**A blue and white background

Description automatically generated**Shape, rectangle

Description automatically generated

Elok Nur Hamdana, S.T., M.T

Annisa Taufika Firdausi, ST., MT

Muhammad Shulhan Khairy, S.Kom, M.Kom

Farid Angga Pribadi, S.Kom.,M.Kom

Retno Damayanti, S.Pd., M.T.

PERTEMUAN 11

MySQL - DML

**Team Teaching:**

Dwi Puspitasari, S.Kom., M.Kom.

Yan Watequlis Syaifudin, ST., MMT., PhD.

Annisa Puspa Kirana, S. Kom, M.Kom

Yoppy Yunhasnawa, S.ST., M.Sc.

Candra Bella Vista, S.Kom., MT

**JOBSHEET**

PRAKTIKUM BASIS DATA

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG



|  |  |
| --- | --- |
|  | Jurusan Teknologi Informasi Politeknik Negeri Malang  **Jobsheet-11: MySQL – Data Manipulation Language (DML)**  **Mata Kuliah Basis Data**  Pengampu: Tim Ajar Basis Data  *2024* |

**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".

1. **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 jelaskan maksud dari setiap tahapan tersebut.** |
| 2a |  |
| 2b |  |
| 2c |  |
| 2d |  |
| 2e |  |
| 2f |  |
| 2g |  |
| 2h |  |
| 2i |  |
| 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**. |
|  |
| 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. |
|  |
| 2 | Apabila data di-insert-kan pada semua kolom tabel, maka kita dapat langsung menggunakan klausa VALUES tanpa harus menuliskan nama-nama kolom dahulu. |
|  |
| 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. |
|  |
| 3 | Untuk menambahkan beberapa kolom sekaligus dalam 1 statement digunakan statement dengan format seperti berikut. |
|  |
| 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. |
|  |
| 4 | Dan seperti berikut, jika hanya kolom tertentu saja yang akan diberi nilai dengan cara menyebutkan nama kolomnya. |
|  |
| 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. |
|  |
| 5 | Statement INSERT juga dapat dieksekusi dengan menggunakan klausa SET alih-alih VALUES. |
|  |
| 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. |
|  |
| 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”)** |
|  |
| 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. |
|  |
| 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: |
|  |
| 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 |
|  |
| 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: |
|  |
|  |
| 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. |
|  |
|  |
| 5 | Untuk mengubah beberapa kolom sekaligus dalam satu kali eksekusi statement UPDATE, dapat digunakan format berikut. |
|  |
|  |
| 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. |
|  |  |
|  |  |
| 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: |
|  |  |
|  |  |
| 2 | HATI-HATI apabila kita menggunakan statement DELETE tanpa WHERE! Cobalah eksekusi syntax SQL berikut: |
|  |  |
| 3 | Semua data dalam satu tabel jadwal akan hilang! |
|  |  |
| 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.