Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-11: MySQL – Data Manipulation Language (DML)

Mata Kuliah Basis Data

Pengampu: Tim Ajar Basis Data

2021

Topik

Data Manipulation Language (DML) pada DBMS MySQL

Tujuan

Mahasiswa diharapkan dapat:

1. Memahami penggunaan SQL statement INSERT.
2. Memahami penggunaan SQL statement UPDATE.
3. Memahami penggunaan SQL statement DELETE.

Pendahuluan

DML merupakan istilah untuk beberapa sintaksis (syntax) dari SQL yang digunakan untuk melakukan perubahan pada data (isi tabel-tabel) dalam suatu database. DML terdiri dari 3 klausa utama yaitu:

1. **INSERT** : Menambah baris baru pada sebuah tabel
2. **UPDATE** : Mengubah nilai suatu baris pada sebuah tabel.
3. **DELETE** : Menghapus suatu baris dari sebuah tabel.

Pada DML terdapat dua jenis bahasa, yaitu :

1. **High-Level(Non_procedural) DML.**
   - Digunakan secara interaktif (interpreter)
   - Dapat dijadikan satu dengan general purpose programming language (embedded)

   High-Level DML yang biasa digunakan secara interaktif disebut "Query Language".
2. **Low-Level(Proedural) DML.**
   Digunakan secara embedded dalam suatu general purpose programming language

Bilamana kedua jenis DML diatas digunakan secara "embedded", maka : bahasa pemrograman yang digunakan disebut sebagai "Host Language" dan "DML-nya disebut "Sub Language"

Operasi INSERT

Operasi insert bertujuan untuk menyisipkan satu tuple baru ke dalam suatu relasi R.

Klausa pembentuk:

1. **INSERT**
2. **INTO**
3. **VALUES**

Format:

1. **INSERT INTO** nama_tabel (kolom1, kolom2, ...dst.) **VALUES** (nilai_kolom1, nilai_kolom2, ...dst.);
2. **INSERT INTO** nama_tabel **VALUES** (nilai_kolom1, nilai_kolom2, ...dst.);
3. [Salah satu dari kedua format sebelumnya]**,** (nilai_kolom_kolom_baris1)**,** (nilai_kolom_kolom_baris2)**,** ...dst.

Operasi ini memungkinkan untuk melanggar empat jenis constraint sebagaimana dijelaskan berikut ini :

1. **DOMAIN Constraint** dapat dilanggar jika suatu nilai attribute yang diberikan tidak ada dalam domain yang berkorespondensi dengan attribute tadi.
2. **KEY Constraint** dapat dilanggar jika nilai key dalam tuple baru t sudah ada dalam tuple lain dalam relasi r(R).
3. **ENTITY INTEGRITY** Constraint dapat dilanggar jika primary key dari tuple baru t adalah NULL
4. **REFERENTIAL INTEGRITY** Constraint dapat dilanggar jika nilai dari suatu foreign key dalam t mengacu ke suatu tuple yang tidak ada dalam relasi yang diacu.

Ada dua pilihan tindakan yang dapat dilakukan jika ada satu atau lebih constraint yang dilanggar akibat operasi insert, yaitu :

1. Menolak (reject) operasi insertion. Biasanya DBMS memberikan penjelasan mengapa proses insertion ditolak.
2. Berusaha memperbaiki alasan penolakan proses insertion. Dimana insertion akan diterima jika user melakukan perubahan nilai-nilai attribute sehingga insertion diterima.

Operasi DELETE

Operasi delete bertujuan untuk menghapus satu atau beberapa tuple di dalam suatu relasi R. Operasi ini hanya dapat melanggar **referential integrity**, jika tuple yang dihapus diacu oleh kunci-kunci tamu dari tuple yang lain dalam basis data.

Klausa pembentuk:

1. **DELETE**
2. **FROM**
3. **WHERE** [opsional]

Format:

1. **DELETE FROM** nama_tabel **WHERE** nama_kolom_patokan **[operator_perbandingan]** nilai_patokan;
2. **DELETE * FROM** nama_tabel**;** atau **DELETE FROM** nama_tabel**;**

*Operator perbandingan/comparison operator dapat berupa: **=, <, >, <=, >=, <>**

Ada empat pilihan tindakan yang dapat dilakukan jika suatu deletion melanggar constraint yang telah ditentukan, yaitu :

1. Menolak(reject) proses deletion.
2. Berusaha untuk melakukan “cascade deletion”, yaitu dengan menghapus sejumlah tuple yang mengacu pada tuple yang akan dihapus.
3. Melakukan modifikasi nilai attribute yang mengacu pada tuple yang dihapus, yaitu setiap nilai diset NULL atau diganti dengan nilai dari tuple lain yang valid sebagai acuan baru. Akan tetapi, bila attribute yang mengacu yang menyebabkan pelanggaran adalah bagian dari primary key, maka ia tidak dapat diset NULL (karena melanggar entity integrity).
4. Kombinasi 2 dan 3.

Operasi UPDATE

Operasi update digunakan untuk merubah nilai-nilai satu atau lebih attribute dalam satu atau lebih tuple dalam sejumlah relasi R.

Klausa pembentuk:

1. **UPDATE**

2. **SET**
3. **WHERE** [opsional]

Format:

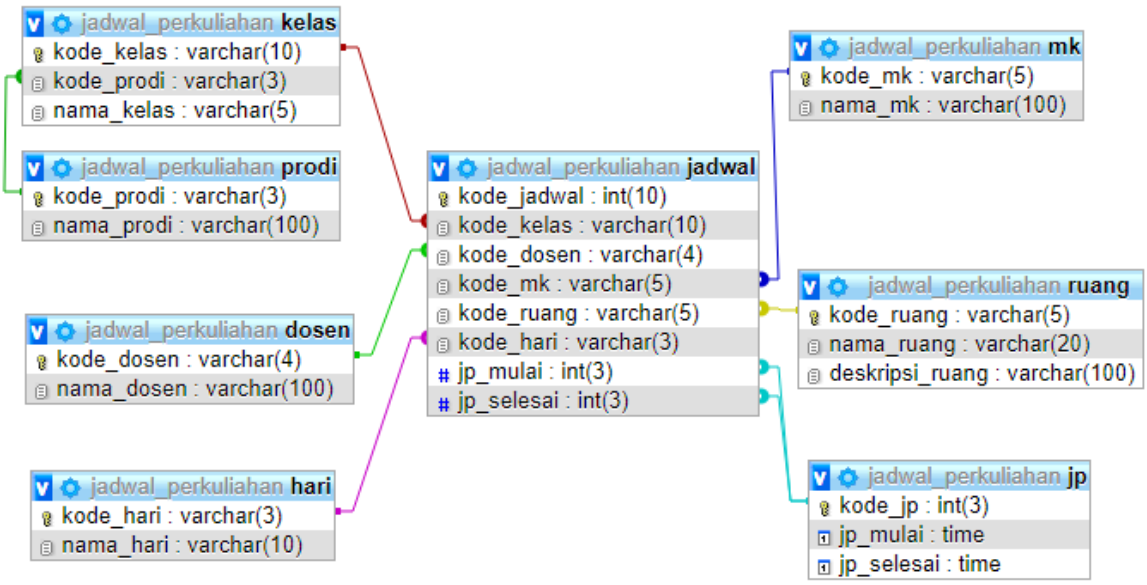
1. **UPDATE** nama_tabel **SET** nama_kolom **=** nilai_baru **WHERE** nama_kolom_patokan **[operator_perbandingan]** nilai_patokan;
2. **UPDATE** nama_tabel **SET** nama_kolom**1 =** nilai_baru**1**, nama_kolom**2 =** nilai_baru**2**, ...dst. **WHERE** nama_kolom_patokan **[operator_perbandingan]** nilai_patokan;

*Operator perbandingan/comparison operator dapat berupa: **=, <, >, <=, >=, <>**

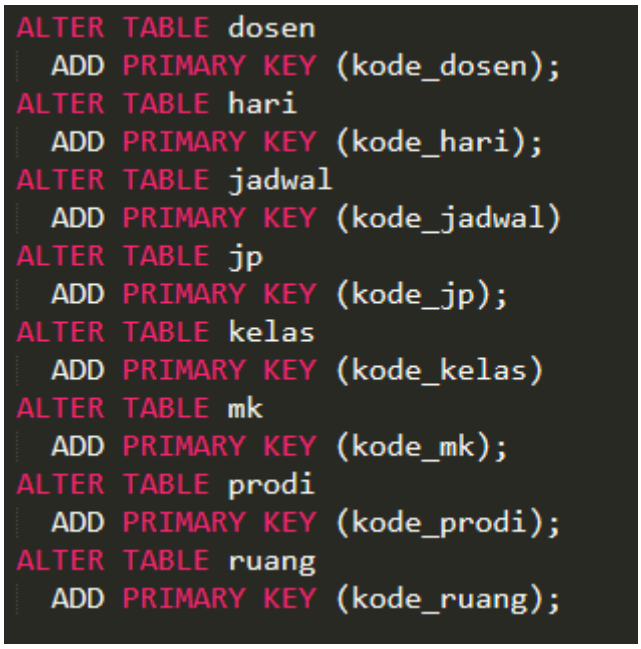
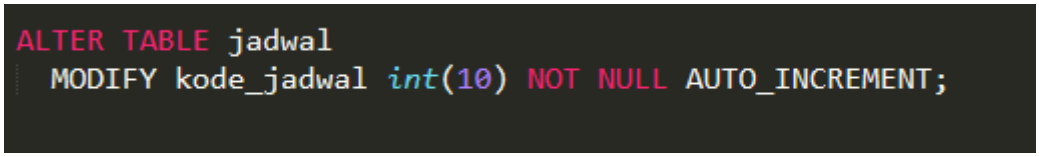
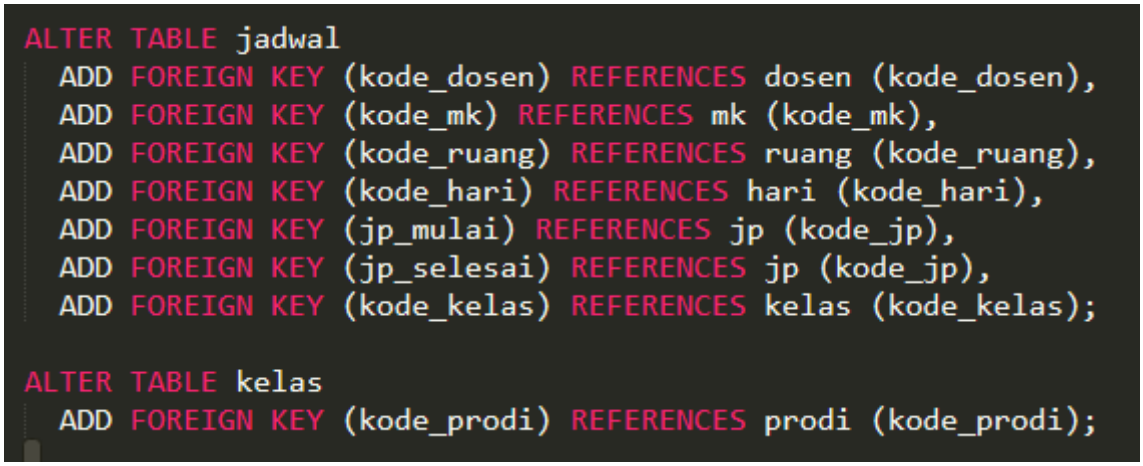
Operasi UPDATE bisa dilakukan pada tiga jenis attibut, dengan permasalahan yang berbeda sebagaimana berikut ini :

1. Modifikasi nilai suatu foreign key, maka DBMS harus melakukan pengecekan bahwa nilai-nilai baru yang diberikan mengacu pada tuple yang ada dalam relasi-relasi yang dijadikan acuan.
2. modifikasi nilai suatu primary key serupa dengan proses deletion satu tuple dan inserting yang lain pada tempat yang sama. Akibatnya, pilihan- pilihan seperti yang dilakukan pada operasi INSERT dan DELETE dapat dipakai agar modifikasi tidak melanggar constraint.
3. modifikasi suatu attribute yang bukan primary key atau bukan foreign key biasanya tidak akan menimbulkan masalah. DBMS hanya perlu untuk mengecek apakah nilai-nilai baru yang diberikan mempunyai tipe data dan domain yang valid.

Praktikum – Bagian 1: Membuat Database untuk Percobaan

Langkah	Keterangan
1	Perhatikan skema/model relasional/EER diagram dari database berikut.
	
2	Skema tersebut adalah sekema database pada sebuah sistem informasi penjadwalan di Jurusan Teknologi Informasi. Pertama-tama, **buatlah database tersebut dengan mengeksekusi baris-baris kode DDL berikut dan <u>jelaskan maksud dari setiap tahapan tersebut</u>.**
2a	`DROP DATABASE IF EXISTS jadwal_perkuliahan;` `CREATE DATABSE jadwal_perkuliahan;` `USE jadwal_perkuliahan;`
2b	`CREATE TABLE dosen (` `  kode_dosen varchar(4) NOT NULL,` `  nama_dosen varchar(100) DEFAULT NULL` `);`
2c	`CREATE TABLE hari (` `  kode_hari varchar(3) NOT NULL,` `  nama_hari varchar(10) DEFAULT NULL` `) ;`

2d	```sql CREATE TABLE jadwal (kode_jadwal int(10) NOT NULL, kode_kelas varchar(10) DEFAULT NULL, kode_dosen varchar(4) DEFAULT NULL, kode_mk varchar(5) DEFAULT NULL, kode_ruang varchar(5) DEFAULT NULL, kode_hari varchar(3) DEFAULT NULL, jp_mulai int(3) DEFAULT NULL, jp_selesai int(3) DEFAULT NULL) ; ```
2e	```sql CREATE TABLE jp (kode_jp int(3) NOT NULL, jp_mulai time DEFAULT NULL, jp_selesai time DEFAULT NULL) ; ```
2f	```sql CREATE TABLE kelas (kode_kelas varchar(10) NOT NULL, kode_prodi varchar(3) DEFAULT NULL, nama_kelas varchar(5) DEFAULT NULL) ; ```
2g	```sql CREATE TABLE mk (kode_mk varchar(5) NOT NULL, nama_mk varchar(100) DEFAULT NULL) ; ```
2h	```sql CREATE TABLE prodi (kode_prodi varchar(3) NOT NULL, nama_prodi varchar(100) DEFAULT NULL) ; ```
2i	```sql CREATE TABLE ruang (kode_ruang varchar(5) NOT NULL, nama_ruang varchar(20) DEFAULT NULL, deskripsi_ruang varchar(100) DEFAULT NULL) ; ```

2j	
2k	
2l	
3	Cek database Anda dengan perintah '**SHOW TABLES**' untuk memastikan bahwa semua tabel sudah dibuat. Setelah selesai membuat database diatas, lanjutkan ke **Praktikum – Bagian 2**.

Praktikum – Bagian 2: Percobaan Statement INSERT

Langkah	Keterangan
1	Untuk menambahkan data (mengisi) suatu tabel, digunakan statement (pernyataan) **INSERT**. Eksekusi SQL berikut untuk menambahkan 1 baris (record) baru pada tabel **mk**.
	`INSERT INTO mk (kode_mk, nama_mk) VALUES ('02010', 'Basis Data');`
	Statement SQL tersebut menambahkan 1 baris baru ke tabel **mk** pada kolom yang dinyatakan di dalam tanda kurung () pertama. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut. Pembahasan lebih lengkap mengenai SELECT dijadwalkan untuk disampaikan pada pertemuan berikutnya, namun secara umum, statement SELECT digunakan untuk menyajikan record-record yang ada pada suatu tabel. Karakter * akan menampilkan isi dari semua kolom yang ada pada tabel.
	`SELECT * FROM mk`
2	Apabila data di-insert-kan pada semua kolom tabel, maka kita dapat langsung menggunakan klausa VALUES tanpa harus menuliskan nama-nama kolom dahulu.
	`INSERT INTO mk VALUES('02041', 'Teknologi Data');`
	Statement SQL tersebut menambahkan 1 baris baru ke tabel **mk** pada kolom yang ada pada struktur tabel **mk** tanpa menyebutkan nama kolomnya. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut.
	`SELECT * FROM mk`
3	Untuk menambahkan beberapa kolom sekaligus dalam 1 statement digunakan statement dengan format seperti berikut.
	`INSERT INTO mk VALUES` `('02004', 'Aljabar Linier'),` `('02005', 'Analisis Dan Desan Berorientasi Objek'),` `('02006', 'Bahasa Indonesia');`
	Statement SQL tersebut menambahkan 3 baris baru ke tabel **mk** pada kolom yang ada pada struktur tabel **mk** tanpa menyebutkan nama kolomnya. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut.
	`SELECT * FROM mk`
4	Dan seperti berikut, jika hanya kolom tertentu saja yang akan diberi nilai dengan cara menyebutkan nama kolomnya.

	``sql INSERT INTO mk (kode_mk, nama_mk) VALUES ('02001', 'Agama'), ('02002', 'Alajabar Linier'), ('02003', 'Algoritma dan Struktur Data'); ```
	Statement SQL tersebut menambahkan 3 baris baru ke tabel **mk** pada kolom yang ada pada struktur tabel **mk**. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut.
	``sql SELECT * FROM mk ```
5	Statement INSERT juga dapat dieksekusi dengan menggunakan klausa SET alih-alih VALUES.
	``sql INSERT INTO mk SET kode_mk = '02011', nama_mk = 'Desain Pemrograman Web'; ```
	Statement SQL tersebut menambahkan 1 baris baru ke tabel **mk** pada kolom yang ada pada struktur tabel **mk**. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut.
	``sql SELECT * FROM mk ```
6	Pada statement INSERT juga dapat digunakan klausa SELECT. Misalnya kita ingin menyalin semua baris pada tabel **mk** ke tabel **mk_backup**, maka kita SQL berikut dapat digunakan. **(Buat terlebih dahulu tabel "mk_backup" dengan struktur tabel yang sama dengan tabel "mk")**
	``sql INSERT INTO mk_backup SELECT * FROM mk; ```
	Statement SQL tersebut menambahkan data baru dari tabel **mk** ke tabel **mk_backup**. Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement **SELECT** seperti berikut.
	``sql SELECT * FROM mk_backup; ```
7	Setelah berhasil mengeksekusi SQL tersebut, lanjutkan ke **Praktikum - Bagian 3**.

Praktikum - Bagian 3: Percobaan Statement UPDATE

Langkah	Keterangan
1	UPDATE digunakan untuk mengubah nilai suatu baris pada sebuah tabel. Sebelum memulai praktikum bagian 3, Import terlebih dahulu file **isi_data_jadwal_perkuliahan.sql** pada database **jadwal_perkuliahan** yang sudah dibuat pada Bagian 1. Format dasar statement Update ini adalah sebagai berikut:
	`UPDATE jadwal SET kode_dosen='D010'`
	Statement tersebut mengubah nilai SEMUA baris dari tabel **jadwal** pada kolom **kode_dosen** dengan nilai **D010**. Apabila kita tampilkan isi tabel, maka sekarang semua mata kuliah akan diampu oleh dosen dengan kode_dosen tersebut. Tampilkan isi data menggunakan statement SELECT berikut
	`SELECT * FROM jadwal`
2	Untuk mengubah nilai **pada baris tertentu saja**, kita tambahkan klausa **WHERE** pada statement UPDATE. Misalkan kita akan menjadikan dosen dengan kode_dosen D022 sebagai pengampu mata kuliah dengan kode_mk 02010, maka dapat digunakan SQL sebagai berikut:
	`UPDATE jadwal` `  SET kode_dosen='D022'` `  WHERE kode_mk='02010'`
	`SELECT * FROM jadwal WHERE kode_mk='02010'`
4	Klausa WHERE tidak selalu hanya membatasi UPDATE pada 1 baris saja, ia juga bisa memberlakukan UPDATE pada banyak baris sekaligus. Semuanya tergantung pada kondisi yang kita tentukan. Statement berikut ini akan mengosongkan **kode_dosen** untuk semua **mata kuliah** yang diampu oleh dosen dengan **kode_dosen** D010.
	`UPDATE jadwal` `  SET kode_dosen = NULL` `  WHERE kode_dosen = 'D010'`
	`SELECT * FROM jadwal`
5	Untuk mengubah beberapa kolom sekaligus dalam satu kali eksekusi statement UPDATE, dapat digunakan format berikut.
	`UPDATE jadwal` `  SET kode_dosen = 'D012',` `  kode_ruang = '0702'` `  WHERE kode_kelas = '2021020204'`
	`SELECT * FROM jadwal WHERE kode_kelas = '2021020204'`

6	Kita juga dapat menggunakan statement UPDATE dengan SELECT. Misalkan kita ingin mengeset kode_dosen dari kode_mk '02010' dengan kode_dosen dari dosen yang bernama 'Dika Rizky Yunianto SKom., MKom.', maka dapat digunakan SQL dengan format berikut.
	`` UPDATE jadwal SET kode_dosen = (SELECT kode_dosen FROM dosen WHERE nama_dosen='Dika Rizky Yunianto SKom., MKom.') WHERE kode_mk = '02010' ```
	`` SELECT * FROM jadwal WHERE kode_mk='02010' ```
7	Setelah berhasil mengeksekusi SQL tersebut, lanjutkan ke **Praktikum - Bagian 4**.

Praktikum - Bagian 4: Percobaan Statement DELETE

Langkah	Keterangan
1	DELETE digunakan untuk menghapus satu atau lebih baris dari sebuah tabel. Misalkan kita ingin menghapus jadwal yang memiliki nilai pada kolom kode_dosen, maka format dasar statement seperti berikut dapat kita gunakan:
	`DELETE FROM jadwal WHERE kode_dosen IS NOT NULL;`
	`SELECT * FROM jadwal`
2	HATI-HATI apabila kita menggunakan statement DELETE tanpa WHERE! Cobalah eksekusi syntax SQL berikut:
	`DELETE FROM jadwal;`
3	Semua data dalam satu tabel jadwal akan hilang!
	`SELECT * FROM jadwal`
4	Lanjutkan ke bagian **Tugas**!

Tugas

- Import kembali **isi_data_jadwal_perkuliahan.sql**.
- Screenshot sintaks dan hasil SELECT dari setiap soal dibawah ini!

1. Ubah nama mata kuliah “Basis Data” menjadi “Basis Data Dasar”!
2. Ubah semua jadwal kuliah mata kuliah “Basis Data Dasar” menjadi hari Senin di jam pelajaran ke 5 sampai dengan jam pelajaran ke 10!
3. Hapus jadwal perkuliahan “Kewarganegaraan” pada tabel jadwal!
4. Tambahkan mata kuliah “Pancasila”, “Bela Negara”, “Wawasan Nusantara” pada tabel mk!
5. Hapus semua isi data pada tabel mk_backup!
6. Isi data tabel mk_backup dengan isi dari tabel mk!
7. Buatlah tabel mahasiswa dengan atribut nim, nama_mahasiswa, kode_kelas. Dimana kode_kelas mereferensi kepada tabel kelas. Isi tabel tersebut dengan 10 nama mahasiswa yang memiliki nomor presensi berturt-turut setelah anda di kelas anda. Isi kode_kelas sesuai kode kelas anda saat ini. Ubahlah kode_dosen mata kuliah “Basis Data Dasar” kelas anda pada tabel jadwal sesuai dengan dosen pengampu mata kuliah basis data anda saat ini !

-- Selamat Mengerjakan –

Daftar Pustaka

- Dwi Puspitasari, S.Kom, “**Buku Ajar Dasar Basis Data**”, *Program Studi Manajemen Informatika Politeknik Negeri Malang*, 2012.