# PERKULIAHAN KE 10

# BAHASA QUERY TERAPAN

## 8.1 *Structured Query Language* (SQL)

1. Pengenalan SQL

Bahasa Query Formal yang sudah kita bahas sebelumnya, menjadi dasar dalam pembentukan bahasa query terapan. Dengan bahasa query formal itulah, algoritma dan sintaks dari ekspresi-ekspresi dalam bahasa query terapan disusun. Penguasaan terhadap bahasa query formal akan sanagat memudahkan kita dalam mempelajari dan menguasai pemakaian bahasa query terapan.

Bahasa query yang paling popular tentu saja adalah SQL (*Sturctured Query Language*), karena bahasa ini diakomodasi oleh hampir semua DBMS. Menurut (Fathansyah, 2012) menyampaikan bahwa SQL merupakan bahasa query yang paling banyak digunakan oleh DBMS dan diterapkan dalam berbagai *development tools* dan program aplikasi ketika berinteraksi dengan Basis Data. Bahasa ini dibangun dengan dasar Aljabar Relational dan sedikit Kalkulus Relational.

Menurut (Indrajani, 2009), SQL mudah dipelajari karena merupakan bahasa non procedural, cukup menspesifikasikan informasi apa yang dibutuhkan daripada bagaimana mendapatkannya.

2. SQL Sebagai Subbahasa

Menurut (Ladjamudin, 2004), Penyebuatan SQL sebagai bahasa query sebenarnya tidak tepat sebab kemampuan SQL tidak terbatas hanya untuk query (memperoleh data), tetapi juga mencakup kemampuan seperti : pendefinisian struktur

data, pebgubahan data, pengaturan sekuritasn, dan lain-lain. Terkadang SQL dikatakan sebagai subbahasa data. Adapun alas an SQL dikatanakn sebagai subbahasa data adalah karena SQL tidak mendukung persyaratan bahasa yang lengkap, sekalipun SQL dipakai untuk mengakses basis data. SQL tidak menyediakan hal-hal seperti : pernyataan pengujian kondisi dan pernyataan pengulangan atau iterasi.

Subdivisi SQL:

a. *Data Definition Language* (DDL)

   Query-query ini digunakan untuk mendefinisikan struktur atau skema basis data.

b. *Data Manipulation Language* (DML)

   Query-query ini digunakan untuk manajemen data dalam basis data.

3. Pengelompokkan SQL

*a. Data Definition Language* (DDL)

Menurut (Ladjamudin, 2004), DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, atribut (kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Yang termasuk dalam kelompok DDL ini adalah CREATE, ALTER dan DROP.

1) CREATE

   a) Pembuatan Database

      Sintaks : CREATE DATABASE nama_database

      Nama Database adalah yang dapat mewakili suatu kejadian dapat berupa nama organisasi atau perusahaan.

Contoh : Buat database dengan nama KAMPUS

Query : CREATE DATABASE KAMPUS

b) Pembuatan Tabel

Sintaks : CREATE TABLE nama_table ( nama_kolom1 tipe_data_kolom1, nama_kolom2 tipe_data_kolom2,….)

Keterangan :

(1) Nama tabel merupakan nama tabel yang baru, panjangnya tidak dapat lebih dari 8 karakter, tidak memakai spasi, berisi huruf maupun angka.

(2) Nama Kolom adalah nama untuk kolom yang baru, panjangnya tidak dapat lebih dari 10 karakter, tidak memiliki spasi, berisi huruf dan angka.

(3) Tipe Data adalah jenis data yang nilainya akan dimasukkan dalam kolom yang telah ditentukan.

Contoh :

(a) Char (n) : karakter string sebanyak N karakter, max 254 char, dan isinya ditulis dengan tanda (‘) atau (“)

(b) Integer, digunakan untuk bilangan bulat sebanyak 11 digit

(c) Small int, untuk bilangan angka sebanyak 6 digit

(d) Decimal(p,q), untuk bilnagan angka sebanyak p digit dengan tempat desimal q

(e) Float(x,y), untuk bilangan angka (floating point sebanyak x digit dengan y digit dari tidak desimal)

(4) Sebagai tambahan, setiap kolom pada pendefinisian tabel dapat dilengkapi dengan UNIQUE, NULL, NOT NULL dan NOT UNIQUE.

(a) NULL, menyatakan bahwa nilai kolom bisa tidak diisi (default)

(b) NOT NULL, menyatakan bahwa nilai kolom harus diisi

(c) UNIQUE, menyatakan bahwa nilai pada kolom tidak boleh ada yang sama

(d) NOT UNIQUE, menyatakan bahwa nilai pada kolom boleh kembar

Contoh : Buat struktur tabel dengan nama tabel Mahasiswa dengan data NIM char(8), NAMA char(25), ALAMAT char(30)

Query : CREATE TABLE Mahasiswa (NIM char(8) not null, NAMA char(25) notnull, ALAMAT char(30) notnull)

Perintah ini maksudnya adalah membuat tabel Mahasiswa dimana NIM wajib diisi, NAMA wajib diisi, ALAMAT wajib diisi.

c) Pembuatan Index

Indeks biasa diciptakan dengan tujuan sebagai berikut :

(1) Indeks dapat meningkatkan kinerja

(2) Indeks menjamin bahwa suatu kolom bersifat unik.

Dengan adanya indeks, maka pencarian suatu data yang berdasarkan kolom yang diindeks akan dapat dilakukan dengan cepat. Namun kelebihan ini tentu saja juga dibayar dengan suatu kelemahan. Pengindeksan memperlambat proses penambahan dan penghapusan baris pada tabel, karena saat terjadi penambahan atau penghapusan baris, indeks perlu diperbaharui.

Sintaks : CREATE [UNIQUE] INDEX nama_index ON nama_table (nama_kolom) ;

Keterangan :

(1) Unique adalah pilihan perincian yang dapat digunakan untuk menguatkan nilai data didalam kolom nama index menjadi unik.

(2) Nama_index adalah nama index yang akan diciptakan

(3) Nama_Tabel adalah nama tabel yang berisi kolom index akan dibuat (nama tabel yang akan mengindeks)

(4) Nama_kolom (asc atau dec) adalah nama dari kolom tempat index akan dibuat.(nama kolom untuk mengindeks)

Contoh : Buat index data Mahasiswa berdasarkan NIM dengan nama MHSIDX Dimana NIM tidak boleh sama

Query : CREATE UNIQUE INDEX MHSIDX ON Mahasiswa(NIM)

d) Pembuatan View

Pembuatan View lebih bersifat memanipulasi data daripada pernyataan definisi data, harus menggunakan SELECT untuk mengerjakan perintah ini.

Sintaks : CREATE VIEW nama_view [ (nama_kolom1,….) ] AS SELECT statement [WITH CHECK OPTION] ;

Keterangan :

(1) nama_view harus dimulai dengan huruf, bilangan, dan garis bawah, panjangnya harus kurang dari 9 huruf

(2) Nama_kolom merupakan sebuah nama kolom optimal yang harus diberikan kepada satu kolom view.

(3) Pernyataan SELECT, berupa pernyataan select apa saja kecuali bahwa :

(a) Pernyataan tersebut tidak boleh berisi sebuah klause UNION

(b) Tidak boleh berisi klausa ORDER BY

(c) Tidak boleh berisi klausa SAVE TO TEMP

(4) [With Check Option] adalah klausa optimal yang menyebabkan semua update dan penyisipan ke view akan diperiksa untuk mengetahui apakah semua itu memenuhi definisi view.

Contoh : Buat view dengan nama MHSVIEW yang berisi semua data mahasiswa

Query : CREATE VIEW MHSVIEW AS SELECT * FROM Mahasiswa

2) DROP

a) DROP DATABASE ( menghapus database)

Sintaks:

DROP DATABASE nama_database;

Contoh : Menghapus Database KAMPUS

Query : DROP DATABASE KAMPUS;

b) DROP TABLE ( menghapus tabel)

Sintaks:

DROP TABLE nama_table;

Contoh : Mengahapus Tabel MHS

Query : DROP TABLE MHS

c) DROP INDEX ( menghapus index)

Sintaks:

DROP INDEX nama_index;

Contoh : Menghapus Index MHSIDX

Query : DROP INDEX MHSIDX;

d) DROP View( Menghapus View)

Sintaks:

Drop View nama_view;

Contoh : Menghapus View MHSVIEW

Query : DROP VIEW MHSVIEW

Hasil dari bentuk drop tabel adalah

a) Semua record dalam tabel akan dihapus

b) Index dan view di tabel akan hilang

c) Deskripsi tabel akan hilang.

3) ALTER TABLE

Digunakan untuk merubah struktur dari tabel yang telah dibuat dalam database. Perubahan struktur yang dapat dilakukan adalah menambah kolom baru, merubah nama kolom, merubah tipe data, menambah kunci, mengahapus kolom yang ada.

Sintaks :

ALTER TABLE nama_tabel ADD nama_kolom jenis_kolom

[FIRST | AFTER nama_kolom]

CHANGE [COLUMN] oldnama newnama

MODIFY nama_kolom jenis kolom, …

DROP nama_kolom

RENAME newnama_tabel

Keterangan :

a) ADD digunakan untuk menambah kolom baru

Sintaks :

ALTER TABLE nama_tabel ADD nama_kolom jenis_kolom

[FIRST | AFTER nama_kolom];

FIRST : Penambahan kolom baru diletakkan pada urutan kolom pertama

AFTER : Penambahan kolom baru diletakkan setelah kolom yang ditunjuk

Jika ingin menambah kolom Primary Key maka Sintaksnya :

ALTER TABLE nama_tabel ADD PRIMARY KEY nama_kolom;

b) CHANGE digunakan untuk merubah nama kolom

Sintaks :

ALTER TABLE nama_tabel CHANGE [COLUMN] oldnama newnama ;

c) MODIFY digunakan untuk merubah tipe data kolom

Sintaks :

ALTER TABLE nama_tabel MODIFY nama_kolom jenis kolom ;

d) DROP digunakan untuk menghapus nama kolom

Sintaks :

ALTER TABLE DROP nama_kolom ;

e) RENAME digunakan untuk mengganti nama tabel.

Sintaks :

ALTER TABLE newnama_tabel;

Contoh :

(1) Tambahkan kolom JKEL dengan panjang 1 char pada tabel Mahasiswa

Query : ALTER TABLE Mahasiswa ADD JKEL char(1);

(2) Ubah panjang kolom JKEL menjadi 15 char

Query : ALTER TABLE Mahasiswa MODIFY COLUMN JKEL char(15);

(3) Hapus kolom JKEL dari data table MHS

Query : ALTER TABLE Mahasiswa DROP JKEL;

*b. Data Manipulation Language* (DML)

Menurut (Ladjamudin, 2004), DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan. Yang termasuk dalam kelompok DML adalah SELECT (memilih data), INSERT (menambah data), DELETE (menghapus data) dan UPDATE (menguah data)

1) INSERT

Digunakan untuk penambahan record baru kedalam sebuah tabel.

Sintaks: INSERT INTO Nama_tabel [(nama_kolom1,…)] values (nilai atribut1, …)

Contoh: Masukan data Mahasiswa dengan Nim 10296832, Nama Nurhayati beralamat di Jakarta

Query : INSERT INTO Mahasiswa (Nim, Nama, Alamat) values (“10296832”,”Nurhayati”,“Jakarta”);

2) DELETE

Digunakan untuk menghapus record dari sebuah tabel.

Sintaks: DELETE FROM nama_table WHERE kondisi

Keterangan :

a) nama_tabel : nama tabel yang baris2 nya ingin dihapus

b) WHERE, klausa yang menentukan baris2 yang akan dihapus

Contoh: Hapus data Mahasiswa yang mempunyai NIM “21198002”

Query : DELETE FROM Mahasiswa WHERE NIM=” 21198002”

3) UPDATE

Digunakan untuk mengubah nilai atribut pada suatu record dari sebuah tabel.

Sintaks:

UPDATE nama_tabel SET nama_kolom = value_1 WHERE kondisi ;

Keterangan :

a) nama_tabel adalah nama tabel yang akan di update

b) SET untuk menentukan kolom yang akan diubah dan nilai penggantinya

c) WHERE kondisi adalah klausa yang menetapkan baris2 yang akan di update

Contoh: Ubah alamat menjadi “Depok” untuk mahasiswa yang memiliki NIM “10296832”

Query : UPDATE Mahasiswa SET ALAMAT=”Depok” WHERE NIM=” 10296832”;

4) SELECT

Digunakan untuk menampilkan isi tabel.

Sintaks: SELECT [DISTINCT | ALL] nama_kolom FROM nama_tabel [ WHERE condition ] [ GROUP BY column_list ] [HAVING condition ] [ ORDER BY column_list [ASC | DESC]]

Keterangan :

a) SELECT, memilih data yang akan ditampilkan berdasarkan atribut
b) DISTINCT, menghilangkan duplikasi
c) FROM, mendefinisikan tabel yang akan digunakan dalam query
d) WHERE, menentukan syarat yang akan dipilih
e) GROUP BY, mengelompokkan data yang mempunyai nilai sama
f) HAVING, syarat data yang dikelompokkan digunakan bersama GROUP BY
g) ORDER BY, mengurutkan data

Contoh :

Tabel Mahasiswa

| NIM | NAMA | ALAMAT |
|---|---|---|
| 10296832 | Nurhayati | Jakarta |
| 10296126 | Astuti | Jakarta |
| 31296500 | Budi | Depok |
| 41296525 | Prananigrum | Bogor |
| 50096487 | Pipit | Bekasi |
| 21196353 | Quraish | Bogor |
| 10296001 | Fintri | Depok |
| 21198002 | Julizar | Jakarta |

Tabel Nilai

| NIM | KD_MK | MID | FINAL |
|---|---|---|---|
| 10296832 | KK021 | 60 | 75 |
| 10296126 | KD132 | 70 | 90 |
| 31296500 | KK021 | 55 | 40 |
| 41296525 | KU122 | 90 | 80 |
| 21196353 | KU122 | 75 | 75 |
| 50095487 | KD132 | 80 | 0 |

Tabel MataKuliah

| KD_MK | NAMA_MK | SKS |
|---|---|---|
| KK021 | Sistem Basis Data | 2 |
| KD132 | Sistem Informasi Manajemen | 3 |
| KU122 | Pancasila | 2 |

Gambar 8.1 Contoh Tabel Mahasiswa, Tabel Nilai. Tabel Matakuliah

(1) Tampilkan semua data Mahasiswa

Query : SELECT NIM,NAMA,ALAMAT FROM Mahasiswa;

Atau SELECT * FROM Mahasiswa;

Hasilnya :

| NIM | NAMA | ALAMAT |
|---|---|---|
| 10296832 | Nurhayati | Jakarta |
| 10296126 | Astuti | Jakarta |
| 31296500 | Budi | Depok |
| 41296525 | Prananingrum | Bogor |

(2) Tampilkan Mata Kuliah yang SKS nya 2

Query : SELECT NAMA_MK FROM MataKuliah WHERE SKS=2

Hasilnya :

| NAMA_MK |
|---|
| Sistem Basis Data |
| Pancasila |

(3) Tampilkan semua data nilai dimana nilai MID lebih besar sama dengan 60 atau nilai finalnya lebih besar 75.

Query : SELECT * FROM Nilai WHERE MID >= 60 OR FINAL > 75

Hasilnya :

| NIM | KD_MK | MID | FINAL |
|---|---|---|---|
| 10296832 | KK021 | 60 | 75 |
| 10296126 | KD132 | 70 | 90 |
| 41296525 | KU122 | 90 | 80 |
| 21196353 | KU122 | 75 | 75 |

(4) Menampilkan NIM beserta NAMA yang diuruttkan berdasarkan NAMA

Query : SELECT NIM, NAMA FROM Mahasiswa ORDER BY NAMA;

Hasilnya:

| NIM | NAMA |
|---|---|
| 10296126 | Astuti |
| 31296500 | Budi |
| 10296001 | Fintri |
| 21198002 | Julizar |
| 10296832 | Nurhayati |
| 50096487 | Pipit |
| 41296525 | Prananigrum |
| 21196353 | Quraish |

(5) Mengelompokkan Alamat Mahasiswa

Query : SELECT ALAMAT FROM Mahasiswa GROUP BY ALAMAT;

Hasilnya :

| ALAMAT |
|---|
| Bekasi |
| Bogor |
| Depok |
| Jakarta |

(6) Query : SELECT NAMA, ALAMAT FROM Mahasiswa GROUP BY ALAMAT HAVING COUNT (ALAMAT ) > 1;

Klausa HAVING disediakan untuk mendukung klausa GROUP BY. Kegunaannya adalah untuk menentukan kondisi bagi GROUP BY. Kelompok yang memenuhi kondisi HAVING saja yang akan dihasilkan. Perintah dengan HAVING diatas hanya akan menghasilkan baris untuk ALAMAT yang NAMA mahasiswa lebih dari satu.

Hasilnya :

| ALAMAT |
|---|
| Bogor |
| Depok |
| Jakarta |

c. *Data Access (Data Control Language / DCL)*

Menurut (Ladjamudin, 2004), DCL berisi perintah-perintah untuk mengendalikan pengaksesan data. Pengendalian dapat dilakukan per tabel, per kolom maupun per operasi. Yang termasuk dalam kelompok DCL atau Data Access adalah GRANT (memberi kendali pengaksesan data) dan REVOKE (mencabut kemampuan pengaksesan data)

1) GRANT

Digunakan untuk memberikan hak akses.

Sintaks :

GRANT hak_akses ON nama_db TO nama_pemakai [IDENTIFIED BY] [PASSWORD] ‘Password’ [WITH GRANT OPTION]; atau

GRANT hak_akses ON [nama_db]nama_tabel TO nama_pemakai [IDENTIFIED BY] [PASSWORD] ‘Password’ [WITH GRANT OPTION];

Contoh: Berikan hak akses kepada Adi untuk menampikan nilai final test pada tabel Nilai.

Query : GRANT SELECT (FINAL) ON NILAI TO ADI;

2) REVOKE

Digunakan untuk mencabut kembali hak akses yang sudah diberikan.

Sintaks :

REVOKE hak_akses ON nama_db FROM nama_pemakai ; atau

REVOKE hak_akses ON nama_tabel FROM nama_pemakai ;

Contoh: Tarik kembali dari Adi hak akses untuk menampilkan nilai final test

Query : REVOKE SELECT (FINAL) ON NILAI FROM ADI;

d. *Data Integrity*

Data Integrity merupakan perintah yang digunakan untuk mengembalikan data sebelum terjadi kerusakan.Yang termasuk di dalam kelompok Data Integrity adalah RECOVER TABLE.

Sintaks :

RECOVER TABLE nama_tabel;

Contoh : Kembalikan keadaan data mahasiswa seperti pada saat sebelum terjadi kerusakan

Query : RECOVER TABLE MHS ;

e. *Data Auxiliary*

Data Auxiliary merupakan perintah yang digunakan untuk mengubah data maupun kolom pada tabel. Yang termasuk dalam kelompok Data auxiliary adalah SELECT INTO OUTFILE, LOAD, RENAME TABLE

1) SELECT … INTO OUTFILE ‘filename’

Digunakan untuk mengekspor data dari tabel ke file lain.

Sintaks :

SELECT … INTO OUTFILE ‘Nama File’ [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char'] ] ;

Contoh: Ubah semua data mahasiswa ke bentuk ASCII dan disimpan ke file teks di directory/home/adi dengan pemisah antar kolom ‘|’

Query : SELECT * FROM MHS INTO OUTFILE “/home/adi/teks” FIELDS TERMINATED BY “|”;

2) LOAD

Digunakan untuk mengimpor data dari file lain ke tabel.

Sintaks :

LOAD DATA INFILE “ nama_path” INTO TABLE nama_tabel [ nama_kolom] ; [FIELDS | COLUMNS] [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char'] ];

Contoh: Memasukkan data-data dari file teks yang berada pada direktori "/home/adi" ke dalam tabel MHS_2. Dimana pemisah antara kolom dalam file teks adalah tab (\t) :

Query : LOAD FROM "/home/adi/teks" INTO MHS_2 FILELDS TERMINATED BY '\t';

3) RENAME TABLE

Digunakan untuk mengganti nama tabel.

Sintaks :

RENAME TABLE OldnamaTabel TO NewNamaTabel

Contoh : RENAME TABLE MHS TO MAHASISWA