



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.

Klausula 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.

Klausula 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_kolom1 = nilai_baru1, nama_kolom2 = nilai_baru2, ...dst. **WHERE** nama_kolom_patokan [operator_perbandingan] nilai_patokan;

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

Operasi UPDATE bisa dilakukan pada tiga jenis attribut, 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.
	<pre> erDiagram jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_kelas : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_prodi : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_dosen : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_hari : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_mk : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_ruang : "has" jadwal_perkuliahan_jadwal --o{ jadwal_perkuliahan_jp : "has" jadwal_perkuliahan_jadwal { int PK_kode_jadwal varchar kode_kelas varchar kode_dosen varchar kode_mk varchar kode_ruang varchar kode_hari int jp_mulai int jp_selesai } jadwal_perkuliahan_kelas { varchar PK_kode_kelas varchar kode_prodi varchar nama_kelas } jadwal_perkuliahan_prodi { varchar PK_kode_prodi varchar nama_prodi } jadwal_perkuliahan_dosen { varchar PK_kode_dosen varchar nama_dosen } jadwal_perkuliahan_hari { varchar PK_kode_hari varchar nama_hari } jadwal_perkuliahan_mk { varchar PK_kode_mk varchar nama_mk } jadwal_perkuliahan_ruang { varchar PK_kode_ruang varchar nama_ruang varchar deskripsi_ruang } jadwal_perkuliahan_jp { int PK_kode_jp time jp_mulai time jp_selesai } </pre>
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 jelaskan maksud dari setiap tahapan tersebut.

2a	<pre>DROP DATABASE IF EXISTS jadwal_perkuliahan; CREATE DATABASE jadwal_perkuliahan; USE jadwal_perkuliahan;</pre>
2b	<pre>CREATE TABLE dosen (kode_dosen varchar(4) NOT NULL, nama_dosen varchar(100) DEFAULT NULL);</pre>
2c	<pre>CREATE TABLE hari (kode_hari varchar(3) NOT NULL, nama_hari varchar(10) DEFAULT NULL);</pre>
2d	<pre>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);</pre>
2e	<pre>CREATE TABLE jp (kode_jp int(3) NOT NULL, jp_mulai time DEFAULT NULL, jp_selesai time DEFAULT NULL);</pre>
2f	<pre>CREATE TABLE kelas (kode_kelas varchar(10) NOT NULL, kode_prodi varchar(3) DEFAULT NULL, nama_kelas varchar(5) DEFAULT NULL);</pre>
2g	<pre>CREATE TABLE mk (kode_mk varchar(5) NOT NULL, nama_mk varchar(100) DEFAULT NULL);</pre>

2h	<pre>CREATE TABLE prodi (kode_prodi varchar(3) NOT NULL, nama_prodi varchar(100) DEFAULT NULL) ;</pre>
2i	<pre>CREATE TABLE ruang (kode_ruang varchar(5) NOT NULL, nama_ruang varchar(20) DEFAULT NULL, deskripsi_ruang varchar(100) DEFAULT NULL) ;</pre>
2j	<pre>ALTER TABLE dosen ADD PRIMARY KEY (kode_dosen); ALTER TABLE hari ADD PRIMARY KEY (kode_hari); ALTER TABLE jadwal ADD PRIMARY KEY (kode_jadwal) ALTER TABLE jp ADD PRIMARY KEY (kode_jp); ALTER TABLE kelas ADD PRIMARY KEY (kode_kelas) ALTER TABLE mk ADD PRIMARY KEY (kode_mk); ALTER TABLE prodi ADD PRIMARY KEY (kode_prodi); ALTER TABLE ruang ADD PRIMARY KEY (kode_ruang);</pre>
2k	<pre>ALTER TABLE jadwal MODIFY kode_jadwal int(10) NOT NULL AUTO_INCREMENT;</pre>
2l	<pre>ALTER TABLE jadwal ADD FOREIGN KEY (kode_dosen) REFERENCES dosen (kode_dosen), ADD FOREIGN KEY (kode_mk) REFERENCES mk (kode_mk), ADD FOREIGN KEY (kode_ruang) REFERENCES ruang (kode_ruang), ADD FOREIGN KEY (kode_hari) REFERENCES hari (kode_hari), ADD FOREIGN KEY (jp_mulai) REFERENCES jp (kode_jp), ADD FOREIGN KEY (jp_selesai) REFERENCES jp (kode_jp), ADD FOREIGN KEY (kode_kelas) REFERENCES kelas (kode_kelas); ALTER TABLE kelas ADD FOREIGN KEY (kode_prodi) REFERENCES prodi (kode_prodi);</pre>
3	<p>Cek database Anda dengan perintah 'SHOW TABLES' untuk memastikan bahwa semua tabel sudah dibuat. Setelah selesai membuat database diatas, lanjutkan ke <u>Praktikum – Bagian 2.</u></p>

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 .
	<pre>INSERT INTO mk (kode_mk, nama_mk) VALUES ('02010', 'Basis Data');</pre>
	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.
	<pre>SELECT * FROM mk</pre>
2	Apabila data di-insert-kan pada semua kolom tabel, maka kita dapat langsung menggunakan klausa VALUES tanpa harus menuliskan nama-nama kolom dahulu.
	<pre>INSERT INTO mk VALUES('02041', 'Teknologi Data');</pre>
	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.
	<pre>SELECT * FROM mk</pre>
3	Untuk menambahkan beberapa kolom sekaligus dalam 1 statement digunakan statement dengan format seperti berikut.
	<pre>INSERT INTO mk VALUES ('02004', 'Aljabar Linier'), ('02005', 'Analisis Dan Desain Berorientasi Objek'), ('02006', 'Bahasa Indonesia');</pre>
	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.
	<pre>SELECT * FROM mk</pre>
4	Dan seperti berikut, jika hanya kolom tertentu saja yang akan diberi nilai dengan cara menyebutkan nama kolomnya.

	<pre>INSERT INTO mk (kode_mk, nama_mk) VALUES ('02001', 'Agama'), ('02002', 'Aljabar Linier'), ('02003', 'Algoritma dan Struktur Data');</pre> <p>Statement SQL tersebut menambahkan 3 baris baru ke tabel mk pada kolom yang ada pada struktur tabel mk.</p> <p>Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement SELECT seperti berikut.</p> <pre>SELECT * FROM mk</pre>
5	<p>Statement INSERT juga dapat dieksekusi dengan menggunakan klausa SET alih-alih VALUES.</p> <pre>INSERT INTO mk SET kode_mk = '02011', nama_mk = 'Desain Pemrograman Web';</pre> <p>Statement SQL tersebut menambahkan 1 baris baru ke tabel mk pada kolom yang ada pada struktur tabel mk.</p> <p>Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement SELECT seperti berikut.</p> <pre>SELECT * FROM mk</pre>
6	<p>Pada statement INSERT juga dapat digunakan klausa SELECT.</p> <p>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”)</p> <pre>INSERT INTO mk_backup SELECT * FROM mk;</pre> <p>Statement SQL tersebut menambahkan data baru dari tabel mk ke tabel mk_backup.</p> <p>Untuk melihat hasil SQL yang kita eksekusi tersebut, gunakan statement SELECT seperti berikut.</p> <pre>SELECT * FROM mk_backup;</pre>
7	<p>Setelah berhasil mengeksekusi SQL tersebut, lanjutkan ke Praktikum - Bagian 3.</p>

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:
	<pre>UPDATE jadwal SET kode_dosen='D010'</pre>
	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
	<pre>SELECT * FROM jadwal</pre>
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:
	<pre>UPDATE jadwal SET kode_dosen='D022' WHERE kode_mk='02010'</pre>
	<pre>SELECT * FROM jadwal WHERE kode_mk='02010'</pre>
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.
	<pre>UPDATE jadwal SET kode_dosen = NULL WHERE kode_dosen = 'D010'</pre>
	<pre>SELECT * FROM jadwal</pre>
5	Untuk mengubah beberapa kolom sekaligus dalam satu kali eksekusi statement UPDATE, dapat digunakan format berikut.
	<pre>UPDATE jadwal SET kode_dosen = 'D012', kode_ruang = '0702' WHERE kode_kelas = '2021020204'</pre>
	<pre>SELECT * FROM jadwal WHERE kode_kelas = '2021020204'</pre>

6	<p>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 Yudianto SKom., MKom.', maka dapat digunakan SQL dengan format berikut.</p>
	<pre>UPDATE jadwal SET kode_dosen = (SELECT kode_dosen FROM dosen WHERE nama_dosen='Dika Rizky Yudianto SKom., MKom.') WHERE kode_mk = '02010'</pre>
	<pre>SELECT * FROM jadwal WHERE kode_mk='02010'</pre>
7	<p>Setelah berhasil mengeksekusi SQL tersebut, lanjutkan ke <u>Praktikum - Bagian 4.</u></p>

Practicum - Part 4: DELETE Statement Experiment

Step	Description
1	DELETE is used to delete one or more rows from a table. Suppose we want to remove jadwal which has a value in the kode_dosen , then the basic format of the statement as follows we can use:
	<pre>DELETE FROM jadwal WHERE kode_dosen IS NOT NULL;</pre>
	<pre>SELECT * FROM jadwal</pre>
2	BE CAREFUL when we use DELETE statements without WHERE! Try executing the following SQL syntax:
	<pre>DELETE FROM jadwal;</pre>
3	All data in one schedule table will be lost!
	<pre>SELECT * FROM jadwal</pre>
4	Go to <u>Tasks</u> section!

Tugas

- Impor kembali **isi_data_jadwal_perkuliahan.sql**.
 - screenshot hasil dari masing-masing pertanyaan di bawah ini!
1. Ubah nama kursus "Database" menjadi "Database Dasar"!
 2. Ubah semua **jadwal kuliah mata kuliah** "Basic Database" menjadi hari Senin di jam pelajaran ke-5 hingga jam pelajaran ke-10!
 3. Hapus **jadwal perkuliahan** "Kewarganegaraan" di tabel **jadwal**!
 4. Tambahkan **mata kuliah** "Pancasila", "Bela Negara", "Wawasan Nusantara" di tabel **MK**!
 5. Hapus semua konten data pada tabel **mk_backup**!
 6. Isi data tabel **mk_backup** dengan isi **tabel mk**!
 7. Buat tabel siswa dengan atribut nim, nama_mahasiswa, kode_kelas. Di mana kode_kelas merujuk ke tabel kelas. Isi tabel dengan 10 nama siswa yang memiliki nomor presensi reguler setelah Anda yang berada di kelas Anda. Isi kode_kelas sesuai dengan kode kelas Anda saat ini. Ubah kode_dosen kursus "Basis Data Dasar" kelas Anda dalam tabel jadwal sesuai dengan dosen Anda saat ini!

-- Selamat Mengerjakan --

Reference

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