
    KodeMK VARCHAR(10),
    NilaiHuruf CHAR(1),
    PRIMARY KEY (NIM, KodeMK),
    FOREIGN KEY (NIM) REFERENCES Mahasiswa(NIM),
    FOREIGN KEY (KodeMK) REFERENCES MataKuliah(KodeMK)
);
```

Mengisi Tabel Nilai

```
INSERT INTO Nilai (NIM, KodeMK, NilaiHuruf) VALUES
('001', 'MK001', 'A'),
('001', 'MK002', 'B'),
('001', 'MK003', 'A'),
('002', 'MK001', 'C'),
('002', 'MK004', 'B'),
('003', 'MK002', 'B'),
('003', 'MK003', 'A'),
('004', 'MK005', 'D'),
('005', 'MK001', 'B'),
('006', 'MK002', 'A');
```

5. Membuat Tabel IndexNilai

```
CREATE TABLE IndexNilai (
    NilaiHuruf CHAR(1) PRIMARY KEY,
    NilaiIndex INT
);
```

Mengisi Tabel IndexNilai

```
INSERT INTO IndexNilai (NilaiHuruf, NilaiIndex) VALUES
('A', 4),
('B', 3),
('C', 2),
('D', 1),
('E', 0);
```

6. Membuat Tabel JadwalDosen

```
CREATE TABLE JadwalDosen (
    IdDosen INT,
    KodeMK VARCHAR(10),
    Hari VARCHAR(10),
    JamMasuk TIME,
    JamKeluar TIME,
```

```
    PRIMARY KEY (IdDosen, KodeMK),
    FOREIGN KEY (IdDosen) REFERENCES Dosen(IdDosen),
    FOREIGN KEY (KodeMK) REFERENCES MataKuliah(KodeMK)
);
```

Mengisi Tabel JadwalDosen

```
INSERT INTO JadwalDosen (IdDosen, KodeMK, Hari, JamMasuk, JamKeluar)
VALUES
(1, 'MK001', 'Senin', '08:00', '10:00'),
(1, 'MK002', 'Selasa', '10:00', '12:00'),
(2, 'MK003', 'Rabu', '12:00', '14:00'),
(3, 'MK004', 'Kamis', '14:00', '16:00'),
(4, 'MK005', 'Jumat', '08:00', '10:00'),
(5, 'MK001', 'Sabtu', '09:00', '11:00');
```

INNER JOIN

INNER JOIN adalah salah satu jenis operasi penyambungan (join) dalam SQL yang digunakan untuk menggabungkan baris dari dua atau lebih tabel. Operasi ini mengembalikan baris yang memiliki kecocokan dalam kedua tabel berdasarkan kondisi yang ditentukan. Jika ada data yang tidak memiliki pasangan di salah satu tabel, baris tersebut tidak akan muncul dalam hasil query.

Karakteristik INNER JOIN:

1. **Mengikutsertakan hanya baris yang sesuai**: Hanya baris yang memiliki kecocokan di kedua tabel yang akan ditampilkan dalam hasil.
2. **Kondisi JOIN**: INNER JOIN menggunakan kolom kunci primari (primary key) dan kunci tamu (foreign key) untuk menentukan kecocokan.
3. **Dapat bergabung lebih dari dua tabel**: INNER JOIN dapat digunakan untuk menggabungkan lebih dari dua tabel, membuatnya sangat berguna dalam analisis data yang kompleks.

Kondisi JOIN: INNER JOIN beroperasi dengan membandingkan kolom kunci dari dua tabel untuk menemukan baris yang sesuai. Baris yang tidak memiliki pasangan di tabel lain tidak akan muncul dalam hasil akhir.

Tutorial Query INNER JOIN

1. INNER JOIN antara Mahasiswa dan Nilai

Tujuan: Menampilkan daftar mahasiswa beserta nilai yang mereka peroleh.

Query:

```
SELECT  M.NIM, M.Nama, N.KodeMK, N.NilaiHuruf
FROM  Mahasiswa M INNER JOIN   Nilai N ON M.NIM = N.NIM;
```

Penjelasan:

- SELECT: Memilih kolom yang ingin ditampilkan (NIM, Nama, KodeMK, dan NilaiHuruf).
- FROM Mahasiswa M: Memilih tabel Mahasiswa dan memberi alias M.
- INNER JOIN Nilai N ON M.NIM = N.NIM: Bergabung dengan tabel Nilai berdasarkan kecocokan NIM pada kedua tabel.

2. INNER JOIN antara Nilai, MataKuliah dan Mahasiswa

Tujuan: Menampilkan daftar semua mahasiswa dengan nama mata kuliah yang diambil dan nilai yang diperoleh.

Query:

```
SELECT  M.NIM,  M.Nama, N.KodeMK, MK.NamaMK, N.NilaiHuruf
FROM  Mahasiswa M INNER JOIN  Nilai N ON M.NIM = N.NIM
INNER JOIN  MataKuliah MK ON N.KodeMK = MK.KodeMK;
```

Penjelasan:

- Menggunakan dua INNER JOIN untuk menggabungkan tabel Mahasiswa, Nilai, dan MataKuliah.
- Kode MK dan nama mata kuliah ditambahkan untuk informasi lebih lengkap.

3. INNER JOIN antara Dosen dan JadwalDosen

Tujuan: Menampilkan jadwal mengajar dosen bersama dengan nama dan mata kuliah yang diajarkan.

Query:

```
SELECT  D.NamaDosen,  MK.NamaMK, JD.Hari, JD.JamMasuk, JD.JamKeluar
```

```
FROM JadwalDosen JD INNER JOIN  Dosen D ON JD.IdDosen = D.IdDosen
INNER JOIN  MataKuliah MK ON JD.KodeMK = MK.KodeMK;
```

Penjelasan:

- Gabungan ini menampilkan nama dosen, mata kuliah yang diajarkan, hari, dan jam mengajar (masuk dan keluar).

LEFT JOIN

LEFT JOIN (atau **LEFT OUTER JOIN**) adalah salah satu jenis operasi JOIN dalam SQL yang digunakan untuk menggabungkan baris dari dua tabel. Jenis JOIN ini mengembalikan semua baris dari tabel di sebelah kiri (left table) dan baris yang cocok dari tabel di sebelah kanan (right table). Jika tidak ada pasangan yang cocok di tabel kanan, hasil dari tabel kanan akan mengisi NULL.

Konsep Dasar

- **Tabel Kiri**: Merupakan tabel utama yang semua barisnya akan ditampilkan dalam hasil, terlepas apakah ada kecocokan di tabel sebelah kanan.
- **Tabel Kanan**: Baris yang dikembalikan berdasarkan kecocokan dengan tabel kiri. Jika tidak ada kecocokan, nilai dari tabel kanan akan menjadi NULL.

Kapan Menggunakan LEFT JOIN

- Anda ingin memastikan bahwa semua data dari tabel kiri ditampilkan, meskipun data yang relevan dari tabel kanan mungkin tidak ada.
- Ketika Anda ingin mendapatkan gambaran lengkap dari satu tabel serta data kelompok dari tabel lain yang saling terkait.

Contoh Tabel

Mari kita gunakan tabel-tabel yang telah didefinisikan dalam contoh sebelumnya:

1. **Tabel Mahasiswa**:
    - NIM: Nomor Induk Mahasiswa
    - Nama: Nama mahasiswa
    - Umur: Umur mahasiswa
    - TglLahir: Tanggal lahir
    - Alamat: Alamat mahasiswa
2. **Tabel Nilai**:
    - NIM: Nomor Induk Mahasiswa (kunci asing dari tabel Mahasiswa)
    - KodeMK: Kode mata kuliah (kunci asing dari tabel MataKuliah)
    - NilaiHuruf: Nilai huruf yang diperoleh mahasiswa

Tutorial LEFT JOIN

Kasus Penggunaan

Tujuan: Menampilkan daftar semua mahasiswa dan nilai yang mereka peroleh, termasuk mahasiswa yang belum memiliki nilai.

Query LEFT JOIN

```
SELECT   M.NIM, M.Nama, N.KodeMK, N.NilaiHuruf
FROM   Mahasiswa M  LEFT JOIN   Nilai N ON M.NIM = N.NIM;
```

Penjelasan Query

- **SELECT**: Memilih kolom-kolom yang ingin ditampilkan dari tabel-tabel yang dijoin, yaitu NIM dan Nama dari tabel Mahasiswa, serta KodeMK dan NilaiHuruf dari tabel Nilai.
- **FROM Mahasiswa M**: Memulai dari tabel Mahasiswa dan memberikan alias M.
- **LEFT JOIN Nilai N ON M.NIM = N.NIM**: Menggabungkan tabel Mahasiswa dengan tabel Nilai, berdasarkan kecocokan NIM antara kedua tabel. Jika mahasiswa tidak memiliki nilai, maka kolom di tabel Nilai akan berisi NULL.

Hasil yang Diharapkan

- **Format Hasil**:
    - Semua mahasiswa akan ditampilkan.
    - Jika seorang mahasiswa tidak memiliki nilai di tabel Nilai, KodeMK dan NilaiHuruf akan menunjukkan NULL.

Contoh hasil query mungkin terlihat seperti berikut:

NIM	Nama	KodeMK	NilaiHuruf
001	Andi	MK001	A
002	Budi	MK002	B
003	Cici	NULL	NULL
004	Doni	MK003	C
005	Eko	NULL	NULL

Contoh Lain dengan Tabel MataKuliah

Misalkan kita ingin melihat semua mahasiswa, mata kuliah yang mereka ambil, dan nilai mereka, termasuk mahasiswa yang tidak terdaftar dalam mata kuliah.

```
SELECT   M.NIM, M.Nama,  MK.NamaMK, N.NilaiHuruf
FROM   Mahasiswa M LEFT JOIN  Nilai N ON M.NIM = N.NIM
LEFT JOIN   MataKuliah MK ON N.KodeMK = MK.KodeMK;
```

Penjelasan Query Terakhir

- **LEFT JOIN Nilai N ON M.NIM = N.NIM**: Mengambil semua mahasiswa meskipun mereka tidak terdaftar dengan nilai.
- **LEFT JOIN MataKuliah MK ON N.KodeMK = MK.KodeMK**: Menggabungkan dengan tabel MataKuliah berdasarkan KodeMK. Jika seorang mahasiswa tidak memiliki nilai, MataKuliah yang diambil akan menjadi NULL.
- Di sini kita melakukan dua LEFT JOIN: Pertama untuk bergabung dengan tabel Nilai, dan kedua untuk bergabung dengan tabel MataKuliah.
- Ini memungkinkan kita untuk melihat semua mahasiswa dan informasi tentang mata kuliah serta nilai yang mereka peroleh.

Hasil yang Diharapkan

Dalam hasil ini, Anda akan mendapatkan semua mahasiswa, nama mata kuliah, dan nilai mereka. Jika mahasiswa tidak mengambil mata kuliah tertentu atau tidak memiliki nilai, informasi tersebut akan diisi dengan NULL.

Kesimpulan

- **LEFT JOIN** digunakan untuk mengambil semua data dari tabel kiri, termasuk baris yang tidak cocok dari tabel kanan.
- Ini sangat berguna untuk laporan dan analisis di mana Anda ingin memastikan semua data dari tabel utama ditampilkan meskipun beberapa relasi mungkin hilang.

Dengan memahami dan menggunakan LEFT JOIN, Anda dapat menangani situasi di mana hubungan antara tabel tidak selalu lengkap. Jika Anda memiliki pertanyaan lebih lanjut atau memerlukan contoh tambahan, jangan ragu untuk bertanya!

RIGHT JOIN

RIGHT JOIN, atau lebih tepat disebut **RIGHT OUTER JOIN**, adalah tipe dari operasi JOIN dalam SQL yang digunakan untuk menggabungkan data dari dua tabel. Dalam RIGHT JOIN, semua baris dari tabel di sebelah kanan (right table) akan ditampilkan, bersama dengan baris yang cocok dari tabel di sebelah kiri (left table). Jika ada baris dari tabel kanan yang tidak memiliki pasangan di tabel kiri, hasil dari tabel kiri akan berisi NULL.

Konsep Dasar

- **Tabel Kanan**: Adalah tabel yang semua barisnya akan ditampilkan dalam hasil. Setiap baris dari tabel kanan akan muncul di hasil akhir.
- **Tabel Kiri**: Hanya baris yang cocok dengan tabel kanan yang akan muncul. Jika tidak ada kecocokan, kolom terkait dari tabel kiri akan diisi dengan nilai NULL.

Kapan Menggunakan RIGHT JOIN

- Ketika Anda perlu memastikan bahwa semua data dari tabel kanan selalu ditampilkan, meskipun tidak semuanya memiliki pasangan di tabel kiri.
- Berguna ketika informasi pada tabel kanan adalah fokus utama dan Anda ingin mendapatkan data tambahan dari tabel kiri jika tersedia.

Kasus Penggunaan

Tujuan: Mendapatkan daftar semua mata kuliah beserta nilai mahasiswa yang mengambilnya, termasuk mata kuliah yang belum diambil oleh mahasiswa manapun.

Query RIGHT JOIN

```
SELECT N.KodeMK, MK.NamaMK, N.NIM, M.Nama AS NamaMahasiswa,
N.NilaiHuruf
FROM Nilai N  RIGHT JOIN  MataKuliah MK ON N.KodeMK = MK.KodeMK
LEFT JOIN Mahasiswa M ON N.NIM = M.NIM;
```

Penjelasan Query

- **RIGHT JOIN**: Di sini, Nilai (N) dijadikan tabel utama RIGHT JOIN dengan MataKuliah (MK). Hal ini memastikan semua mata kuliah akan ditampilkan dengan atau tanpa mahasiswa yang mengambilnya.
- **LEFT JOIN**: Menggunakan LEFT JOIN antara Nilai (N) dan Mahasiswa (M) untuk mendapatkan nama mahasiswa jika ada nilai yang terkait.
- **SELECT**: Menentukan kolom yang ingin ditampilkan—KodeMK, Nama mata kuliah, NIM, Nama mahasiswa, dan NilaiHuruf.

Hasil yang Diharapkan

- Semua mata kuliah akan ditampilkan.
- Jika sebuah mata kuliah belum diambil oleh mahasiswa, kolom terkait mahasiswa (NIM dan NamaMahasiswa) dan NilaiHuruf akan menampilkan NULL.

Contoh Kasus

Tujuan: Tampilkan semua mata kuliah beserta nilai mahasiswa, termasuk mata kuliah yang belum diambil oleh mahasiswa manapun.

Query RIGHT JOIN

```
SELECT  N.KodeMK,  MK.NamaMK, N.NIM, M.Nama AS NamaMahasiswa,  N.NilaiHuruf
FROM  Nilai N RIGHT JOIN  MataKuliah MK ON N.KodeMK = MK.KodeMK
```

```
LEFT JOIN  Mahasiswa M ON N.NIM = M.NIM;
```

Penjelasan Query

- **FROM Nilai N**: Memulai dengan tabel Nilai untuk mendapatkan semua kombinasi nilai.
- **RIGHT JOIN MataKuliah MK ON N.KodeMK = MK.KodeMK**: Menggunakan RIGHT JOIN di sini memastikan bahwa semua mata kuliah akan ditampilkan, termasuk yang belum ada nilai tercatat untuk mahasiswa mana pun.
- **LEFT JOIN Mahasiswa M ON N.NIM = M.NIM**: Menggunakan LEFT JOIN untuk menambahkan nama mahasiswa jika ada nilai yang tercatat.

Contoh Hasil

KodeMK	NamaMK	NIM	NamaMahasiswa	NilaiHuruf
MK001	Matematika Dasar	001	Andi	A
MK002	Fisika Dasar	NULL	NULL	NULL
MK003	Kimia Dasar	003	Cici	B
MK004	Biologi Dasar	NULL	NULL	NULL
MK005	Pemrograman I	004	Doni	B

Interpretasi Hasil

- Mata kuliah MK002 dan MK004 muncul dalam hasil meskipun tidak ada mahasiswa yang terdaftar, karena RIGHT JOIN menjamin semua entri dari MataKuliah (tabel kanan) akan ditampilkan.
- Di mana tidak ada mahasiswa yang terdaftar, kolom terkait mahasiswa (NIM, NamaMahasiswa) dan nilai (NilaiHuruf) berisi NULL.

Kesimpulan

- **RIGHT JOIN**: Berguna ketika Anda berfokus untuk menampilkan semua data dari tabel kanan, dengan informasi tambahan dari tabel kiri jika ada.
- **Kombinasi JOIN Lanjutan**: Menggunakan RIGHT JOIN dengan LEFT JOIN dapat memberikan hasil yang lengkap yang menampilkan data utama dari tabel kanan dan detail tambahan dari beberapa sumber (tabel kiri).

FULL OUTER JOIN

FULL OUTER JOIN adalah jenis JOIN dalam SQL yang mengembalikan semua baris dari tabel di sebelah kiri (left table) dan tabel di sebelah kanan (right table). Tidak seperti INNER JOIN, yang hanya mengembalikan baris yang ada di kedua tabel, FULL OUTER JOIN mengembalikan semua data dari kedua tabel dan mengisi NULL di mana tidak ada kecocokan.

Konsep Dasar

- **Mengakomodasi Ketidaklengkapan**: FULL JOIN berguna untuk situasi di mana Anda perlu melihat semua data dari kedua tabel, meskipun mereka tidak memiliki kecocokan sempurna.
- **Data dari Kedua Sisi**: Hasil dari FULL OUTER JOIN mencakup:
    - Semua baris dari tabel kiri dan baris cocok dari tabel kanan.
    - Semua baris dari tabel kanan dan baris cocok dari tabel kiri.
    - NULL di kolom-kolom di mana tidak ada data yang cocok secara silang.

Kapan Menggunakan FULL OUTER JOIN

- Ketika Anda ingin mempertahankan semua record dari dua tabel bersama-sama.
- Untuk memeriksa ketidaklengkapan data atau untuk mendapatkan gambaran keseluruhan tentang kemungkinan hubungan dan entitas pada masing-masing tabel.

Tutorial dan Studi Kasus FULL OUTER JOIN

Studi Kasus dan Eksekusi Praktis Menggunakan Tabel

Kita akan bekerja dengan tabel-tabel berikut:

1. **Mahasiswa**: Memuat informasi dasar mahasiswa.
2. **Nilai**: Menyimpan nilai yang diperoleh mahasiswa dalam mata kuliah tertentu.
3. **MataKuliah**: Menyimpan informasi mata kuliah.

Studi Kasus 1: Mengetahui Semua Mahasiswa dan Mata Kuliah yang Terdaftar

Tujuan: Mendapat semua mahasiswa, nilai mereka, dan mata kuliah yang terdaftar, termasuk data apabila mahasiswa atau mata kuliah tidak tercatat pada satu sama lain.

Query FULL OUTER JOIN

```
SELECT     M.NIM, M.Nama AS NamaMahasiswa, N.KodeMK, MK.NamaMK,
N.NilaiHuruf
FROM     Mahasiswa M  FULL OUTER JOIN   Nilai N ON M.NIM = N.NIM
FULL OUTER JOIN     MataKuliah MK ON N.KodeMK = MK.KodeMK;
```

Penjelasan Query

- **FULL OUTER JOIN Nilai N ON M.NIM = N.NIM**: Menampilkan semua data dari mahasiswa dan mengisi dengan hasil lainnya dari tabel Nilai.
- **FULL OUTER JOIN MataKuliah MK ON N.KodeMK = MK.KodeMK**: Menambahkan mata kuliah ke hasil sehingga semua kombinasi antara mahasiswa dan mata kuliah disertakan.

Contoh Hasil

NIM	NamaMahasiswa	KodeMK	NamaMK	NilaiHuruf
001	Andi	MK001	Matematika Dasar	A
003	Cici	MK002	Fisika Dasar	B

NIM	NamaMahasiswa	KodeMK	NamaMK	NilaiHuruf
NULL	NULL	MK003	Kimia Dasar	NULL
004	Doni	MK004	Biologi Dasar	NULL
NULL	NULL	MK005	Pemrograman I	NULL
002	Budi	NULL	NULL	NULL
005	Eko	NULL	NULL	NULL

Studi Kasus 2: Mengetahui Celah dan Cakupan dalam Data Mahasiswa dan Mata Kuliah

Tujuan: Memeriksa dan mencatat mahasiswa yang tidak mengambil mata kuliah atau mata kuliah yang tidak diambil oleh mahasiswa manapun.

Modifikasi Query

```
SELECT     M.NIM,     M.Nama AS NamaMahasiswa,   MK.KodeMK,
MK.NamaMK, ISNULL(N.NilaiHuruf, 'Belum Terdaftar') AS Nilai
FROM    Mahasiswa M  FULL OUTER JOIN   Nilai N ON M.NIM = N.NIM
FULL OUTER JOIN   MataKuliah MK ON N.KodeMK = MK.KodeMK;
```

Penjelasan:

- **ISNULL(N.NilaiHuruf, 'Belum Terdaftar')**: Menggunakan fungsi ISNULL untuk menunjukkan bahwa mahasiswa belum terdaftar dalam mata kuliah tertentu dengan menampilkan teks 'Belum Terdaftar'. Ini menggantikan NULL dengan teks deskriptif pada kolom nilai.

Hasil:

Hasil ini memungkinkan analisis pada mahasiswa yang sepenuhnya tidak terdaftar dalam kombinasi nilai-mata kuliah apapun serta mata kuliah yang belum pernah diambil. Hasil inilah yang membantu menunjukkan cakupan penuh dan potensi celah, terutama bermanfaat untuk keputusan administrasi.

Kesimpulan

- **FULL OUTER JOIN** digunakan untuk menyatukan semua data, bahkan ketika pasangan data tidak ada di tabel lain.
- Sangat berguna untuk analisis lengkap dan penilaian data - memastikan tidak adanya pemadaman data dari salah satu tabel.
- Studi kasus di atas menyoroti penggantian nilai NULL dengan teks informatif untuk menyoroti informasi terperinci.

CROSS JOIN

CROSS JOIN adalah jenis operasi JOIN dalam SQL yang mengkombinasikan setiap baris dari satu tabel dengan setiap baris dari tabel lain. Ini menghasilkan **produk Cartesian** dari dua tabel, yang berarti jika satu tabel memiliki m baris dan tabel lainnya memiliki n baris, maka produk dari CROSS JOIN akan menghasilkan m * n baris dalam hasil.

Konsep Dasar

- **Tanpa Kondisi**: CROSS JOIN tidak memerlukan kondisi eksklusif untuk bergabung antara dua tabel; karena itu, setiap baris dari tabel pertama akan dipasangkan dengan setiap baris dari tabel kedua.
- **Implementasi Kasus**: Biasanya digunakan untuk situasi di mana Anda ingin menguji semua kemungkinan kombinasi dari dua set data.

Kapan Menggunakan CROSS JOIN

- Untuk menghasilkan semua kombinasi yang mungkin dari dua set data.
- Ketika Anda perlu melihat hubungan potensial antara setiap entri dari dua tabel meskipun tidak ada hubungan yang jelas di antara mereka.

Tutorial dan Studi Kasus untuk CROSS JOIN

Kita akan bekerja dengan tabel-tabel yang telah didefinisikan sebelumnya:

1. **Tabel Mahasiswa**
    - NIM: Nomor Induk Mahasiswa
    - Nama: Nama mahasiswa
2. **Tabel MataKuliah**
    - KodeMK: Kode mata kuliah
    - NamaMK: Nama mata kuliah

Studi Kasus 1: Menghasilkan Semua Kombinasi Mahasiswa dan Mata Kuliah

Tujuan: Melihat semua kemungkinan kombinasi antara mahasiswa dan mata kuliah yang tersedia.

Query CROSS JOIN

```
SELECT  M.NIM,   M.Nama, MK.KodeMK,  MK.NamaMK
FROM    Mahasiswa M  CROSS JOIN   MataKuliah MK;
```

Penjelasan Query

- **SELECT**: Memilih kolom NIM dan Nama dari tabel Mahasiswa serta KodeMK dan NamaMK dari tabel MataKuliah.
- **CROSS JOIN**: Menggabungkan semua baris dari tabel Mahasiswa (M) dengan semua baris dari tabel MataKuliah (MK).

Contoh Hasil

Misalkan kita memiliki data sebagai berikut:

Tabel Mahasiswa

NIM	Nama
001	Andi
002	Budi

NIM	Nama
003	Cici

Tabel MataKuliah

KodeMK	NamaMK
MK001	Matematika Dasar
MK002	Fisika Dasar
MK003	Kimia Dasar

Hasil dari Query CROSS JOIN

NIM	Nama	KodeMK	NamaMK
001	Andi	MK001	Matematika Dasar
001	Andi	MK002	Fisika Dasar
001	Andi	MK003	Kimia Dasar
002	Budi	MK001	Matematika Dasar
002	Budi	MK002	Fisika Dasar
002	Budi	MK003	Kimia Dasar
003	Cici	MK001	Matematika Dasar
003	Cici	MK002	Fisika Dasar
003	Cici	MK003	Kimia Dasar

Studi Kasus 2: Pengujian Kombinasi Potensial

Tujuan: Memeriksa apakah semua mahasiswa mungkin mengambil semua mata kuliah. Ini juga bisa digunakan untuk kebutuhan test case atau simulasi.

Query CROSS JOIN yang Sama

Menggunakan query yang sama seperti sebelumnya, kita dapat melakukan analisis lebih lanjut untuk memverifikasi kemungkinan dan datanya.

```
SELECT M.NIM, M.Nama, MK.KodeMK, MK.NamaMK
FROM Mahasiswa M CROSS JOIN MataKuliah MK;
```

Hasil yang Diharapkan

Hasilnya akan sama seperti yang telah ditunjukkan sebelumnya, dengan setiap mahasiswa dikombinasikan dengan setiap mata kuliah, dan memberikan gambaran lengkap tentang semua potensi enrolmen yang ada.

Kesimpulan

- **CROSS JOIN** digunakan untuk menghasilkan semua kombinasi dari dua tabel tanpa memperhatikan relasi yang ada antara baris-barisnya.

SELF JOIN

SELF JOIN adalah versi dari JOIN yang mengizinkan sebuah tabel untuk digabungkan dengan dirinya sendiri. Ini berguna ketika kita ingin melakukan analisis atau perbandingan berdasarkan data dalam tabel yang sama.

Konsep Dasar

- **Menggunakan Alias**: Dalam self join, kita harus menggunakan alias untuk membedakan antara dua instance dari tabel yang sama. Misalnya, kita mungkin ingin membandingkan dua baris di dalam tabel berdasarkan tertentu.
- **Komparatif**: Berguna untuk analisis data yang membutuhkan perbandingan antar entri, misalnya untuk melihat relasi antar mahasiswa dalam kategori yang sama, seperti nilai atau kelompok.

Kapan Menggunakan SELF JOIN

- Ketika Anda ingin membandingkan data dalam satu tabel.
- Ketika terdapat hierarki atau relasi di antara data di dalam tabel yang sama, seperti karyawan dan manajer dalam tabel yang sama.

Tutorial SELF JOIN

Untuk tutorial ini, kita akan menggunakan tabel **Mahasiswa** yang telah didefinisikan sebelumnya dan kita akan melakukan self join untuk menunjukkan bagaimana mahasiswa bisa dibandingkan berdasarkan atribut yang ada.

Studi Kasus 1: Mencari Mahasiswa dengan Umur yang Sama

Tujuan: Menampilkan mahasiswa yang memiliki umur yang sama.

Struktur Tabel Mahasiswa

NIM	Nama	Umur	TglLahir	Alamat
001	Andi	21	2003-01-15	Jakarta
002	Budi	22	2002-02-20	Bandung
003	Cici	21	2003-10-25	Surabaya
004	Doni	23	2000-01-30	Medan
005	Eko	22	2001-05-10	Yogyakarta

Query SELF JOIN

```
SELECT   A.Nama AS Mahasiswa1,   B.Nama AS Mahasiswa2,   A.Umur
FROM     Mahasiswa A JOIN   Mahasiswa B ON A.Umur = B.Umur
WHERE      A.NIM <> B.NIM;
```

Penjelasan Query

- **SELECT**: Memilih nama dua mahasiswa yang memiliki usia yang sama, dan menampilkan umur mereka.
- **FROM Mahasiswa A**: Memulai dengan tabel Mahasiswa dan memberi alias A.
- **JOIN Mahasiswa B ON A.Umur = B.Umur**: Melakukan join antara dua instance dari tabel yang sama berdasarkan umur.

- **WHERE A.NIM <> B.NIM**: Menyaring untuk memastikan kita tidak membandingkan mahasiswa dengan dirinya sendiri.

Contoh Hasil

Mahasiswa1	Mahasiswa2	Umur
Andi	Cici	21
Cici	Andi	21
Budi	Eko	22
Eko	Budi	22

Studi Kasus 2: Mencari Pasangan Mahasiswa dalam 1 Kelas

Tujuan: Menampilkan semua mahasiswa yang memiliki nama yang sama di dalam satu tabel, mirip dengan melihat data mahasiswa dalam konteks kelompok kelas.

Query SELF JOIN

Untuk skenario ini, kita bisa membuat pendaftaran mahasiswa yang menggunakan nama dan melihat seluruh data mahasiswa yang memiliki nama yang sama.

```
SELECT    A.Nama, COUNT(*) AS Jumlah
FROM   Mahasiswa A  GROUP BY   A.Nama
HAVING      COUNT(*) > 1;
```

Penjelasan Query

- **SELECT A.Nama**: Memilih nama mahasiswa.
- **COUNT(*) AS Jumlah**: Memperoleh jumlah mahasiswa dengan nama yang sama.
- **GROUP BY A.Nama**: Mengelompokkan hasil berdasarkan nama mahasiswa.
- **HAVING COUNT(*) > 1**: Filter untuk hanya menampilkan mahasiswa yang memiliki lebih dari satu entri, yang berarti mereka memiliki nama yang sama.

Contoh Hasil

Nama	Jumlah
Budi	2

Kesimpulan

- **SELF JOIN** memungkinkan Anda untuk beroperasi pada data yang sama dalam konteks perbandingan atau hubungan.
- Sangat berguna untuk analisis di mana entitas dalam tabel yang sama saling berinteraksi, seperti mencari nama kembar atau membandingkan atribut.

SUBQUERY

Subquery, atau juga dikenal sebagai **nested query** atau **inner query**, adalah query yang terletak di dalam query lain. Subquery digunakan untuk menghasilkan data yang menjadi filter, pemilihan, atau operasi lebih lanjut dalam query utama. Subquery dapat ditempatkan di klausa SELECT, FROM, WHERE, atau HAVING.

Konsep Dasar

- **Penerapan**: Subquery memberikan cara untuk mendapatkan subset data yang diperlukan untuk operasi lebih lanjut, memungkinkan pengguna untuk melakukan analisis data yang lebih kompleks dalam satu pernyataan SQL.
- **Keterhubungan**: Subquery biasanya digunakan untuk menyaring, menghitung, atau mendapatkan informasi terkait dari tabel lain, dan dapat menghasilkan hasil yang digunakan dalam query luar.

Kapan Menggunakan Subquery

- Ketika Anda perlu melakukan analisis kompleks di mana hasil dari satu query diperlukan untuk menyaring atau memilih data dari query lain.
- Ketika Anda ingin mendapatkan data agregat untuk membandingkannya dengan data lain di tabel yang berbeda.

Studi Kasus 1: Mendapatkan Mahasiswa dengan Nilai Tertinggi

Tujuan: Menampilkan mahasiswa yang mendapatkan nilai tertinggi dalam satu mata kuliah tertentu.

Query Subquery

Misalkan kita ingin menemukan mahasiswa dengan nilai tertinggi untuk mata kuliah tertentu.

```
SELECT    M.NIM,  M.Nama,  N.NilaiHuruf
FROM   Mahasiswa M JOIN   Nilai N ON M.NIM = N.NIM
WHERE  N.NilaiHuruf = (SELECT MAX(NilaiHuruf) FROM Nilai WHERE KodeMK
= 'MK001');
```

Penjelasan Query

- **SELECT**: Memilih NIM, Nama, dan NilaiHuruf dari mahasiswa.
- **JOIN**: Bergabung dengan tabel Nilai berdasarkan NIM.
- **WHERE N.NilaiHuruf = (SELECT MAX...)**: Subquery menghasilkan nilai maksimum dari kolom NilaiHuruf untuk KodeMK tertentu. Hasil tersebut digunakan untuk memfilter hasil di query utama.

Contoh Hasil

NIM	Nama	NilaiHuruf
001	Andi	A

Studi Kasus 2: Mendapatkan Mahasiswa yang Tidak Memiliki Nilai

Tujuan: Menampilkan mahasiswa yang tidak memiliki nilai terdaftar dalam tabel Nilai.

Query Subquery

```
SELECT   M.NIM, M.Nama
FROM  Mahasiswa M WHERE  M.NIM NOT IN (SELECT NIM FROM Nilai);
```

Penjelasan Query

- **SELECT**: Memilih NIM dan Nama dari tabel Mahasiswa.
- **WHERE M.NIM NOT IN**: Subquery di dalam klausa WHERE menghasilkan daftar NIM yang ada dalam tabel Nilai. Mahasiswa yang NIM tidak ada dalam daftar tersebut akan dipilih.

Contoh Hasil

NIM	Nama
005	Eko

Studi Kasus 3: Menghitung Rata-Rata Nilai Mahasiswa

Tujuan: Menampilkan mahasiswa dengan rata-rata nilai di atas batas tertentu.

Query Subquery

```
SELECT   M.NIM, M.Nama
FROM   Mahasiswa M
WHERE (SELECT AVG(NilaiHuruf) FROM Nilai N WHERE N.NIM = M.NIM) > 'B';
```

Penjelasan Query

- **SELECT**: Memilih NIM dan Nama dari mahasiswa.
- **WHERE (SELECT AVG(NilaiHuruf)**: Subquery menghitung rata-rata nilai mahasiswa tertentu. Jika rata-rata ini lebih tinggi dari ‘B’, mahasiswa tersebut dipilih.
- Penting untuk diingat bahwa dalam banyak situasi nilai harus dinormalisasi (misalnya: menggunakan nilai indeks).