STRUCTURE QUERY LANGUAGE

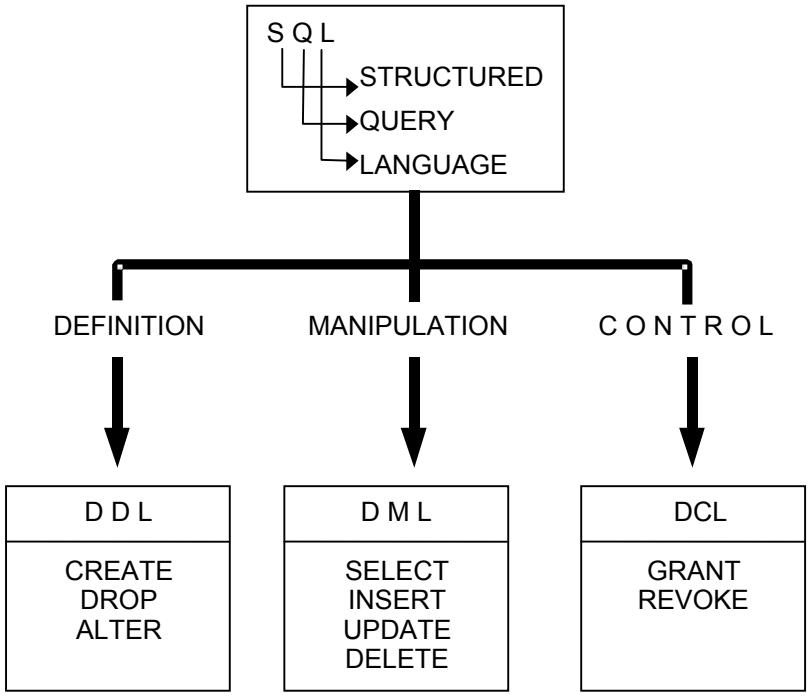
Structure Query Language (SQL) merupakan komponen bahasa *relational database system*. SQL merupakan bahasa baku (ANSI/SQL), *non procedural*, dan berorientasi himpunan (*set-oriented language*). SQL dapat digunakan baik secara interaktif atau ditempelkan (*embedded*) pada sebuah program aplikasi.

Komponen-Komponen SQL

a. **Data Definition Language** (DDL) :
   Digunakan untuk mendefinisikan data dengan menggunakan perintah : *create, drop, alter.*

b. **Data Manipulation Language** (DML) :
   Digunakan untuk memanipulasi data dengan menggunakan perintah : *select, insert, update, delete.*
   Data Manipulation Language merupakan bagian terpadu bahasa SQL. Perintah-perintahnya dapat dibuat secara interaktif atau ditempelkan pada sebuah program aplikasi. Pemakai hanya perlu menentukan 'APA' yang ia inginkan, DBMS menentukan 'BAGAIMANA' cara mendapatkannya.

c. **Data Control Language** (DCL) :
   Digunakan untuk mengontrol hak para pemakai data dengan perintah : *grant, revoke*



DATA DEFINITION LANGUAGE

1. CREATE TABLE

Fungsi : membuat tabel

Sintaks : **CREATE TABLE tbname**

(col 1 data type data spec,

col 2 data type data spec,

.

.

PRIMARY KEY (col1,……))

Contoh :

```
CREATE TABLE PERSONEL
(REGNO    CHAR(10)   NOT NULL,
 NAME     CHAR(45)   NOT NULL,
 ADDRESS  CHAR(45),
 BIRTH    DATE   NOT NULL WITH DEFAULT,
 PRIMARY KEY (REGNO))
```

NULL

Spesifikasi *NULL*, *NOT NULL*, *NOT NULL WITH DEFAULT*

NULL :

dapat diinterpretasikan sebagai nilai yang tidak diketahui atau tidak tersedianya suatu nilai. Null bukan berarti kosong (blank) atau 0 (Nol)

NOT NULL :

pemakai atau program harus memberikan nilai-nilai pada saat memasukkan record

NOT NULL WITH DEFAULT :

nilai *default* disimpan pada saat record dimasukkan tanpa nilai yang ditentukan untuk kolom ini.

Nilai *default*-nya :

Nol	untuk tipe field NUMERIC
Blank	untuk tipe field CHARACTER
CURRENT DATE	untuk tipe field DATE
CURRENT TIME	untuk tipe field TIME

Pada saat membuat tabel, salah satu atribut tersebut di atas dispesifikasikan pada sebuah kolom.

2. CREATE VIEW

Fungsi : membuat tabel view.

View merupakan bentuk alternatif penyajian data dari satu atau lebih tabel. View dapat berisi semua atau sebagian kolom yang terdapat pada tabel dimana kolom tersebut didefinisikan.

Tujuan membuat view :

- Meningkatkan keamanan data
- Meningkatkan kemandirian data
- Penyederhanaan bagi end user (data yang sedikit, nama-nama kolom yang baru dan dapat dibaca dengan lebih baik)

Properti :

- Tidak terdapatnya data tambahan
- View mencakup subset kolom dan / atau baris
- View dapat berisikan data dari beberapa tabel dan / atau tabel-tabel view lainnya
- View dapat berisikan perolehan data, misal : nilai rata-rata
- Manipulasi data melalui view terbatas

Sintaks : **CREATE VIEW viewname (column1, column2, ……..)**
AS SELECT statement FROM tbname
[WITH CHECK OPTION]

Keterangan :

View-name : nama view yang akan dibuat.
Column : nama atribut untuk view
Statement : atribut yang dipilih dari tabel basis data.
Tabel-name : nama tabel basis data.

Contoh :

```
CREATE VIEW VPERSON (REGNO, NAME) AS
    SELECT  REGNO, NAME FROM PAUL.PERSONEL
```

3. CREATE INDEX

Fungsi : membuat index

Sintaks : **CREATE [UNIQUE] INDEX indexname**
ON nama_table (nama_kolom)

Contoh :

```
CREATE UNIQUE INDEX PRSONIDX
    ON PERSONEL(REGNO)
```

Dengan indeks memungkinkan suatu tabel diakses dengan urutan tertentu tanpa harus merubah urutan fisik dari datanya dan dapat pula diakses secara cepat

melalui indeks yang dibuat berdasar nilai field tertentu. Spesifikasi UNIQUE akan menolak key yang sama dalam file.

4. DROP TABLE

Fungsi : menghapus Tabel

Sintaks : **DROP TABLE tbname**

Contoh : DROP TABLE PERSONEL

Dengan perintah itu obyek lain yang berhubungan dengan tabel tersebut otomatis akan dihapus atau tidak akan berfungsi seperti :

- semua record dalam tabel akan terhapus
- index dan view pada tabel akan hilang
- deskripsi tabel akan hilang

5. DROP VIEW

Fungsi : menghapus view

Sintaks : DROP VIEW viewname

Contoh : DROP VIEW VPERSON

6. DROP INDEX

Fungsi : menghapus index

Sintaks : **DROP INDEX indexname**

Contoh : DROP INDEX PRSONIDX

7. ALTER

Fungsi : merubah atribut pada suatu tabel

Sintaks : **ALTER TABLE tbname**
MODIFY (nama_kolom tipe_kolom)
ADD (nama_kolom tipe_kolom [[before, nama_kolom]])
DROP (nama_kolom tipe_kolom)

Contoh : merubah Tabel TABX dengan menambah Field D.

```
ALTER TABLE TABX
ADD D CHAR(3)
```

DATA MANIPULATION LANGUAGE

1. INSERT

Fungsi : menambah baris (record) baru

Sintaks : **INSERT INTO tbname**
(col1, ...) VALUES (value1, ...)

Catatan :

Sintaks tersebut dapat digunakan jika jumlah kolom = jumlah nilai, tetapi jika dalam tabel semua kolom akan diisi dapat digunakan sintaks berikut ini :

Sintaks : **INSERT INTO tbname**
VALUES (value1, value2, ...)

Nilai-nilai diisikan sebanyak kolom yang terdapat di tabel tersebut.

2. UPDATE

Fungsi : merubah record

Sintaks : **UPDATE tbname SET field = ekspresi**
WHERE kondisi

3. DELETE

Fungsi : menghapus record

Sintaks : **DELETE FROM tbname**
WHERE kondisi

4. SELECT

Fungsi : menampilkan record

Sintaks : SELECT [DISTINCT] colname FROM tbname
[WHERE kondisi]
[GROUP BY kondisi]
[HAVING kondisi]
[ORDER BY kondisi]

Contoh Kasus DDL :

- **Membuat Tabel (CREATE TABLE)**

1. CREATE TABLE S
(Sn Char(5) NOT NULL,
Sname Char(20) NOT NULL,
Status Smallint NOT NULL,
City Char(15) NOT NULL);

2. CREATE TABLE P
(Pn Char(6) NOT NULL,
Pname Char(20) NOT NULL,
Color Char(6) NOT NULL,
Weight Smallint NOT NULL);

3. CREATE TABLE SP
(Sn Char(5) NOT NULL,
Pn Char(6) NOT NULL,
QTY INTEGER NOT NULL);

4. CREATE UNIQUE INDEX Sidx ON S(Sn);
   CREATE UNIQUE INDEX Pidx ON P(Pn);
   CREATE INDEX Sdx ON SP(Sn);
   CREATE INDEX Pdx ON SP(Pn);

- **Modifikasi Table P dengan perintah :**

  RENAME COLUMN P.COLOR TO WARNA
  ALTER TABLE P ADD (City CHAR(15) NOT NULL)

- **Membuat View (CREATE VIEW)**

1. Membuat view untuk suplier yang statusnya lebih besar dari 15

```
CREATE VIEW GOOD_SUPPLIERS
        AS SELECT Sn, Status, City FROM S
            WHERE Status > 15;
```

2. Membuat view yang berisi supplier yang tinggal di Paris

```
CREATE VIEW Paris_Suppliers
   AS SELECT * FROM Supliers
        WHERE City = ' Paris '
```

3. Membuat view dengan mengganti nama_atributnya

```
CREATE VIEW Parts (PNum, Part_Name, WT)
   AS SELECT P#, Pname, Weight FROM Part
        WHERE COLOR = 'Red'
```

Contoh Kasus DML :

- **Menambah record (INSERT)**

INSERT INTO S VALUES ('S1','Smith',20,'London');
INSERT INTO S VALUES ('S2','Jones,10,'Paris');
INSERT INTO S VALUES ('S3','Blake',30,'Paris')

Tabel S, P dan SP isikan dengan data-data sebagai berikut :

TABEL S

Sn	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

TABEL P

Pn	Pname	Warna	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

TABEL SP

Sn	Pn	qty
S1	P1	300
S1	P2	200
S1	P	40
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- **Merubah record (UPDATE)**

  1. Merubah data (record) pada tabel P yang mempunyai nomor part P2, warnanya dirubah menjadi Kuning dan beratnya ditambah 5

     ```
     UPDATE P SET Warna  = 'Yellow',
                  Weight = Weight + 5
            WHERE Pn  = 'P2'
     ```

  2. Merubah record pada tabel S, statusnya menjadi dua kali status awal untuk supplier yang bertempat tinggal di kota London

     ```
     UPDATE S SET Status    = 2 * Status
            WHERE City = 'London'
     ```

- **Menghapus record (DELETE)**

  Menghapus record pada tabel S yang nomor supplier-nya S5

  ```
  DELETE FROM S
       WHERE Sn ='S5'
  ```

- **Menampilkan record (SELECT 1 tabel)**

  1. Menampilkan semua data supplier

     ```
     SELECT * FROM S
     ```

     atau

     ```
     SELECT Sn, Sname, Status, City  FROM S
     ```

  2. Menampilkan semua nilai Pn pada tabel SP

     ```
     SELECT Pn FROM SP
     ```

3. Menampilkan nomor supplier dan status untuk supplier yang tinggal di Paris

```
SELECT Sn, Status FROM S
        WHERE City ='Paris'
```

4. Menampilkan no.supplier yang tinggal di Paris dengan status > 20

```
SELECT Sn FROM S
        WHERE City ='Paris" AND Status > 20
```

5. Menampilkan jumlah pengiriman P1

```
SELECT COUNT(*) FROM SP
        WHERE Pn = 'P1'
```

6. Perintah untuk menghindari hasil data yang sama terulang kembali (distinct)

```
SELECT DISTINCT Pn FROM SP
```

7. Menampilkan no.supplier dan status bagi supplier yang tinggal di Paris dalam urutan status menurun

```
SELECT  Sn,Status FROM S
        WHERE City = 'Paris'
                ORDER BY Status desc
```

8. Menampilkan no.Part dari semua part yang dipasok oleh lebih dari seorang supplier

```
SELECT Pn FROM SP
        GROUP BY Pn
                HAVING COUNT(*) > 1
```

9. Menampilkan semua part yang nomornya dimulai dengan huruf C

```
SELECT * FROM P
    WHERE Pname LIKE 'C%'
```

- **Menampilkan record (SELECT lebih dari satu tabel / JOIN)**

1. Menampilkan semua supplier dan part yang keduanya bertempat tinggal pada kota yang sama

```
SELECT Sn, Sname,S tatus, S.City , Pn, Pname, Warna, Weight FROM S,P
    WHERE S.City = P.City
```

2. Menampilkan nama supplier yang memasok barang dengan nomor part P2

```
SELECT Sname FROM S, SP
    WHERE S.Sn = SP.Sn AND SP.Pn = 'P2'
```

3. Menampilkan nama supplier yang memasok part berwarna merah

```
SELECT Sname FROM S, SP, P
    WHERE S.Sn = SP.Sn
        AND SP.Pn = P.Pn
            AND P.COLOR = 'RED'
```

- **Menampilkan record (SELECT lebih dari satu tabel / SELECT Bertingkat)**

1. Menampilkan nama supplier yang memasok barang dengan nomor part P2

```
SELECT Sname FROM S WHERE Sn IN
    (SELECT Sn FROM SP WHERE Pn = 'P2')
atau
SELECT Sname FROM S WHERE Sn = ANY
    (SELECT Sn FROM SP WHERE Pn = 'P2')
```

2. Menampilkan nama supplier yang memasok part berwarna merah

```
SELECT Sname FROM S WHERE Sn IN
    (SELECT Sn FROM SP WHERE Pn IN
        (SELECT Pn FROM P WHERE Warna = 'Red'))
```

3. Menampilkan no.supplier dengan nilai status lebih kecil daripada nilai maksimum status yang ada pada tabel S

```
SELECT Sn FROM S WHERE Status <
    (SELECT  MAX(Status) FROM S)
```

4. Menampilkan nama supplier yang tidak memasok barang dengan nomor part P2

```
SELECT Sname FROM S WHERE Sn NOT IN
    (SELECT Sn FROM SP WHERE  Pn = 'P2')
```

5. Menampilkan semua nomor supplier yang sama lokasinya dengan S1

```
SELECT Sn FROM S WHERE CITY =
    (SELECT CITY FROM S WHERE Sn = 'S1')
```

- **Fungsi Perhitungan**

  COUNT : jumlah baris dan kolom
  SUM : jumlah nilai dam kolom
  AVG : rata - rata nilai dalam kolom
  MAX : nilai terbesar dalam kolom
  MIN : nilai terkecil dalam kolom

  Untuk SUM dan AVG nilainya harus numerik (INT, SMALLINT, FLOAT). Fungsi-fungsi tsb jika dikenakan pada nilai yang NULL maka nilainya akan diabaikan kecuali untuk COUNT(*)

1. Menghitung jumlah supplier
   SELECT COUNT(*) FROM S

   atau
   SELECT COUNT (Sn) FROM S

2. Menampilkan nomor part dan total kuantitas pengiriman dari setiap part

   SELECT Pn, SUM(QTY) FROM SP
   GROUP BY Pn

3. Menghitung jumlah kuantitas dari P2 yang telah disupply

   SELECT SUM (QTY) FROM SP WHERE Pn = 'P2'

4. Menampilkan jumlah pengiriman barang dengan nomor P4 dan dipasok oleh nomor suppplier S1

   SELECT COUNT(*) FROM SP
   WHERE Pn = 'P4' AND Sn = 'S1'

5. Menampilkan nomor part dan total kuantitas dari masing-masing part

   SELECT Pn, SUM(QTY) FROM SP
   GROUP BY P3

DATA CONTROL LANGUAGE

1. GRANT

Fungsi : digunakan untuk memberikan izin akses kepada user

Sintaks : **GRANT privileges ON tbname TO user**

Contoh :

GRANT SELECT ON CLUB TO PUBLIC

GRANT SELECT, INSERT, UPDATE, DELETE ON CLUB TO USER01

2. REVOKE

Fungsi : digunakan untuk mencabut izin akses kepada user

Sintaks : **REVOKE privileges ON tbname FROM user**

Contoh :

REVOKE INSERT, UPDATE, DELETE ON CLUB FROM USER01

REVOKE ALL ON CLUB FROM PUBLIC