Tugas Basis Data 002

Nama : Muhammad Aspian
Kelas : XI RPL 2
Guru Pengajar : Hidayatullah

SMK NEGERI 4 BANJARMASIN

1. Persiapan

Untuk sisi *server*, saya menggunakan kontainer dengan image *Percona* karena beberapa masalah yang muncul saat menjalankan image *MySQL*.

```
~
❯ podman run -dp 3306:3306 --name sql -e MYSQL_ROOT_PASSWORD="uh" docker.io/percona
2970356385e9e16ac51694a57707fb0128ebb172c1280f6a5af513b2f4baee0c
```

Pada bagian *client* saya menggunakan *mycli* dikarenakan banyaknya fitur dibandingkan dengan *client* bawaan. Sesuai dengan ketentuan, nama database yang digunakan adalah **"tugas_dbs_<nama siswa>"**.

```
~
❯ PYTHONWARNINGS='ignore' mycli mysql://root:'uh'@localhost
《(none)》❯ create database tugas_dbs_aspian;
Query OK, 1 row affected
Time: 0.001s
《(none)》❯ use tugas_dbs_aspian;
You are now connected to database "tugas_dbs_aspian" as user "root"
Time: 0.001s
《tugas_dbs_aspian》❯
```

2. Pembuatan Tabel

Dalam kasus ini, tabel tidak dapat dibuat tanpa perencanaan yang jelas. Tabel harus disusun berdasarkan urutan yang sistematis. Setelah melakukan analisis, saya memutuskan untuk membuat tabel dengan urutan: tujuan, supir, armada, kursi, dan penumpang.

1. Tabel Tujuan

```
『Aspian』❯ CREATE TABLE tujuan(
          → kode_tujuan VARCHAR(11)
          → PRIMARY KEY,
          → tujuan TEXT,
          → harga VARCHAR(11),
          → jam VARCHAR(17));
Query OK, 0 rows affected
Time: 0.059s
```

2. Tabel Supir

```
『Aspian』❯ CREATE TABLE supir(
          → kode_supir VARCHAR(7),
          → nama VARCHAR(75),
          → jk VARCHAR(11),
          → alamat TEXT,
          → telepon VARCHAR(33),
          → umur VARCHAR(9),
          → PRIMARY KEY (kode_supir));
Query OK, 0 rows affected
Time: 0.067s
```

3. Tabel Armada

```
『Aspian』 ❯ CREATE TABLE armada(
          → plat VARCHAR(11) PRIMARY KEY,
          → jenis VARCHAR(21), warna VARCHAR(21),
          → kursi VARCHAR(2), kode_supir VARCHAR(7),
          → FOREIGN KEY (kode_supir) REFERENCES
          → supir (kode_supir)
          → ON DELETE CASCADE ON UPDATE CASCADE);
Query OK, 0 rows affected
Time: 0.083s
```

4. Tabel Kursi

```
『Aspian』 ❯ CREATE TABLE kursi(
          → kode_kursi VARCHAR(11) PRIMARY KEY,
          → nomor VARCHAR(21), plat VARCHAR(11),
          → FOREIGN KEY (plat) REFERENCES
          → armada (plat)
          → ON UPDATE CASCADE
          → ON DELETE CASCADE);
Query OK, 0 rows affected
Time: 0.098s
```

5. Tabel Penumpang

```
『Aspian』 ❯ CREATE TABLE penumpang(
          → kode_penumpang VARCHAR(11) PRIMARY KEY,
          → nama VARCHAR(75), alamat TEXT, jk VARCHAR(15),
          → umur VARCHAR(9), telepon VARCHAR(21),
          → kode_tujuan VARCHAR(11), kode_kursi VARCHAR(11),
          → plat VARCHAR(11), tanggal DATE,
          → FOREIGN KEY (kode_tujuan) REFERENCES tujuan (kode_tujuan)
          → ON DELETE CASCADE ON UPDATE CASCADE,
          → FOREIGN KEY (kode_kursi) REFERENCES kursi (kode_kursi)
          → ON DELETE CASCADE ON UPDATE CASCADE,
          → FOREIGN KEY (plat) REFERENCES armada (plat)
          → ON DELETE CASCADE ON UPDATE CASCADE);
Query OK, 0 rows affected
Time: 0.097s
```

Jika *Foreign Key* belum ditambahkan saat pembuatan tabel, *Foreign Key* masih dapat ditambahkan melalui modifikasi tabel yang sudah ada.

```
『Aspian』 ❯ CREATE TABLE kur_si(
          → kode_kursi VARCHAR(11) PRIMARY KEY,
          → nomor VARCHAR(21), plat VARCHAR(11));
Query OK, 0 rows affected
Time: 0.057s
『Aspian』 ❯ ALTER TABLE kur_si ADD FOREIGN KEY
          → (plat) REFERENCES armada (plat)
          → ON DELETE CASCADE ON UPDATE CASCADE;
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
Query OK, 0 rows affected
Time: 0.188s
```

3. Pembahasan

Fokus utama dari tugas ini adalah **Relasi Tabel**. Secara singkat, relasi tabel dalam database relasional merujuk pada hubungan antara dua atau lebih tabel yang dihubungkan melalui kolom-kolom tertentu, yang biasanya menggunakan *Foreign Key*. *Foreign Key* adalah sebuah kolom atau kumpulan kolom dalam sebuah tabel yang digunakan untuk menciptakan hubungan antar tabel dalam sebuah database. Terdapat beberapa jenis relasi, yaitu One-to-One, One-to-Many, dan Many-to-Many.

Dalam pengelolaan relasi tabel, *Referential Actions* memiliki fungsi krusial yang mendukung integritas data saat melakukan operasi *update* atau *delete. Referential Actions* adalah tindakan yang diambil ketika operasi *update a*tau *delete d*ilakukan pada tabel induk yang memiliki hubungan *Foreign Key* dengan tabel anak. *Referential Actions* sering digunakan bersamaan dengan *Foreign Key* untuk memaksimalkan fungsionalitasnya. Beberapa contoh *Referential Action* adalah:

- *CASCADE*: Jika baris dalam tabel induk dihapus atau diperbarui, maka semua baris terkait dalam tabel anak juga akan dihapus atau diperbarui secara otomatis.
- *SET NULL*: Jika baris di tabel induk dihapus atau diperbarui, kolom *Foreign Key* yang terkait di tabel anak akan diatur ke *NULL*.

Selain *CASCADE* dan *SET NULL*, ada juga *Referential Actions* lainnya, seperti RESTRICT, NO ACTION dan *SET DEFAULT*. Informasi lebih detail dapat dilihat pada dokumentasi resminya.

Pada desain database yang diberikan, terdapat ketidaksesuaian panjang data pada kolom yang dihubungkan melalui *Foreign Key*. Hal ini dapat menyebabkan inkonsistensi data dan potensi kesalahan di masa mendatang. Oleh karena itu, langkah yang saya ambil adalah menyesuaikan panjang tipe data pada kolom di tabel anak dengan tabel induk. Kolom yang bermasalah antara lain *kode_kursi* dan *kode_supir*.

4. Pengujian

Pengujian dilakukan dengan memasukkan data ke dalam tabel dan melakukan *update* dan *delete* pada tabel induk. Tujuan dari pengujian ini adalah untuk memastikan bahwa *Foreign Key* dan *Referential Actions* bekerja dengan benar.

```
『Aspian』 ❯ INSERT INTO tujuan VALUES
          → ("tu001", "Siring","Rp10.000", "08:10"),
          → ("tu002", "Pulau Kembang", "Rp30.000", "09:05"),
          → ("tu003", "Bati Bati", "Rp50.000", "10:00"),
          → ("tu004", "Sungai Andai", "Rp25.000", "07:90");
Query OK, 4 rows affected
Time: 0.013s
```

```
『Aspian』 ❯ INSERT INTO supir VALUES
          → ("su001", "Johan", "Pria", "JL. Simpang Siur", "+62aaaaaaaaaaa", "39"),
          → ("su002", "Aditya Surya", "Wanita", "JL. Simpang Ampat", "", "25");
Query OK, 2 rows affected
Time: 0.019s
```

```
『Aspian』 ❯ INSERT INTO armada VALUES
          → ("p001", "Kapal Karam", "Hijau", "99", "su001"),
          → ("p002", "Kapal Selam", "Pink", "50", "su002");
Query OK, 2 rows affected
Time: 0.027s
```

```
『Aspian』 ❯ INSERT INTO kursi VALUES
          → ("k001", "01", "p001"), ("k002", "02", "p001"),
          → ("k003", "01", "p002"), ("k004", "02", "p002");
Query OK, 4 rows affected
Time: 0.018s
```

```
『Aspian』 ❯ INSERT INTO penumpang VALUES
          → ("c001", "Reno Halimawan", "Bati Bati", "Pria", "20",
          → "+62bbbbbbbbbbb", "tu001", "k004", "p002", "2024-09-29"),
          → ("c002", "Ali Baba", "Sungai Andai", "Pria", "19",
          → "+62ccccccccccc", "tu002", "k004", "p002", "2025-11-21"),
          → ("c003", "Jefri Nichol", "Alalak", "Pria", "19",
          → "+62ddddddddddd", "tu003", "k001", "p001", "2024-09-30"),
          → ("c004", "Fulana", "Kelayan C", "Pria", "16",
          → "+62eeeeeeeeeee", "tu004", "k003", "p002", "2045-01-01");
Query OK, 4 rows affected
Time: 0.013s
```

```
『Aspian』 ❯ UPDATE tujuan set kode_tujuan = "tu010" WHERE kode_tujuan = "tu001";
Query OK, 1 row affected
Time: 0.013s
『Aspian』 ❯ SELECT kode_penumpang, nama, kode_tujuan FROM penumpang;
+----------------+----------------+-------------+
| kode_penumpang | nama           | kode_tujuan |
+----------------+----------------+-------------+
| c001           | Reno Halimawan | tu010       |
| c002           | Ali Baba       | tu002       |
| c003           | Jefri Nichol   | tu003       |
| c004           | Fulana         | tu004       |
+----------------+----------------+-------------+
```

```
『Aspian』 ❯ DELETE FROM tujuan WHERE kode_tujuan = "tu003";
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
Query OK, 1 row affected
Time: 0.009s
『Aspian』 ❯ SELECT kode_penumpang, nama, kode_tujuan FROM penumpang;
+----------------+----------------+-------------+
| kode_penumpang | nama           | kode_tujuan |
+----------------+----------------+-------------+
| c001           | Reno Halimawan | tu010       |
| c002           | Ali Baba       | tu002       |
| c004           | Fulana         | tu004       |
+----------------+----------------+-------------+

3 rows in set
Time: 0.011s
```

Dalam gambar-gambar tersebut, dapat dilihat bahwa ketika saya mengubah *kode_tujuan* dari *tu001* menjadi *tu010*, data pada tabel penumpang juga ikut berubah, Hal ini membuktikan bahwa *Referential Actions* berfungsi dengan baik. Hal yang sama juga berlaku untuk operasi *delete*. Ketika saya menghapus tujuan dengan kode *tu003*, semua penumpang yang memiliki tujuan *tu003* akan turut dihapus.

5. Referensi

- https://chatgpt.com
- https://www.w3schools.com/sql/
- https://stackoverflow.com/questions/13444859/sql-on-delete-cascade-which-way-does-the-deletion-occur
- https://stackoverflow.com/questions/10028214/add-foreign-key-to-existing-table
- https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html
- https://www.youtube.com/watch?v=6xU8K_w7gN4
- https://stackoverflow.com/questions/6889065/inserting-multiple-rows-in-mysql
- https://medium.com/@fahmisyaifudin35/relasi-tabel-database-one-to-one-one-to-many-many-to-many-44010f703f57