Structured Query Language (SQL) Lanjut

Modul Praktikum Sistem Manajemen Basis Data 2025

Tim Asisten
Sistem Manajemen
Basis Data 2025

Data Definition Language (DDL) : Referential Actions

Referential Actions adalah aturan yang menentukan apa yang terjadi pada data foreign key di tabel anak (child table) ketika data di tabel induk (parent table) berubah atau dihapus.

Referential Action	Yang Akan Terjadi Apabila Parent Dihapus/Diubah
CASCADE	Data anak ikut terhapus atau diperbarui secara otomatis.
RESTRICT	Mencegah penghapusan/perubahan jika masih ada data anak (ditolak oleh sistem MySQL).
SET NULL	Foreign key di data anak diubah menjadi NULL.
NO ACTION	Tidak ada aksi yang dilakukan. Dengan kata lain, referensi tetap ada dan tidak ada tindakan apapun yang diambil pada tabel terkait.

Contoh:

1. Membuat database db_mobil, dan menggunakan db_mobil

```
CREATE DATABASE db_mobil;
USE db_mobil;
```

2. Membuat tabel mobil dan pemilik

```
CREATE TABLE pemilik (
id_pemilik INT NOT NULL AUTO_INCREMENT,
nama VARCHAR(255)NOT NULL,
alamat TEXT NOT NULL,
no_telp VARCHAR(15) NOT NULL,
PRIMARY KEY (id_pemilik)
);
```

3. Cara 1: membuat referential action pada saat membuat tabel

```
CREATE TABLE mobil (
id_mobil INT NOT NULL AUTO_INCREMENT,
plat_nomor VARCHAR(20) NOT NULL,
id_pemilik INT(11),
PRIMARY KEY (id_mobil),
FOREIGN KEY (id_pemilik) REFERENCES pemilik(id_pemilik) ON
UPDATE CASCADE ON DELETE RESTRICT
```

```
);
```

4. Cara 2: Apabila tabel sudah dibuat, tetapi foreign key-nya belum diset, bisa menggunakan alter table.

```
ALTER TABLE mobil
ADD FOREIGN KEY(id_pemilik) REFERENCES pemilik(id_pemilik)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

Note: 'id_pemilik' harus sudah ada pada tabel.

5. Cara 3: Apabila tabel sudah dibuat dan foreign keynya sudah ada, tetapi referential action-nya belum diset, bisa menggunakan alter table dengan menghapus terlebih dahulu foreign key yang sudah dibuat.

   Cari nama constraint foreign key atribut 'id_pemilik' pada tabel 'mobil'

```
SELECT CONSTRAINT_NAME FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE TABLE_NAME =
'mobil' AND COLUMN_NAME = 'id_pemilik';
```

   Drop foreign key

```
ALTER TABLE  mobil
DROP CONSTRAINT mobil_ibfk_1;
```

   Add lagi foreign key-nya

```
ALTER TABLE mobil
ADD CONSTRAINT mobil_ibfk_1 FOREIGN KEY(id_pemilik)
REFERENCES pemilik(id_pemilik)
ON UPDATE CASCADE ON DELETE RESTRICT;
```

6. Mengisi tabel mobil dan pemilik

```
INSERT INTO pemilik (nama, alamat, no_telp) VALUES
('Datuk Daneswara Raditya Samsura', 'Jl. Merdeka No. 10,
Jakarta', '081234567890'),
('Hafsah Hamidah', 'Jl. Sudirman No. 15, Bandung',
'081298765432'),
('Muhammad Alvinza', 'Jl. Ahmad Yani No. 25, Surabaya',
'081377788899'),
('Muhammad Fathan Putra', 'Jl. Gajah Mada No. 30, Medan',
'081255566677'),
('Nashwa Nadria Futi', 'Jl. Diponegoro No. 5, Semarang',
```

```
'081344455566'),
('Raffi Adzril Alfaiz', 'Jl. Pahlawan No. 20, Yogyakarta',
'081322233344'),
('Mr.Sigma', 'Jl. Imam Bonjol No. 12, Malang',
'081399988877');
```

```
INSERT INTO mobil (plat_nomor, id_pemilik) VALUES
('B 1234 ABC', 1),
('D 5678 XYZ', 2),
('B 8765 DEF', 3),
('L 3456 GHI', 4),
('N 7890 JKL', 5),
('B 4567 PQR', 6),
('F 1122 MNO', NULL);
```

7. Coba hapus data pada tabel utama (pemilik)

```
DELETE FROM pemilik WHERE id_pemilik = 1;
```

Data tidak bisa dihapus karena memiliki referential action ON DELETE RESTRICT karena ada id_pemilik 1 yang dipakai di tabel 'mobil'.

```
MariaDB [db_mobil]> DELETE FROM pemilik WHERE id_pemilik = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key co
nstraint fails (`db_mobil`.`mobil`, CONSTRAINT `mobil_ibfk_1` FOREIGN KEY
(`id_pemilik`) REFERENCES `pemilik` (`id_pemilik`) ON UPDATE CASCADE)
```

8. Coba perbarui data pada tabel utama (pemilik)

```
UPDATE pemilik
SET id_pemilik = 10
WHERE id_pemilik = 3;
```

Id_pemilik 3 di tabel 'pemilik' berubah menjadi 10, dan id_pemilik dengan nilai 3 juga berubah pada tabel anak yaitu 'mobil' otomatis berubah menjadi 10.

```
+------------+---------------------------------+
| id_pemilik | nama                            |
+------------+---------------------------------+
|          1 | Datuk Daneswara Raditya Samsura |
|          2 | Hafsah Hamidah                  |
|          4 | Muhammad Fathan Putra           |
|          5 | Nashwa Nadria Futi              |
|          6 | Raffi Adzril Alfaiz             |
|         10 | Muhammad Alvinza                |
+------------+---------------------------------+
```

Over dan Partition by

1. **OVER**

   OVER() adalah adalah fungsi dalam SQL yang digunakan untuk menerapkan perhitungan agregat tanpa mengelompokkan atau menghilangkan baris dalam hasil query. Klausa OVER mirip dengan GROUP BY. OVER akan menambahkan kolom baru yang berisi hasil agregat, tanpa mengubah tampilan data.

2. **PARTITION BY**

   PARTITION BY digunakan dalam OVER() untuk membagi data ke dalam grup tertentu, sehingga perhitungan agregat dilakukan dalam setiap grup tersebut.

   Syntax penggunaan OVER dan PARTITION BY

```
SELECT kolom1, kolom2, ...,
agregat_fungsi(*) OVER(PARTITION BY kolom_pemecah) AS
nama_kolom_baru
FROM nama_tabel;
```

Misalkan ada tabel mobil:

id_mobil	plat_nomor	id_pemilik
1	B 1234 ABC	1
2	D 5678 XYZ	2
3	B 8765 DEF	3
4	L 3456 GHI	1

5	N 7890 JKL	2

Kita akan menggunakan COUNT untuk menampilkan jumlah_mobil per pemilik.

```
SELECT id_pemilik, COUNT(*) AS jumlah_mobil
FROM mobil
GROUP BY id_pemilik;
```

Jika menggunakan GROUP BY, data per id_mobil tidak semuanya ditampilkan secara individu.

id_pemilik	jumlah_mobil
1	2
2	2
3	1

```
SELECT id_mobil, plat_nomor, id_pemilik,
COUNT(*) OVER(PARTITION BY id_pemilik) AS jumlah_mobil
FROM mobil;
```

Jika menggunakan OVER dan PARTITION BY, semua data per id_mobil akan ditampilkan dan lebih rinci karena kolom id_mobil dan plat_nomor tetap ditampilkan.

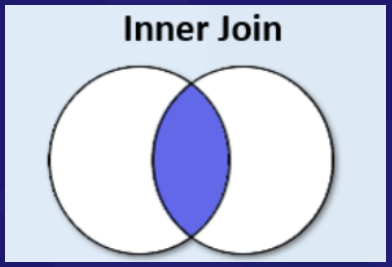
id_mobil	plat_nomor	id_pemilik	jumlah_mobil
1	B 1234 ABC	1	2
4	L 3456 GHI	1	2
2	D 5678 XYZ	2	2
5	N 7890 JKL	2	2
3	B 8765 DEF	3	1

Data Manipulation Language (DML): Join

JOIN adalah perintah dalam SQL yang digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan kolom yang memiliki hubungan.

1. **INNER JOIN**

   Mengembalikan record data yang memiliki nilai yang cocok di kedua tabel atau lebih. (Beririsan)



Syntax 2 tabel:

```
SELECT table1.column_name, table2.column_name
FROM table1 INNER JOIN table2 ON table1.column_name =
table2.column_name;
```

Syntax 3 tabel:

```
SELECT table1.column_name, table2.column_name,
table3.column_name
FROM table1
INNER JOIN table2 ON table1.column_name =
table2.column_name
INNER JOIN table3 ON table1.column_name =
table3.column_name;
```

Contoh:

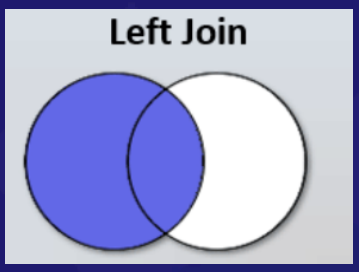
```
SELECT pemilik.id_pemilik, pemilik.nama, mobil.plat_nomor
FROM pemilik
INNER JOIN mobil ON pemilik.id_pemilik = mobil.id_pemilik;
```

Pemilik tanpa mobil (id_pemilik = 7) dan mobil tanpa pemilik (F 1122 MNO) tidak ditampilkan.

id_pemilik	nama	plat_nomor
1	Datuk Daneswara Raditya Samsura	B 1234 ABC
2	Hafsah Hamidah	D 5678 XYZ
3	Muhammad Alvinza	B 8765 DEF
4	Muhammad Fathan Putra	L 3456 GHI
5	Nashwa Nadria Futi	N 7890 JKL
6	Raffi Adzril Alfaiz	B 4567 PQR

2. LEFT JOIN

Mengambil semua data dari tabel **kiri** dan yang cocok dari tabel kanan.



Syntax 2 tabel:

```
SELECT table1.column_name, table2.column_name
FROM table1 LEFT JOIN table2 ON table1.column_name =
table2.column_name;
```

Syntax 3 tabel:

```
SELECT table1.column_name, table2.column_name,
table3.column_name
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name
LEFT JOIN table3 ON table1.column_name =
table3.column_name;
```

Contoh:

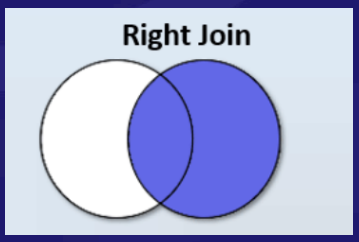
```
SELECT pemilik.id_pemilik, pemilik.nama, mobil.plat_nomor
FROM pemilik
LEFT JOIN mobil ON pemilik.id_pemilik = mobil.id_pemilik;
```

Mr.Sigma tidak memiliki mobil, sehingga plat_nomor = NULL.

id_pemilik	nama	plat_nomor
1	Datuk Daneswara Raditya Samsura	B 1234 ABC
2	Hafsah Hamidah	D 5678 XYZ
3	Muhammad Alvinza	B 8765 DEF
4	Muhammad Fathan Putra	L 3456 GHI
5	Nashwa Nadria Futi	N 7890 JKL
6	Raffi Adzril Alfaiz	B 4567 PQR
7	Mr.Sigma	NULL

3. RIGHT JOIN

Mengambil semua data dari tabel **kanan** dan yang cocok dari tabel kiri.



Syntax 2 tabel:

```
SELECT table1.column_name, table2.column_name
FROM table1 RIGHT JOIN table2 ON table1.column_name =
table2.column_name;
```

Syntax 3 tabel:

```
SELECT table1.column_name, table2.column_name,
table3.column_name
FROM table1
RIGHT JOIN table2 ON table1.column_name =
table2.column_name
RIGHT JOIN table3 ON table1.column_name =
table3.column_name;
```

Contoh:

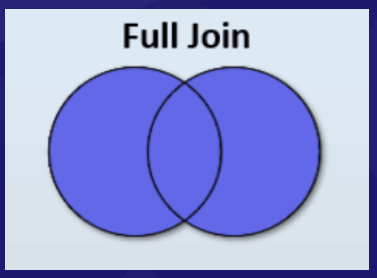
```
SELECT pemilik.id_pemilik, pemilik.nama, mobil.plat_nomor
FROM pemilik
RIGHT JOIN mobil ON pemilik.id_pemilik = mobil.id_pemilik;
```

Mobil "F 1122 MNO" tidak memiliki pemilik, sehingga nama = NULL.

id_pemilik	nama	plat_nomor
1	Datuk Daneswara Raditya Samsura	B 1234 ABC
2	Hafsah Hamidah	D 5678 XYZ
3	Muhammad Alvinza	B 8765 DEF
4	Muhammad Fathan Putra	L 3456 GHI
5	Nashwa Nadria Futi	N 7890 JKL
6	Raffi Adzril Alfaiz	B 4567 PQR
NULL	NULL	F 1122 MNO

4. FULL OUTER JOIN

Menggabungkan `LEFT JOIN` dan `RIGHT JOIN`, menampilkan semua data di tiap tabel.



Syntax 2 tabel:

```
SELECT table1.column_name, table2.column_name
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name

UNION

SELECT table1.column_name, table2.column_name
FROM table1
RIGHT JOIN table2 ON table1.column_name =
table2.column_name;
```

Syntax 3 tabel:

```
SELECT table1.column_name, table2.column_name,
table3.column_name
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name
LEFT JOIN table3 ON table1.column_name = table3.column_name

UNION

SELECT table1.column_name, table2.column_name,
table3.column_name
FROM table1
RIGHT JOIN table2 ON table1.column_name =
table2.column_name
RIGHT JOIN table3 ON table1.column_name =
table3.column_name;
```

Contoh:

```
SELECT pemilik.id_pemilik, pemilik.nama, mobil.plat_nomor
FROM pemilik
LEFT JOIN mobil ON pemilik.id_pemilik = mobil.id_pemilik

UNION
```

```sql
SELECT pemilik.id_pemilik, pemilik.nama, mobil.plat_nomor
FROM pemilik
RIGHT JOIN mobil ON pemilik.id_pemilik = mobil.id_pemilik;
```

Menampilkan semua pemilik dan semua mobil, meskipun tidak ada pasangan.

id_pemilik	nama	plat_nomor
1	Datuk Daneswara Raditya Samsura	B 1234 ABC
2	Hafsah Hamidah	D 5678 XYZ
3	Muhammad Alvinza	B 8765 DEF
4	Muhammad Fathan Putra	L 3456 GHI
5	Nashwa Nadria Futi	N 7890 JKL
6	Raffi Adzril Alfaiz	B 4567 PQR
7	Mr.Sigma	NULL
NULL	NULL	F 1122 MNO

Tugas

ez bgt reallllll

Penutup

Kami mengucapkan terima kasih kepada Tuhan, Bangsa, dan Almamater. Yang telah berkontribusi dalam pembuatan modul ini. Semoga pengetahuan dan pengalaman yang diperoleh dapat memberikan manfaat yang signifikan bagi masa depan. Aamiin.

Referensi

Asisten Praktikum Sistem Basis Data (2024). Modul Sistem Basis Data :

Modul 5: SQL Lanjut