Bab 5
STRUKTUR QUERY LANGUAGE, SQL DAN DDL

Muhammad Imam Dinata, S.Kom.,M.T

SQL adalah Non-Procedural yang digunakan untuk mengakses database. SQL adalah singkatan dari Structured Query Language. Sedangan **pengertian SQL adalah**suatu bahasa (language) yang digunakan untuk mengakses data di dalam sebuah database relasional. SQL sering juga disebut dengan istilah query, dan bahasa SQL secara praktiknya digunakan sebagai bahasa standar untuk manajemen database relasional. Hingga saat ini hampir seluruh server database atau software database mengenal dan mengerti bahasa SQL.

Sejarah SQL

Awal mula lahirnya bahasa SQL yaitu pada bulan Juni 1970, dimana saat Jhonny Oracle yang merupakan seorang peneliti dari perusahaan IBM memiliki gagasan pembuatan basis data relasional, ide tersebut dituangkan dalam sebuah artikel. Di dalam artikel tersebut juga dibahas mengenai kemungkinan membuat sebuah bahasa standar untuk mengakses data dalam database tersebut. Bahasa standar tersebut diberinama SEQUEL (Structured English Query Language). Setelah kemunculan artikel tersebut lalu IBM memutuskan untuk mengembangkan pembuatan bahasa SEQUEL. Namun penamaan SEQUEL dalam bahasa standar tersebut bermasalah dengan hukum sehingga diubahlah menjadi SQL.

Tahun 1992 muncul versi SQL92 dan di tahun 1999 dengan berbagai pembaharuan muncullah SQL99. Namun, dalam dunia IT SQL yang sering digunakan mereferensikan pada SQL92.

5.1 SQL DATABASE TABLES

Sebuah database harus memilikisatu atau lebih table(File). Setiap table tersebut diidentifikasikan dengan nama (misal “Dosen” atau “Mahasiswa” dan sebagainya). Setiap tabel memiliki records (Baris-Baris)

Nama Depan	Nama Belakang	Alamat	Kota
Edy Syahputra	Nasution	Barelang	Batam
Putri Handayani	Handayani	Nongsa	Batam
Firman Ardimas	Ardiams	Batam Centre	Batam

Berikut ini contoh dari sebuah tabel yang diberi nama mahasiswa :

Tabel diatas memiliki tiga record (Satu Baris untuk setiap mahasiswa) dan empat kolom (field) dengan Nama Depan,Nama Belakang,Alamat dan Kota.

5.2 SQL SYNTAKS

SQL merupakan singkatan dari *Structured Query Language*. SQL atau juga sering disebut sebagai query merupakan suatu bahasa (*language*) yang digunakan untuk mengakses database. SQL dikenalkan pertama kali dalam IBM pada tahun 1970 dan sebuah standar ISO dan ANSII ditetapkan untuk SQL. Standar ini tidak tergantung pada mesin yang digunakan (IBM, Microsoft atau Oracle). Hampir semua software database mengenal atau mengerti SQL. Jadi, perintah SQL pada semua software database hampir sama.

Terdapat 3 (Tiga) jenis perintah SQL, yaitu **DDL, DML dan DCL**.

A. DDL atau Data Definition Language

DDL merupakan perintah SQL yang berhubungan dengan pendefinisian suatu struktur database, dalam hal ini*database* dan *table*. . Atau juga merupakan kelompok perintah yang

berfungsi untuk mendefinisikan atribut-atribut database, table, atribut kolom, batasan-batasan terhadap suatu atribut serta hubungan antar table.

Beberapa perintah dasar yang termasuk DDL ini antara lain :

- **CREATE**

Perintah ini digunakan untuk membuat, termasuk di antaranya membuat database baru, tabel baru, view baru, dan kolom.

Contoh:

create table mahasiswa (nim char(8) primary key,nama_mahasiswa varchar(20),nilai integer(3),alamat varchar(25);

- **ALTER**

Perintah ALTER TABLE digunakan untuk mengubah struktur tabel dalam database
Contoh :

alter table mahasiswa (nim char(8) primary key, add nama_mahasiswa varchar(20);

- **RENAME**

Dengan perintah ini digunakan untuk mengubah nama tabel A menjadi tabel B, atau sebaliknya

Contoh:

rename table mahasiswa to table mahasiswa_ti;

- DROP

DROP : Perintah ini digunakan untuk menghapus database dan tabel.

Contoh :

Alter table ‘mahasiswa’ drop ‘alamat’

B. DML atau Data Manipulation Language

DML merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau *record* dalam table. Perintah SQL yang termasuk dalam DML antara lain :

- SELECT

Perintah ini digunakan untuk mengambil data atau menampilkan data dari satu tabel atau beberapa tabel dalam relasi. Data yang diambil dapat kita tampilkan dalam layar prompt MySQL secara langsung maupun ditampilkan pada tampilan aplikasi.

Contoh :
Select nama_mahasiswa from mahasiswa where nilai = 70;

- INSERT

Perintah ini digunakan untuk menyisipkan atau memasukkan data baru ke dalam tabel. Penggunaannya setelah database dan tabel selesai dibuat.

Contoh :

Insert into mahasiswa values (“08052926”, “Frenky”,”70”);

- UPDATE

Perintah ini digunakan untuk memperbarui data lama menjadi data terkini. Jika Anda memiliki data yang salah atau kurang up to date dengan kondisi sekarang, maka dapat diubah isi datanya menggunakan perintah UPDATE.

Contoh:

>mahasiswa set nim = ‘08052926’ = 08052927;

- DELETE

Perintah ini digunakan untuk menghapus data dari tabel. Biasanya data yang dihapus merupakan data yang sudah tidak diperlukan lagi. Pada saat menghapus data, perintah yang telah dijalankan tidak dapat digagalkan, sehingga data yang telah hilang tidak dapat dikembalikan lagi

Contoh :

>delete form mahasiswa;

C. **DCL atau Data Control Language**

DCL (bukan BCL) merupakan perintah SQL yang berhubungan dengan pengaturan hak akses user MySQL, baik terhadap server, database, tabel maupun field. Perintah SQL yang termasuk dalam DCL antara lain :

- GRANT

Perintah ini digunakan untuk memberikan hak/ijin akses oleh administrator (pemilik utama) server kepada user (pengguna biasa). Hak akses tersebut berupa hak membuat (CREATE), mengambil (SELECT), menghapus (DELETE), mengubah (UPDATE), dan hak khususberkenaan dengan sistem databasenya.

- REVOKE

Perintah ini memiliki kegunaan terbalik dengan GRAND, yaitu untuk menghilangkan atau mencabut hak akses yang telah diberikan kepada user oleh administrator.

5.3 DDL atau Data Definition Language

Contoh DDL dibawah ini adalah :

1. MEMBUAT TABEL

Berikut adalah generik sintaks SQL untuk membuat tabel MySQL:

CREATE TABLE table_name (column_name column_type);

```
tutorials_tbl (
tutorial_id INT NOT NULL AUTO_INCREMENT,
tutorial_title VARCHAR(100) NOT NULL,
tutorial_author VARCHAR(40) NOT NULL,
submission_date DATE,
PRIMARY KEY ( tutorial_id )
);
```

Berikut beberapa item perlu penjelasan:

- Lapangan Atribut **NOT NULL** sedang digunakan karena kita tidak ingin field ini untuk menjadi NULL.SO jika pengguna akan mencoba untuk membuat rekor dengan nilai NULL maka MySQL akan meningkatkan kesalahan.
- Lapangan Atribut **AUTO_INCREMENT** memberitahu ke MySQL untuk terus maju dan menambahkan nomor yang tersedia di sebelah field id.
- **PRIMARY KEY** kata kunci digunakan untuk mendefinisikan kolom sebagai kunci primer.Anda dapat menggunakan beberapa kolom dipisahkan dengan tanda koma untuk mendefinisikan kunci primer

2. MEMBUAT INDEX

Sebuah indeks database adalah sebuah struktur data yang meningkatkan kecepatan operasi dalam sebuah tabel.Indeks dapat dibuat menggunakan satu atau lebih kolom, menyediakan dasar untuk kedua pencarian acak yang cepat dan efisien pemesanan akses ke catatan.Sementara menciptakan indeks itu harus dipertimbangkan bahwa apa yang kolom yang akan digunakan untuk membuat query SQL dan membuat satu atau lebih indeks pada kolom tersebut.Praktis, Indeks juga jenis tabel yang menjaga kunci primer atau bidang indeks dan pointer ke setiaprecord dalam ke meja yang sebenarnya.

Para pengguna tidak dapat melihat index, mereka hanya digunakan untuk mempercepat query dan akan digunakan oleh Database Engine Search untuk mencari catatan sangat cepat.INSERT dan UPDATE laporan membutuhkan waktu lebih lama di meja memiliki indeks di mana sebagai pernyataan SELECT menjadi cepat pada mereka tabel.Alasannya adalah bahwa saat melakukan insert atau update, database perlu inert atau memperbarui nilai indeks juga.

A. UNIQE INDEX

Membentuk indeks yang tunggal dari tabel. Sebuah Indeks unik adalah menjadikan dua baris tidak mungkin memiliki kesamaan nilai indeks,Bentuk Penulisannya :

```
CREATE UNIQUE INDEX index_name
ON table_name ( column1, column2,...);
```

Dan dapat menggunakan satu atau lebih kolom untuk membuat indeks.Sebagai contoh kita dapat membuat indeks pada tutorials_tbl menggunakan tutorial_author

```
CREATE UNIQUE INDEX AUTHOR_INDEX
ON tutorials_tbl (tutorial_author)
```

dapat membuat index sederhana di atas meja. Hanya menghilangkan UNIQUE dari query untuk membuat indeks sederhana.Indeks sederhana ini memungkinkan nilai-nilai duplikat dalam sebuah tabel.Jika Anda ingin indeks nilai dalam sebuah kolom di urutan, Anda dapat menambahkan kata DESC dicadangkan setelah nama kolom:

```
mysql> CREATE UNIQUE INDEX AUTHOR_INDEX
ON tutorials_tbl (tutorial_author DESC)
```

5.4 DML atau Data Manipulation Language

Contoh DML dibawah ini adalah .

- **SELECT**

Perintah **SELECT** pada MySql biasanya digunakan untuk menampilkan data yang berada di dalam tabel. Perintah **SELECT**mempunyai banyak sekali variasi. Mungkin bisa disebut perintah yang mempunyai variasi paling banyak di antara perintah-perintah lainnya.

Seperti yang kita ketahui bahwa MySql adalah sebuah **RDBMS(Relational Database Management System)** yang artinya bahwa tabel-tabel tersebut **bisa ber-relasi/tidak ber- relasi dengan tabel lainnya** perintah**SELECT** adalah untuk menampilkan data yang berada di dalam tabel. Hal tersebut nantinya akan kita sebut sebagai **Join Table** dengan menggunakanperintah **SELECT**. Terdapat dua Pilihan untuk select table, yaitu :

- **Inner Join**
- **Outer Join**.

Bentuk umum perintah select

- select [kolom-yang-ingin-ditampilkan] from [nama tabel]
- where [condition];

*penggunaan **where[condition]** merupakan opsional/pilihan/tidak wajib, gunakan jika memang dibutuhkan.

➢ **Menampilkan semua data dalam sebuah table**

- select * from mahasiswa;

➢ **Menampilkan data dalam kolom tertentu**

- select nama,fakultas,jurusan from mahasiswa;

➢ **Menampilkan data dalam kolom tertentu dengan kondisi tertentu**

➢ select nama,jurusan from mahasiswa where pendaftaran=2001;

DELETE

Apabila suatu data pada sebuah table tidak digunakan lagi, kita dapat menghapus data tersebut dengan menggunakan pernyataan DELETE. Bentuk umum perintah DELETE adalah sebagai berikut :

```
DELETE

FROM nama_table

[WHERE kondisi];
```

Pemakaian **WHERE** bersifat opsional, artinya :

- jika disertakan, maka hanya baris tertentu saja yang terhapus.
- jika tidak disertakan, semua baris pada tabel bersangkutan akan dihapus.

Berhati-hatilah dalam penggunaan perintah **DELETE**.

Sebelumnya mari kita buat sebuah tabel untuk latihan. Kita beri nama tabel **"mahasiswa"**, dengan kolom/column/fieldnim dan **nama**. Berikut ini adalah kutipan perintahnya :

```
CREATE TABLE mahasiswa (

nim int(5) NOT NULL AUTO_INCREMENT,

nama varchar(30) NOT NULL,

PRIMARY KEY (nim)

);.
```

Berikut ini adalah beberapa contoh penggunaan perintah DELETE berserta variasinya :

1. Menghapus sebuah baris

```
DELETE FROM mahasiswa

WHERE nim = '12345';
```

Jika diterjemahkan maka akan berarti : menghapus data dari tabel mahasiswa yang mempunya nim 12345.

2. Menghapus beberapa baris

```
DELETE FROM mahasiswa

WHERE nim '12345' AND '12346';
```

Jika diterjemahkan maka akan berarti : menghapus data dari tabel mahasiswa yang mempunyai nim 12345 dan 12346.

3. Menghapus semua baris

```
DELETE FROM mahasiswa;
```

□ PENGGUNAAN LEBIH DARI SATU TABEL□

Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggunakan bentuk perintah JOIN. Dalam tutorial ini, akan dijelaskan secara bertahap mengenai bagaimana menggabungkan dua tabel atau lebih, terutama untuk menampilkan data yang berasal dari beberapa tabel. Contoh-contoh dalam tutorial ini secara khusus telah dicoba di database MySQL, namun demikian secara umum perintah penggabungan tabel di semua jenis database tidak jauh berbeda alias sama.

Masing-masing vendor database memiliki bahasanya sendiri sebagaian besar spesifikasinya mengacu pada standar ANSI tersebut dengan berbagai ekstensi tambahan. SQL Server menggunakan bahasa Transact-SQL dalam produknya, sedangkan Oracle menggunakan PL/SQL. Penggabungan tabel dalam perintah SQL menggunakan keyword JOIN , berikut ini, merupakan perintah join dalam syntax SQL atau ANSI

3. INNER JOIN

Inner Join adalah perintah untuk operasi penggabungan yang paling umum digunakan dan biasanya dikatakan sebagai default join-type. Inner Join hanya menampilkan data yang benar-benar terdapat di dalam tabel yang saling dihubungkan.Misalnya menggabungkan tabel Customers dengan Orders, maka field penghubung yang digunakan adalah CustomerID. Dalam teknik INNER JOIN maka hasil yang ditampilkan hanya record yang memiliki CustomerID sama di kedua tabel tersebut. Apabila terdapat customr yang CustomerID nya tidak ditemukan di tabel Orders maka data tersebut tidak ditampilkan.

Berikut ini contoh perintah yang menggunakan syntax inner join :

```
SELECT CompanyName, OrderID, OrderDate
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

Perintah SQL tersebut mengambil field CompanyName dari tabel Customers, sedangkan field OrderID dan OrderDate diambil dari tabel Orders. Kunci utama penggabungan dua tabel tersebut adalah keyword INNER JOIN dengan kriteria CustomerID.

INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID

Antara nama tabel dengan nama field dibatasi dengan tanda titik (.) misalnya Orders.CustomerID. Tujuan pencantuman nama tabel tersebut adalah menghindari ambiguitas yang mengakibatkan error apabila perintah tersebut dieksekusi. Karena CustomerID tersebut terdapat di kedua tabel maka nama tabel harus dicantumkan agar dapat diidentifikasi secara unik field mana yang dimaksud. Untuk memberikan gambaran lebih kompleks mengenai penerapan penggabungan tabel ini anda dapat memodifikasi contoh perintah yang terdapat penggunaan

GROUP BY. Pada perintah tersebut anda menghitung Jumlah dan rataan produk yang terjual, tetapi hanya ditampilkan ProductID saja sedangkan nama produknya tidak terlihat karena terdapatdi tabel lain. Anda dapat menggabungan tabel Order Details tersebut dengan tabel Products yang menyimpan nama produk. Dengan demikian dapat ditampilkan baik ProductID maupun nama produknya.

4. LEFT JOIN

Berikut merupakan contoh perintah join dalam syntax left join :

```
SELECT Customers.CustomerID, Customers.CompanyName,
COUNT(Orders.OrderID) AS Frekuensi
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.CompanyName
ORDER BY COUNT(Orders.OrderID)
```

Karena menggunakan perintah LEFT JOIN maka semua data di tabel Customers ditampilkan seluruhnya walaupun frekuensi ordernya 0 Kita dapat melihat terdapat 2 customer yang frekuensi 0 tetapi tetap ditampilkan. Di akhir tampilan hasil tersebut terlihat ada 91 baris yang ditampikan. Untuk melakukan pengecekan jalankan perintah berikut :

```
SELECT COUNT(*) from dbo.Customers
```

Artinya terdapat 91customer di tabel Customers, yaitu sama dengan jumlah baris di perintah sebelumnya.Dengan demikian memang benar bahwa LEFT JOIN telah menampilkan seluruh Customer yang berjumlah 91.

5. RIGHT JOIN

Perintah RIGHT JOIN merupakan kebalikan dari LEFT JOIN, yaitu menampilkan semua isi tabel yang disebutkan kedua dalam perintah join. Dalam contoh di atas apabila LEFT JOIN diganti RIGHT JOIN maka semua isi tabel Orders akan ditampilkan semuanya.

Cobalah memodifikasi perintah diatas menjadi sebagai berikut :

```
SELECT Customers.CustomerID, Customers.CompanyName,
COUNT(Orders.OrderID) AS Frekuensi
FROM Customers
RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.CompanyName
ORDER BY COUNT(Orders.OrderID)
```

Setelah anda jalankan perintah tersebut lihatlah perbedaaanya dengan perintah LEFT JOIN sebelumnya. Maka terlihat hanya dihasilkan 89 baris, berarti terdapat 2 customer yang CustomerID nya tidak terdapat di table Orders atau frekuensi ordernya = 0.

6. FULL JOIN

Jenis join terakhir adalah FULL JOIN yang menampilkan semua data dari dua tabel yang dihubungkan meskipun terdapat data yang tidak memiliki pasangan di tabel lainnya. Misalnya kita mengambil data dari tabel Country dengan tabel City menggunakan FULL JOIN. Data dari kedua tabel akan ditampilkan semuanya baik untuk nama kota yang tidak memiliki data negara maupun sebaliknya.

7. EXIST

Seperti halnya perintah **IN** perintah **EXIST** juga sering dipergunakan dalam subquery di SQL, walaupun ada sedikit perbedaan cara penulisan keduanya tetapi kita bisa menggunakan salah satu dari perintah ini jika dalam suatu kasus kita harus menggunakan subquery.

Contoh:

Anda mempunyai sekumpulan data dalam suatu tabel, kita beri nama saja tabel_A berisi {‘1001’, ‘1002’, ‘1003’, ‘1004’, ‘1005’}. Dan Anda juga mempunyai tabel yang lain, anggap saja namanya tabel_B berisi {‘1002’,’1005’,’1006’,’1007’}.

Dari dua tabel ini Anda ingin menampilkan anggota himpunan tabel_A yang juga merupakan anggota himpunan tabel_B. Kalau kita terjemahkan dalam konsep himpunan, ini merupakan irisan dari dua himpunan. Dan hasilnya {‘1002’,’1005’}. Tapi Anda harus ingat dalam praktek pengelolaan data nantinya kita harus berhadapan dengan jumlah ribuan data bahkan jutaan data lebih yang harus dikelola cepat. Jadi tidak bisa lagi dengan manual seperti diatas cukup dilihat kita sudah tahu jawabannya. Untuk itu perlu dikelola dengan perintah SQL. Contoh diatas merupakan contoh sederhana tapi sering kali kita ketemu kasus tersebut.

Bentuk perintah SQL-nya sebagai berikut

```
SELECT* FROM tabel_A a WHERE EXISTS(SELECT* FROM table_B b WHERE b.kode = a.kode)
```

Silakan dicoba hasilnya sama dengan yang diinginkan.Kasus kebalikannya juga sering terjadi dalam mengelola database, yaitu Anda ingin menampilkan himpunan tabel_A dimana tidak mengandung suatu anggota himpunan tabel_B.

- tabel_A = {‘1001’, ‘1002’, ‘1003’, ‘1004’, ‘1005’}
- tabel_B = {‘1002’,’1005’,’1006’,’1007’}

disini terlihat {‘1001’,’1003’,’1004’} bukan merupakan anggota himpunan tabel_B

Dapat diterjemahkan dalam bahasa SQL sebagai berikut:

```
SELECT* FROM tabel_A a WHERE NOT EXISTS(SELECT* FROM table_B b WHERE b.kode = a.kode)
```

Dalam perintah ini ada tambahan perintah **NOT** .Dalam mencoba perintah-perintah diatasAnda harus mempunyai database sendiri, kemudian di **DATABASE** tersebut harus **CREATE TABLE** tabel_A dan tabel_B yang kemudian datanya diisi dengan perintah **INSERT** ke masing- masing tabelnya. Diharapkan Anda sudah mengerti dalam urusan membuat **DATABASE** dan **CREATE TABLE** ini.

2. IN

Fungsi lain yang mengikuti where adalah in dan not in. Fungsi in adalah untuk memberikan beberapa batasan data. Penggunaan in atau not in yaitu untuk melakukan filtering terhadap record

yang dipilih. Jika data pada suatu kolom sesuai dengan daftar in atau not in maka record yang mengandung data tersebut ditampilkan.

3. Some dan All

Perintah Some dan All digunakan untuk membandingkan isi field dengan isi field table lain sebagai table pembanding. Bedanya Some membandingkan salah satu record pada table lookup sedangkan All membandingkan semua record pada table look.

Perbedaan antara some dan all dapat dilihat dengan membandingkan query sebelumnya dengan query di atas. Jika some, nilai akan bernilai benar jika memenuhi syarat (dalam hal ini >) minimal satu record hasil subquery. Namun untuk all, akan bernilai benar jika syarat (dalam hal ini <) terpenuhi pada semua record hasil subquery.

4. EXISTS Dan NOT EXISTS

Digunakan untuk melakukan pengecekan apakah hasil dari suatu ‘correlated nested query’ berisi tuple atau tidak.

EXISTS (Q): Memberikan nilai return True, jika dalam hasil query Q minimal terdiri dari satu tuple

NOT EXISTS (Q): Memberikan nilaai return TRUE, jika tak satupun tuple yang dihasilkan dalam hasil query Q.