PENGGUNAAN SQL

Kategori Utama Bahasa SQL

Secara umum, SQL dibagi atas 2 bagian, yakni :

- **DML (*Data Manipulation Language*),** yang memperbolehkan proses atau manipulasi obyek basis data.
- **DDL (*Data Definition Language*),** yang mendukung definisi atau pembuatan dari obyek basis data seperti tabel, indeks, *sequence* dan *view*.

- Meskipun pada kenyataannya, beberapa vendor basis data mempunyai lebih dari dua kategori.

Data Definition Language (DDL)

- DDL memiliki ciri khas tertentu, yaitu perintah-perintah yang menggunakan klausa, *statement*, pernyataan **CREATE**, **ALTER** atau **DROP.**

Contoh penggunaan DDL :

- **CREATE DATABASE**
- **CREATE TABLE**
- **SHOW TABLES**
- **ALTER TABLE**
- **DROP TABLE**

- Untuk Membuat *Database*

Sintaks:

CREATE DATABASE nama_database;

Contoh :

CREATE DATABASE db_akademik;

Catatan:

- Dalam membuat *database* usahakan jangan menggunakan spasi.
- Gunakan huruf kecil semua, untuk memudahkan mengingat string *database* dalam *programming*.

Data Definition Language - CREATE

- Sintaks DDL untuk pembuatan tabel :

  CREATE TABLE
  nama_tabel
  (*nama_kolom1* *tipe_data_kolom1* *opsi1*,
  ***nama_kolom2* *tipe_data_kolom2* *opsi2*,**
  ...);

- Dengan:
  - *nama_kolom* adalah nama dari *field* yang akan dibuat
  - Tipe_data_kolom adalah tipe data dari kolom tersebut
  - *Opsi* memiliki beberapa kemungkinan, seperti: **primary key**, **not null** dan lain sebagainya. Opsi dapat diletakkan di bawah, setelah menyebutkan semua *nama_kolom* dan *tipe data*.

Data Definition Language - CREATE

```
CREATE TABLE nama_tabel (
        nama_kolom1     tipe_data_kolom1,
        nama_kolom2     tipe_data_kolom2,
        ... ,
        PRIMARY KEY (nama_kolom),
        FOREIGN KEY (nama_kolom)
REFERENCES nama_tabel (nama_kolom)
        );
```

KEY pada Implementasi Tabel

Setiap *Key* yang diciptakan pada tabel memiliki fungsi tersendiri :

- **FOREIGN KEY ➔** Konsep tabel induk dan tabel anak. Merujuk pada tabel lain yang memiliki PRIMARY KEY, untuk menjaga *integrity constraint*.

- **INDEX KEY ➔** Untuk mempercepat pencarian. Efek samping: memperlambat proses INSERT, UPDATE dan DELETE.

- **UNIQUE KEY ➔** Nilai tiap-tiap record atau tupel hanya boleh ada satu.

- **PRIMARY KEY ➔** Berfungsi ganda, yakni sebagai referensi dari FOREIGN KEY, sebagai INDEX KEY dan juga sebagai UNIQUE KEY.

Integrity Constraint

- Merupakan batasan-batasan yang diberikan pada skema basis data, tujuannya untuk menjaga konsistensi data.

- Klausa-klausa yang termasuk dalam ***constraint*** adalah:
  - **NOT NULL** : kolom tidak boleh bernilai *null*
  - **UNIQUE** : kolom hanya memiliki nilai tunggal
  - **PRIMARY KEY** : identifikasi unik untuk setiap baris pada tabel
  - **FOREIGN KEY** : hubungan kolom dengan kolom dari tabel referensi
  - **CHECK** : memberikan suatu kondisi yang bernilai benar

DATA DEFINITION LANGUAGE - ALTER

- Alter berfungsi untuk merubah, menambahi, menghapus sesuatu pada obyek yang telah dibuat.

- Misalnya:
  - Menambah kolom
  - Memodifikasi kolom
  - Memberikan nilai default pada kolom baru
  - Menghapus kolom

DATA DEFINITION LANGUAGE - ALTER

- Perintah dasar **ALTER** untuk **menambah kolom tabel.**

ALTER TABLE *nama_tabel* **ADD COLUMN**
(*nama_kolom* *tipe_data* **[DEFAULT** *ekspresi***],**
nama_kolom *tipe_data* **[DEFAULT** *ekspresi***],**
…);

- Perintah **ALTER** untuk **modifikasi tabel.**

ALTER TABLE nama_tabel MODIFY
(*nama_kolom* *tipe_data* **[DEFAULT** *ekspresi***]**
nama_kolom *tipe_data* **[DEFAULT** *ekspresi***]**
…);

DATA DEFINITION LANGUAGE - ALTER

- PERINTAH ALTER untuk menambah PRIMARY KEY.

ALTER TABLE *nama_tabel*
ADD CONSTRAINT ***nama_kunci*** **PRIMARY KEY** (*nama_kolom*);

- Perintah ALTER untuk menambah *foreign key*.

ALTER TABLE *nama_tabel*
ADD CONSTRAINT ***nama_kunci*** **FOREIGN KEY**(*nama_kolom_tabel_anak*)
REFERENCES *nama_tabel_induk* (*nama_kolom_tabel_induk*);

DATA DEFINITION LANGUAGE - ALTER

- Perintah ALTER untuk menghapus kolom.

  ALTER TABLE *nama_tabel* **DROP (***nama_kolom***);**

- Merubah nama tabel.

  ALTER TABLE *nama_lama* **RENAME** *nama_baru***;**

- Menghapus tabel.

  DROP TABLE *nama_tabel opsi*;

Ringkasan DDL

- **CREATE**
  - CREATE SCHEMA / CREATE DATABASE
  - CREATE TABLE
  - CREATE VIEW
- **ALTER**
  - ALTER SCHEMA / ALTER DATABASE
  - ALTER TABLE
  - ALTER VIEW
- **DROP**
  - DROP SCHEMA
  - DROP TABLE
  - DROP VIEW
- **CONSTRAINT**, contoh:
  - CREATE TABLE … FOREIGN KEY … REFERENCES…
  - ALTER TABLE … ADD FOREIGN KEY … REFERENCES …
  - ALTER TABLE … DROP FOREIGN KEY …

Contoh merubah nama kolom/field pada tabel:

- Sintaks:

  ALTER TABLE *nama_table* CHANGE COLUMN *field_lama field_baru tipe_data kriteria;*

Contoh :

- Nama Tabel: **mahasiswa**
- Merubah Kolom **nim** menjadi **id_mahasiswa**
- Sintaks:

  ALTER TABLE *mahasiswa* CHANGE COLUMN *nim id_mahasiswa VARCHAR(8) NOT NULL;*

- **Menghapus Data dari Tabel :**

  TRUNCATE *nama_tabel*;

  Perintah tersebut guna menghapus baris pada tabel, alias mengosongkan tabel.

- **Menghapus Tabel**

  DROP TABLE *nama_tabel;*

Data Manipulation Language

Pada dasarnya, perintah DML terdiri atas 4 model dasar, yakni:

- SELECT
- INSERT
- UPDATE dan
- DELETE.

DML (*Data Manipulation Language*) :

- **SELECT *nama_field* FROM *nama_tabel***
- **INSERT INTO *nama_tabel* (*field1,field2,…*) VALUES (*nilai1,nilai2,…*)**
- **UPDATE *nama_tabel* SET *field1=nilai1,….***
- **DELETE FROM *nama_tabel* WHERE *field1=nilai1***

SQL

Command SQL :

Perintah	Keterangan
CREATE	Membuat tabel atau field
ALTER	Mengubah tabel dengan menambah field atau mengubah definisi field
DROP	Men-DROP tabel
SELECT	Mendefinisikan recordset, data apa yang akan ditampilkan dari database
INSERT	Menyisipkan recordset
UPDATE	Mengubah recordset
DELETE	Menghapus recordset

SQL

Ketika menggunakan ***query***, pengguna dapat menggunakan klausa berikut untuk diimplementasikan dalam ***statement SQL***. Klausa SQL:

Klausa	Keterangan
FROM	Menentukan tabel mana yang datanya akan ditampilkan
WHERE	Menentukan kondisi query
GROUP BY	Menentukan grup / kelompok dari informasi yang dipilih
HAVING	Digunakan bersama GROUP BY untuk menentukan kondisi untuk tiap grup dalam query
ORDER BY	Menentukan urutan *(sort)* data dari query

Data Manipulation Language - INSERT

- **INSERT** merupakan perintah untuk memasukkan data ke dalam tabel.
- Sintaks dasarnya:

  INSERT INTO *nama_tabel* (*nama_kolom1*, *nama_kolom2*, …)

  VALUES (*nilai1*, *nilai2*, …);
- Jika *nama_kolom* yang akan di-*insert* urutannya telah sesuai dengan yang ada pada struktur tabel, *nama_kolom* tidak perlu disebutkan. Sehingga sintaksnya dapat diperpendek menjadi:

  INSERT INTO *nama_tabel* **VALUES** (*nilai1*, *nilai2*, …);

- Catatan:

  Nilai yang tipe datanya *string* harus menggunakan tanda petik tunggal yang mengapit nilai tersebut.

Data Manipulation Language - UPDATE

- **UPDATE** digunakan untuk mengubah record atau tupel yang telah di-*insert* sebelumnya. Jika tanpa menggunakan syarat, semua tupel akan diganti.
- Sintaks dasarnya:

  UPDATE *nama_tabel*
  SET *nama_kolom* = *nilai*
  [**WHERE** *nama_kolom* operator *syarat*];

- Ada macam-macam tipe operator:
  - Single value: =, <. >, <=, >=
  - Multi value : IN, ALL, ANY

Data Manipulation Language - DELETE

- **Delete** digunakan untuk menghapus tupel. Jika tanpa menggunakan syarat, semua data dalam tabel tersebut akan dihapus.

- Sintaks dasarnya:

  DELETE FROM *nama_tabel*

  [**WHERE** *nama_kolom* operator *syarat*;]

Data Manipulation Language - SELECT

- Perintah ini digunakan untuk mengambil data dari dalam tabel.
- Merupakan perintah yang paling sering digunakan jika dibandingkan perintah-perintah SQL yang lain.

- Struktur dasar sintaks SELECT:

  SELECT [DISTINCT] nama_kolom1, nama_kolom2, …

  FROM daftar_nama_tabel

  [**WHERE** *nama_kolom* operator *syarat* [**AND**/**OR** nama_kolom2 operator syarat …]

  [**GROUP BY** nama_kolom]

  [**HAVING** *fungsi_agregasi*(*nama_kolom*) operator *syarat*]

  [**ORDER BY** nama_kolom **ASC**/**DESC**];

Ekspresi Aritmetika

Operator	Deskripsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian

Urutan Pengerjaan: *, /, +, -

Operator Perbandingan

Operator	Arti
=	Sama dengan
>	Lebih besar dari
>=	Lebih besar atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
<>	Tidak sama dengan
!=	Tidak sama dengan
^=	Tidak sama dengan
BETWEEN ... AND ...	Berada di antara 2 value
IN(*himpunan*)	Yang cocok dengan salah satu yang terdapat dalam set
LIKE	Yang cocok dengan pola karakter tertentu
IS NULL	Jika value-nya merupakan nilai null

Logika Kondisi

Operator	Arti
AND	Menghasilkan TRUE apabila kedua komponen benar
OR	Menghasilkan TRUE apabila salah satu komponen benar
NOT	Menghasilkan TRUE apabila kondisinya false

Tata Urutan Operator

- Dari sekian banyak operator yang telah disebutkan sebelumnya, tata urutan pengerjaannya dapat dilihat pada tabel berikut.

Urutan Pengerjaan	Operator
1	Operator aritmetika
2	Operator penggabungan
3	Operator perbandingan
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Kondisi logika NOT
7	Kondisi logika AND
8	Kondisi logika OR

Operator And

- AND digunakan sebagai kriteria "DAN"
- Penggunaan : operand AND operand

Operator Or

- OR digunakan sebagai kriteria "ATAU"
- Penggunaan : operand OR operand

Operator Not

- Untuk menyatakan "TIDAK" atau "BUKAN"
- Penggunaan : NOT kondisi