PRAKTIKUM SISTEM BASIS DATA

Modul 3: Data Definition Language Foreign Key & Referential Integrity

Deskripsi:

Modul ini berisi materi untuk melatih keterampilan mahasiswa dalam menggunakan perintah Structured Query Language (SQL) untuk Data Definition Language (DDL) dengan memberikan pemahaman lebih lanjut mengenai *referential integrity.*

Tujuan:

Tujuan pembelajaran yang akan dicapai pada modul ini adalah mahasiswa mampu membuat database dan tabel lengkap dengan atribut, konstrain dan relationship menggunakan SQL DDL MySQL.

Materi dan Pembagian waktu:

Materi dan pembagian waktu pelaksanaan perkuliahan meliputi :

No.	Materi	Waktu
1	Pre-test Materi SQL DDL Dasar (online)	5 menit
2	Materi DDL Tabel Berelasi & Referential Integrity	55 menit
3	Latihan SQL DDL dengan Referential Integrity	40 menit
4	Post-test	70 menit
	Total Waktu	170 menit

Penilaian:

Akan dilakukan penilaian berdasarkan materi modul ini pada pertemuan berikutnya dengan bentuk pre-test dan post-test berbobot 6 poin.

Capaian Kompetensi Mahasiswa:

Kompetensi yang akan dicapai oleh mahasiswa pada modul ini meliputi :

1. Mahasiswa dapat membuat, mengubah, dan menghapus database dan tabel menggunakan perintah SQL DDL
2. Mahasiswa dapat menjelaskan dampak pengaturan *referential integrity* pada hubungan antara tabel induk dengan tabel anak.

Alat dan Bahan:

1. Modul Praktikum
2. Komputer
3. Viewer LCD

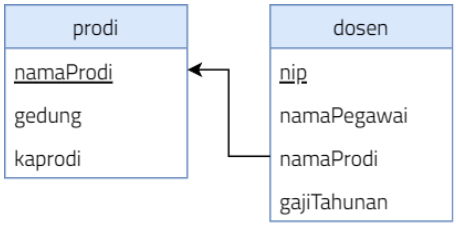
Materi 1: Referential Integrity pada Database

Waktu: 55 menit, **Metode Pembelajaran:** Penyampaian Materi

Pada pertemuan sebelumnya, kita sudah mempelajari bagaimana membuat suatu tabel. Berikutnya apabila kita ingin merelasikan satu tabel dengan tabel lain, kita membutuhkan konstrain Forein Key (FK). Berikut adalah klausa konstrain FK pada statemen CREATE TABLE di MySQL:

```
CONSTRAINT nama_konstrain
FOREIGN KEY (nama_kolom_fk)
REFERENCES tabel_induk(nama_kolom_pk)
[ON DELETE opsi_referensi]
[ON UPDATE opsi_referensi]
```

Sebagai contoh, perhatikan relationship/hubungan antar tabel *prodi* dengan *dosen* pada schema diagram Gambar 1.



Gambar 1. Hubungan antar relasi *prodi* dengan *dosen*

Gambar 1 menunjukkan bahwa atribut *namaProdi* pada tabel *dosen* mengacu pada atribut *namaProdi* di tabel *prodi*. Hal ini memenuhi pernyataan bahwa nilai dari atribut *namaProdi* untuk setiap record di tabel *dosen* harus ada di atribut *namaProdi* di tabel *prodi.*

Kita mengingat kembali istilah Referential Integrity (RI) yang merupakan sekumpulan konstrain yang diterapkan pada FK yang mencegah terjadinya kehilangan referensi akibat perubahan (*insert/update/delete*) baris yang digunakan sebagai acuan. Seperti dicontohkan pada materi di kelas teori, terkait dengan pertanyaan-pertanyaan berikut:

- **Bagaimana jika di tabel *prodi,* nama prodi diubah ke "Teknik Informatika"?** Dalam hal ini sudah ada beberapa record di tabel *dosen* yang nama prodi-nya adalah "Informatika".
- **Bagaimana jika di tabel *prodi,* baris yang berisi "Biologi" dihapus?** Demikian pula dalam hal ini sudah ada beberapa record di tabel *dosen* yang berasal dari prodi Biologi.

Secara implementasi, RI mencegah pengisian (*insert*) baris pada tabel anak yang ber-FK ketika kita tidak memiliki baris yang digunakan sebagai acuan di tabel induk. RI juga digunakan untuk menentukan apa yang akan terjadi jika baris pada tabel induk diubah (*update*) atau dihapus (*delete*), sementara pada saat yang sama di tabel anak ada baris yang mengacu pada baris yang dihapus tersebut. (mysqltutorial.org, n.d.)

Opsi referensi yang dimaksud adalah:

- CASCADE: jika suatu baris dari tabel induk dihapus atau diupdate, baris yang berada di tabel anak turut dihapus atau diupdate.
- SET NULL: jika suatu baris dari tabel induk dihapus atau diupdate, nilai atribut FK di tabel anak akan dikosongkan.

- RESTRICT: jika suatu baris di tabel induk diacu oleh satu atau banyak baris di tabel anak, MySQL akan menolak penghapusan atau pengupdatean baris ini.

Jika kita tidak memilih salah satu, maka nilai default dari MySQL adalah RESTRICT.

Dalam mendefinisikan atau membuat tabel yang berelasi seperti di atas, urutan pembuatannya tidak bisa dibolak-balik. Kita harus mendefinisikan **tabel induk** dahulu—yaitu tabel yang berisi PK—baru kemudian mendefinisikan **tabel anak** atau tabel yang berisi FK yang mengacu ke PK tadi. Berikut SQL DDL untuk mendefinisikan tabel *prodi*:

```
CREATE DATABASE [namaAnda]_p3;
USE [namaAnda]_p3;
CREATE TABLE prodi (
   namaProdi VARCHAR(100) PRIMARY KEY,
   gedung VARCHAR(100) NOT NULL,
   kaprodi VARCHAR(200) NOT NULL
);
```

Setelah tabel dengan PK yang nantinya akan diacu sudah terdefinisikan, kita lanjutkan dengan pendefinisian tabel *dosen*:

```
CREATE TABLE dosen (
   nip VARCHAR(8) PRIMARY KEY,
   namaPegawai VARCHAR(255) NOT NULL,
   namaProdi VARCHAR(100),
   gajiTahunan INT NOT NULL,
   CONSTRAINT namaProdiFK
   FOREIGN KEY (namaProdi)
   REFERENCES prodi(namaProdi)
);
```

Jika diperhatikan pada hasil eksekusi SQL di atas, hasilnya akan menjadi seperti Gambar 2. Di situ tertera Key = MUL untuk atribut *namaProdi* di tabel *dosen* (atau biasa dituliskan *dosen.namaProdi*).

```
MariaDB [lukas_p3]> desc prodi;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| namaProdi | varchar(100) | NO   | PRI | NULL    |       |
| gedung    | varchar(100) | NO   |     | NULL    |       |
| kaprodi   | varchar(200) | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.007 sec)

MariaDB [lukas_p3]> desc dosen;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| nip         | varchar(8)   | NO   | PRI | NULL    |       |
| namaPegawai | varchar(255) | NO   |     | NULL    |       |
| namaProdi   | varchar(100) | YES  | MUL | NULL    |       |
| gajiTahunan | int(11)      | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.005 sec)

MariaDB [lukas_p3]>
```

Gambar 2. Hasil SQL DDL

Keterangan dari key ini menurut (DelftStack, 2021) adalah:

- PRI: merupakan primary key. Otomatis tidak memungkinkan nilai NULL. MySQL mengizinkan beberapa kolom ditetapkan sebagai PRI.
- UNI: merepresentasikan konstrain UNIQUE, yang akan memaksakan keunikan dari suatu baris/record dibandingkan baris lainnya, namun mengizinkan nilai NULL. MySQL mengizinkan beberapa kolom ditetapkan sebagai UNI.
- MUL: merupakan kolom indeks yang tidak menunjukkan sifat dari key PRI maupun UNI. Key ini mengizinkan nilai NULL dan pengisian nilai yang kembar di beberapa baris/record yang berbeda.

Apabila ingin menambahkan opsi referensi saat **CREATE TABLE**, maka dapat dituliskan pada tabel **anak**. Sebagai contoh, di bawah ini ditambahkan opsi referensi CASCADE saat UPDATE dan DELETE:

```
CREATE TABLE mahasiswa (
   nim VARCHAR(8) PRIMARY KEY,
   namaMahasiswa VARCHAR(255) NOT NULL,
   namaProdi VARCHAR(100),
   CONSTRAINT namaProdiFK
   FOREIGN KEY (namaProdi)
   REFERENCES prodi(namaProdi)
   ON DELETE CASCADE
   ON UPDATE CASCADE
);
```

Apabila tabel sudah terlanjur dibuat tapi belum ada opsi referensi, maka harus dilakukan penghapusan FK terlebih dahulu dan melakukan ALTER TABLE menggunakan acuan berikut ini:

```
ALTER TABLE nama_tabel
DROP tipe_konstrain nama_konstrain;

ALTER TABLE nama_tabel
ADD CONSTRAINT nama_konstrain FOREIGN KEY (nama_kolom_fk) REFERENCES
tabel_induk(nama_kolom_pk)
[ON DELETE opsi_referensi]
[ON UPDATE opsi_referensi];
```

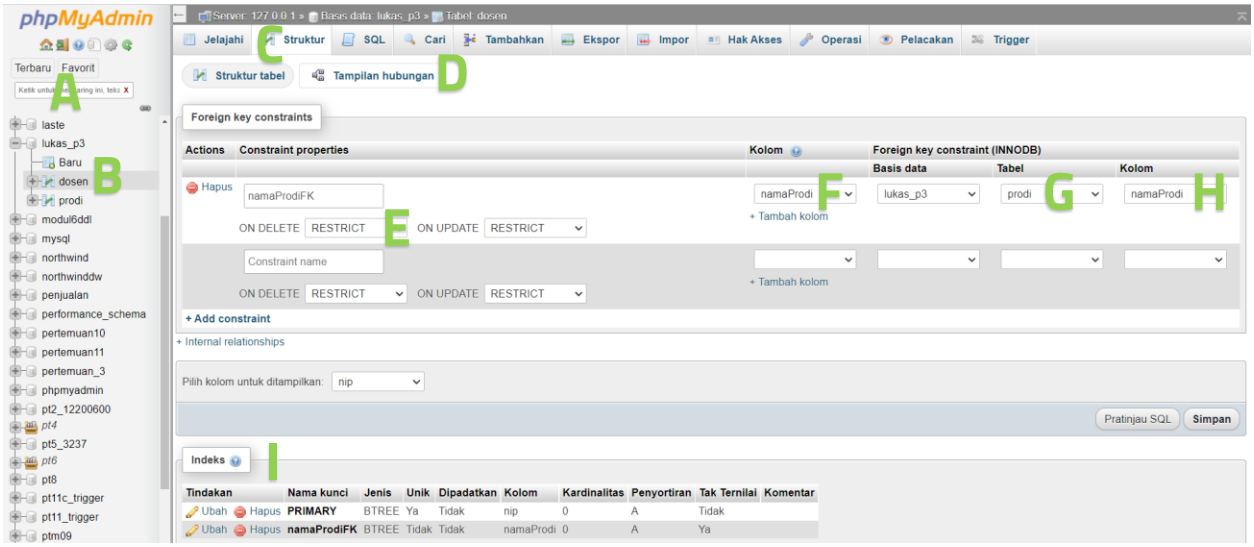
Penetapan konstrain PK-FK ini membantu pendisiplinan pengacuan data. Namun kadang kala kita merasa kesulitan ketika konstrain FK yang tercipta tidak terdokumentasi dengan baik. Kita tidak dapat langsung menghapus tabel induk apabila dia memiliki hubungan ke tabel anak. Demikian juga jika kita hendak mengimpor konten tabel dari sebuah file CSV (comma-separated values) untuk tabel anak misalnya. Jika di tabel induk tidak ada PK yang bisa diacu, maka akan muncul *ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails.* Kita dapat mematikan pengecekan FK untuk sementara dengan statemen:

```
SET foreign_key_checks = 0;
```

Jangan lupa untuk mengembalikannya dengan mengatur nilainya ke 1.

Kita juga dapat melihat konstrain FK secara visual melalui phpMyAdmin. Caranya adalah dengan:

1. Buka database dengan mengklik nama database di kolom kiri.
2. Pilih tabel anak.
3. Pada menu atas, pilih tab Struktur / Structure.
4. Klik button Tampilan hubungan / Show relationships.



Gambar 3. Tampilan Foreign key constraints

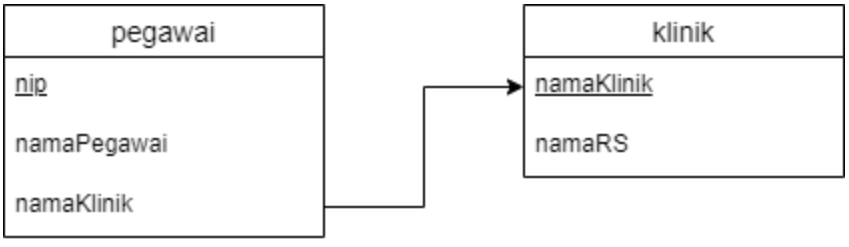
Pada Gambar 3 terlihat kolom database (A), pilihan tabel (B), tab Struktur (C), button Tampilan hubungan (D), jenis referential integrity (E), kolom dari tabel anak yang ditetapkan sebagai FK (F), tabel induk (G), PK dari tabel induk yang digunakan sebagai acuan (H). Karena FK adalah kunci, maka setiap kunci di MySQL diperlakukan sebagai indeks (H).

Materi 2: Latihan DDL Foreign Key & Referential Integrity pada Database

Waktu: 45 menit, **Metode Pembelajaran:** Praktikum *Guided* menggunakan console.

Catatan: Pengerjaan menggunakan console dengan **`SET foreign_key_checks = 1;`**

Buat tabel berikut ini dengan SQL berikut ini:



```sql
CREATE TABLE klinik (
   namaKlinik VARCHAR(100) PRIMARY KEY,
   namaRS VARCHAR(100)
);

CREATE TABLE pegawai (
   nip CHAR(5) PRIMARY KEY,
   namaPegawai VARCHAR(255) NOT NULL,
   namaKlinik VARCHAR(100),
   CONSTRAINT namaKlinikFK FOREIGN KEY (namaKlinik) REFERENCES klinik(namaKlinik)
);
```

1. Dari tabel yang sudah terdefinisikan, tambahkan konstrain RI untuk UPDATE dan DELETE dengan opsi referensi CASCADE.

2. Masukkan sebuah baris pada tabel *pegawai*:

nip	namaPegawai	namaklinik
12211	Endah	Sehati

   Tuliskan SQL INSERT yang tepat. Apa maksud dari error yang keluar? Solusinya bagaimana?

3. Lakukan INSERT pada kedua data ini di tabel *klinik*.

namaKlinik	namaRS
Sehati	RS Koinonia
Anna	RS Agape

   Lakukan insert kembali pada nomor (2). Apa yang terjadi kali ini?

4. Apabila ingin menambahkan kedua data ini, apa yang harus dilakukan terlebih dahulu?

nip	namaPegawai	namaklinik
12255	Petra	Anna
12289	Laksmi	Idaman

5. Lakukan perubahan data pada klinik Sehati menjadi Sehati Sejiwa pada tabel *klinik.*

```sql
UPDATE klinik
SET namaKlinik = 'Sehati Sejiwa'
WHERE namaKlinik = 'Sehati';
```

   Data apa saja dan dari tabel mana saja yang berubah?

6. Lakukan penghapusan data nama prodi Informatika di tabel *prodi.*

```sql
DELETE FROM klinik
WHERE namaKlinik = 'Sehati Sejiwa';
```

   Data apa saja dan dari tabel mana saja yang berubah?

7. Lakukan ALTER TABLE kembali untuk mengubah konstrain RI update maupun delete menjadi SET NULL.

8. Hapus data klinik Anna pada tabel *klinik*. Apa yang terjadi?

9. Ubah data klinik Idaman menjadi Idaman Jiwa pada tabel *klinik*. Apa yang terjadi?

10. Kembalikan pengaturan RI menjadi RESTRICT. Lakukan salah satu query pengujian untuk menunjukkan pelarangan update data terkait dengan RI ini.

TIPS: *Ekspor hasil pekerjaan Anda menjadi file .SQL agar bisa dilanjutkan lagi di rumah atau pada kesempatan lain.*
Cara ekspor:

```
mysqldump -u NAMAUSER NAMADATABASE > namafilexport.sql
```

Cara impor:

```
mysql -u NAMAUSER NAMADATABASE < namafilexport.sql
```

Pastikan sebelum impor, database sudah dibuat terlebih dahulu.

Referensi

DelftStack. (2021). *MUL vs PI vs UNI in MySQL*. https://www.delftstack.com/howto/mysql/mul-vs-pri-vs-uni-in-mysql/

mysqltutorial.org. (n.d.). *MySQL Foreign Key*. https://www.mysqltutorial.org/mysql-foreign-key/