Contoh SQL Constraint

Anda dapat menggunakan constraint untuk membatasi tipe data yang disimpan ke dalam tabel. Constraint dapat digunakan pada saat pertama kali membuat table dengan statement CREATE TABLE atau setelah tabel dibuat dengan perintah statement ALTER TABLE.

Umumnya jenis Constraint mengandung:

- NOT NULL Constraint: untuk memastikan kolom dalam tabel tidak berisi nilai NULL.
  CREATE TABLE Pelanggan
  (Kode Integer NOT NULL,
  Nama Varchar (30) NOT NULL,
  Alamat Varchar(30));

- DEFAULT Constraint: menentukan nilai default pada kolom saat data diinsert pada tabel.
  CREATE TABLE Jurnal_Detail
  (Kode char(4) NOT NULL,
  Keterangan varchar (30),
  Debet Numeric DEFAULT 0,
  Kreditt Numeric DEFAULT 0));

- UNIQUE Constraint: untuk memastikan tidak ada data ganda dalam kolom.
  CREATE TABLE Pelanggan
  (Kode Integer UNIQUE,
  Nama Varchar (30),
  Alamat Varchar(30));

- CHECK Constraint: memastikan data dalam kolom memenuhi kriteria yang ditentukan.
  CREATE TABLE Pelanggan
  (Kode integer CHECK (Kode > 0),
  Nama varchar (30),
  Alamat varchar(30));
  Pada contoh di sini kriteria field Kode harus lebih besar dari 0, jika data dientry lebih kecil dari 0 akan terjadi error dan data tidak akan dapat disimpan ke table.

- Primary Key Constraint: digunakan untuk mengidentifikasi secara unik pada baris.
  MySQL:
  CREATE TABLE Pelanggan
  (Kode integer,
  Nama varchar(30),
  Alamat varchar(30),

PRIMARY KEY (Kode));

Oracle:
CREATE TABLE Pelanggan
(Kode integer PRIMARY KEY,
Nama varchar(30),
Alamat varchar(30));

SQL Server:
CREATE TABLE Pelanggan
(Kode integer PRIMARY KEY,
Nama varchar(30),
Alamat varchar(30));

- Foreign Key Constraint: digunakan untuk integritas referensi dari data.
  MySQL:
  CREATE TABLE ORDERS
  (Kode_Order integer,
  Tgl_Order date,
  Kode_Pelanggan integer,
  Jumlah double,
  Primary Key (Order_ID),
  Foreign Key (Kode_Pelanggan) references Pelanggan(Kode));

  Oracle:
  CREATE TABLE ORDERS
  (Kode_Order integer primary key,
  Tgl_Order date,
  Kode_Pelanggan integer references Pelanggan(Kode),
  Jumlah double);

  SQL Server:
  CREATE TABLE ORDERS
  (Kode_Oder integer primary key,
  Tgl_Order datetime,
  Kode_Pelanggan integer references Pelanggan(Kode),
  Jumlah double);

Referential Integrity

Perintah SQL untuk membuat tabel 'MHS'

```
CREATE TABLE mhs
(
  nim  varchar(8),
  namaMhs varchar(20),
  PRIMARY KEY (nim)
) TYPE = INNODB;
```

Perintah SQL untuk membuat tabel 'MK'

```
CREATE TABLE ambilMK
(
  nim  varchar(8),
  kodeMK varchar(3),
  nilai float(3,2),
  PRIMARY KEY (nim, kodeMK),
  FOREIGN KEY (nim) REFERENCES mhs (nim) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (kodeMK) REFERENCES mk (kodeMK) ON DELETE CASCADE ON UPDATE
CASCADE
) TYPE = INNODB;
```

dilanjutkan untuk membuat tabel untuk 'ambilMK'.

```
CREATE TABLE ambilMK
(
  nim  varchar(8),
  kodeMK varchar(3),
  nilai float(3,2),
  PRIMARY KEY (nim, kodeMK),
  FOREIGN KEY (nim) REFERENCES mhs (nim) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (kodeMK) REFERENCES mk (kodeMK) ON DELETE CASCADE ON UPDATE
CASCADE
) TYPE = INNODB;
```

Tabel di atas terdapat dua primary key yaitu NIM dan KODEMK. Sedangkan field NIM ini juga merupakan foreign key yang direferensikan dari field NIM yang ada dalam tabel MHS. Oleh karena itu tambahkan perintah "FOREIGN KEY (nim) REFERENCES mhs (nim)". Selanjutnya apa maksud dari "ON DELETE CASCADE"? Perintah ini maksudnya bila ada data NIM yang dihapus pada tabel MHS, maka secara otomatis data NIM yang ada dalam tabel AMBILMK ini

juga akan terhapus. Sedangkan “ON UPDATE CASCADE” digunakan untuk proses update otomatis pada NIM dalam tabel AMBILMK, apabila NIM yang ada di tabel MHS ini diupdate.

Hal yang sama juga kita terapkan untuk tabel AMBILMK. Dalam hal ini, KODEMK adalah sebagai foreign key yang direfensikan dari KODEMK yang ada dalam tabel MK.

Sekarang coba Anda masukkan data-data berikut ini pada tabel MHS

```
NIM       NAMAMHS
M0197001  Rosihan Ari Yuana
M0197002  Dwi Amalia Fitriani
M0197003  Faza Fauzan
M0197004  Nada Hasanah
M0197005  Muh. Ahsani Taqwim
```

Masukkan pula data pada tabel MK

```
KODEMK    NAMAMK
M01       Database
M02       OOP
```

untuk mengecek referensial integrity, sekarang kita coba masukkan data pada tabel AMBILMK.

```
INSERT INTO ambilmk VALUES ('M0197001', 'M01', 3.0);
```

Ketika perintah SQL di atas dijalankan, data dapat dimasukkan ke tabel AMBILMK dengan sukses. Kita lihat bahwa NIM M0197001 terdapat dalam tabel MHS, begitu pula pada kode matakuliah M01 yang ada pada tabel MK.

Sehingga isi tabel AMBILMK menjadi

```
NIM       KODEMK    NILAI
M0197001  M01       3.0
```

Sekarang kita coba masukkan data berikut ini

```
INSERT INTO ambilmk VALUES ('M0197006', 'M01', 3.0);
```

perintah di atas akan menghasilkan error. Hal ini disebabkan NIM M0197006 tidak ada dalam tabel MHS.

Sekarang kita coba melakukan proses update. Kita akan mengupdate NIM M0197001 menjadi M0197010 yang ada dalam tabel MHS.

```
UPDATE mhs SET nim = 'M0197010' WHERE nim = 'M0197001';
```

Hasil query di atas pada tabel MHS menjadi

```
NIM       NAMAMHS
M0197010  Rosihan Ari Yuana
M0197002  Dwi Amalia Fitriani
M0197003  Faza Fauzan
M0197004  Nada Hasanah
M0197005  Muh. Ahsani Taqwim
```

Sekarang Anda coba lihat isi tabel AMBILMK. Pastilah isinya menjadi berikut ini

```
NIM       KODEMK    NILAI
M0197010  M01       3.0
```

Selanjutnya kita coba update untuk KODEMK yang ada dalam tabel MK. Misalnya akan diubah kode mk M01 menjadi M09.

```
UPDATE mk SET kodeMK = 'M09' WHERE kodeMK = 'M01';
```

Hasil query di atas pada tabel MK adalah

```
KODEMK    NAMAMK
M09       Database
M02       OOP
```

bila kita lihat data di tabel AMBILMK, pastilah isinya menjadi

```
NIM       KODEMK    NILAI
M0197010  M09       3.0
```

Bagaimana dengan proses penghapusan? Kita cek aja… sekarang kita coba hapus data mahasiswa berNIM M0197010 dalam tabel MHS.

```
DELETE FROM mhs WHERE nim = 'M0197010';
```

Hasil dari query SQL di atas pada tabel MHS adalah

```
NIM       NAMAMHS
M0197002  Dwi Amalia Fitriani
M0197003  Faza Fauzan
M0197004  Nada Hasanah
M0197005  Muh. Ahsani Taqwim
```

Sekarang bila kita lihat isi tabel AMBILMK, pastilah menjadi kosong karena data pengambilan matakuliah terkait dengan mahasiswa NIM M0197010 ini ikut terhapus.

Trigger MySQL

Trigger adalah prosedur tersimpan pada Server DBMS yang secara otomatis dijalankan apabila data di dalam tabel berubah karena eksekusi perintah SQL (INSERT, UPDATE atau DELETE). Beberapa hal yang dapat dilakukan dengan Trigger adalah

- Mengupdate tabel-tabel lain jika ada perubahan (Insert, update atau delete) pada tabel yang sedang aktif.
- Untuk mengimplementasikan suatu sistem log. Setiap terjadi perubahan, secara otomatis akan menyimpan ke tabel log.
- Untuk melakukan validasi dan verifikasi data sebelum data tersebut disimpan.

Berikut adalah cara membuat Trigger database:

```
CREATE TRIGGER name

[BEFORE|AFTER] [INSERT|UPDATE|DELETE]

ON tablename

FOR EACH ROW statement
```

- Name adalah Nama trigger mengikuti peraturan penamaan variabel dalam MySQL
- [BEFORE | AFTER] untuk menentukan kapan proses secara otomatis akan dieksekusi (sebelum atau sesudah proses).
- [INSERT | UPDATE | DELETE] untuk menentukan proses yang dijadikan trigger (pemicu) untuk menjalankan perintah-perintah di dalam triggers.
- Tablename adalah nama tabel dimana trigger berada.
- Statement adalah sekumpulan perintah atau query yang akan secara otomatis dijalankan jika proses yang didefinisikan sebelumnya aktif.

Contoh Penggunaan database Trigger:

Pertama siapkan sebuah database dan tabel.

```
create table ITProgrammer (
kode varchar(10) not null,nama varchar(25) not null,
primary key (kode))
create table ITProgrammer2 (
kode varchar(10) not null,nama varchar(25) not null,
primary key (kode))

create table trans (
kode varchar(10) not null,
kodetrans varchar(10),jumlah double,
primary key nkode (kode,kodetrans))
```

Kemudian buat trigger seperti dibawah ini

```
create trigger auto_insert_ITProgrammer2
before insert on ITProgrammer for each row
begin
  insert into ITProgrammer2 (kode,nama) values (NEW.kode,NEW.nama);
end$$

create trigger auto_update_ITProgrammer2
before update on ITProgrammer for each row
begin
  update ITProgrammer2 set nama=NEW.nama where kode=NEW.kode;
end$$

create trigger auto_delete_ITProgrammer2
before delete on ITProgrammer for each row
begin
     delete from ITProgrammer2 where kode=OLD.kode;
     delete from trans where kode=OLD.kode;
end$$
```

untuk trigger tersebut apabila terjadi perubahan pada tabel ITProgrammer maka akan secara otomatis tabel ITProgrammer2 akan terpengaruh(ikut berubah).