



SQL : DDL & DML

**PJJ-A11-CF 1234
Konsep Basis Data
(3 SKS)**

URL : www.kulino.dinus.ac.id

Pokok Bahasan

- Apa yang termasuk dalam bahasa SQL?
- Bagaimana queries disajikan dalam SQL? Bagaimana makna suatu query dinyatakan dalam standar SQL?
- Bagaimana SQL menjadikan aljabar dan kalkulus relasional sebagai dasar dan memperluas keduanya?
- Apa yang dimaksud dengan “grouping”? Dan bagaimana ia dapat digunakan dalam operasi-operasi aggregasi?
- Apa yang dimaksud dengan “nested queries”?
- Apa yang dimaksud dengan *null* values ?
- Bagaimana queries dapat digunakan utk menuliskan integrity constraints yang komplek?
- Apa yang dimaksud triggers, dan mengapa triggers menjadi berguna? Dan bagaimana kaitan triggers dan integrity constarints?

Overview

- **Data Manipulation Language (DML):** Subset SQL yang dapat digunakan utk menspesifikasikan queries dan utk menyisipkan (*insert*), menghapus (*delete*), dan memodifikasi (*modify*) baris-baris tabel
- **Data Definition Language (DDL):** Subset SQL utk mendukung pembuatan, penghapusan, dan modifikasi tables & views.
 - *Integrity Constraints (ICs)* dapat didefinisikan pada tables pada saat suatu tabel dibuat atau nanti sesudahnya
- **Triggers & Advanced ICs:** Standar SQL/99 memberikan dukungan untuk *triggers*, yang merupakan tindakan-tindakan otomatis yang dijalankan oleh DBMS bilamana perubahan-perubahan pada database memenuhi kondisi yang dispesifikasi dalam trigger
- **Embedded & Dynamic SQL:** Fitur-fitur *embedded SQL* memungkinkan SQL utk dipanggil dari suatu bahasa induk seperti C atau Java. Sedangkan fitur-fitur *dynamic SQL* memungkinkan sebuah query utk disusun (dijalankan) pada saat “run-time” (Ramakrishnan, Chap 6).

Overview (Con't)

- *Client-Server Execution & Remote Database Access:* Perintah-perintah SQL untuk ini dapat digunakan mengendalikan bagaimana suatu program aplikasi *client* dapat dihubungkan ke sebuah SQLdatabase *server*, atau mengakses data dari sebuah database melalui jaringan (Ramakrishnan, Chap 7)
- *Transaction Management:* Berbagai perintah SQL memungkinkan seorang pengguna utk secara eksplisit mengendalikan aspek bgm sebuah transaksi harus dijalankan (Ramakrishnan, Chap 21)
- *Security:* SQL menyediakan mekanisme utk mengendalikan akses pengguna ke obyek database seperti tables & views (Ramakrishnan, Chap 21)

Membuat Obyek Database

DDL (Data Definition Language)

- Create Tables
- Create Indexes
- Altering Tables
- Dropping Tables/Indexes

DDL : Create Tables Structure

- **Create Tables statement**

CREATE TABLE tabel (A₁ D₁, A₂ D₂, .., A_n D_n)

Keterangan :

tabel = nama (Obyek) tabel yang akan dibuat

A₁..A_n = atribut atau variabel field

D₁, D_n= tipe data untuk A₁..A_n

DDL : Create Tables Statement

- **Membuat tabel siswa**

CREATE TABLE **siswa**

```
( NISN char(6), nama_siswa char(30),  
tgl_lahir date(), tempat_lahir char(30),  
KLS char(2), alamat char(30),  
nama_ortu_wali char(30))
```

DDL: Create Tables Statement

- Membuat tabel Customer

```
CREATE TABLE tblCustomers  
 ( customerID INTEGER NOT NULL,  
 [Last name] CHAR(30) NOT NULL,  
 [First name] CHAR(30) NOT NULL,  
 Phone CHAR(12),  
 Email CHAR (50));
```

Quiz/Tugas

**Buatlah Tabel relasi Misal Kepegawaian :
Pegawai Min 4 Tabel, dengan struktur SQL –DDL,**

Jawab Quiz

- **Buat tabel Pegawai**

CREATE TABLE pegawai

```
( NIP char(6) NOT NULL,  
nama_pegawai char(30) NOT NULL,  
    tgl_lahir date() NOT NULL,  
    tempat_lahir char(30) NOT NULL,  
    Jenis_kelamin char(9) NOT NULL,  
    alamat char(30) NOT NULL,  
    Golongan char(1) NOT NULL )
```

Create Index

- **Index**

Index adalah struktur data eksternal yang digunakan untuk mengurutkan atau mengatur pointer data dalam sebuah table

Peringatan :

Jika kita menggunakan beberapa index pada suatu tabel akan menunjukkan penurunan performa dikarenakan extra overhead dalam pemeliharaan indexnya.

Create Index

- **Index**

Selain itu :

Banyak menggunakan index dapat menyebabkan masalah penguncian record, bila digunakan dalam peralatan multiuser.

Dengan demikian :

Gunakanlah index dalam konteks yang benar, sebuah index dapat memperbaiki performa lebih tinggi sebuah aplikasi.

Create Index Statement

Gunakan pernyataan CREATE INDEX untuk membuat index.

Structure pembuatan index adalah :

```
CREATE INDEX nama_index  
ON nama_tabel (nama_field)
```

Contoh membuat index pada tabel customers dalam database invoicing sebagai berikut :

```
CREATE INDEX idxCustomerID  
ON tblCustomers (CustomerID)
```

Create Index Statement

Pengurutan menggunakan INDEX secara baku data diurutkan dari kecil ke besar. Jika Anda ingin mengurutkan data dari nilai terbesar ke nilai terkecil. Strukturnya adalah :

```
CREATE INDEX nama_index  
ON nama_tabel (nama_field DESC)
```

Contoh membuat index pada tabel customers dalam database invoicing sebagai berikut :

```
CREATE INDEX idxCustomerID  
ON tblCustomers (CustomerID DESC)
```

Create Index Statement

DDL

```
CREATE INDEX idxCustomerID  
ON tblCustomers  
(CustomerID) WITH PRIMARY
```

```
CREATE INDEX idxCustomerName  
ON tblCustomers  
([Last Name], [First Name]) WITH PRIMARY
```

Create Index Statement

DDL

```
CREATE UNIQUE INDEX nama_index  
ON nama_tabel (nama_field)
```

Contoh :

```
CREATE UNIQUE INDEX idxCustomerPhone  
ON tblCustomers (Phone)
```

DROP Index Statement

DDL

```
DROP INDEX nama_index  
ON nama_tabel
```

Contoh :

```
DROP INDEX idxCustomerPhone  
ON tblCustomers
```

DROP Table Statement

DDL

DROP TABLE nama_tabel

Contoh :

DROP TABLE tblCustomers

Altering Tables Statement

DDL

- Altering Tables Structure

ALTER TABLE *tabel*

ADD | MODIFY (A_n d_n, A_n d_n, ..., A_n d_n);

Altering Tables Statement

DDL

```
ALTER TABLE tblCustomers  
ADD COLUMN Address TEXT(30)
```

Mengganti ukuran field :

```
ALTER TABLE tblCustomers  
ALTER COLUMN Address TEXT(40)
```

Altering Tables Statement

DDL

- Altering Tables & DROP

Menghapus Field :

ALTER TABLE tblCustomers

DROP COLUMN Address



Structure Query Language [SQL]

Structure Query Language [SQL]

DML (Data Manipulation Language)

- **Insertion** : Menyisipkan data record ke dalam suatu tabel
- **Updating** : Memperbaiki data record dalam suatu tabel
- **Deletion** : Menghapus data record pada suatu tabel
- **Selection** : Menampilkan data record dari suatu tabel

Query Insertion Structure

INSERT INTO tabel (A₁, A₂, .., A_n)

VALUES (C₁, C₂, ..., C_{n-1}, C_n)

CONTOH :

INSERT INTO siswa (NISN, nm_siswa, nilai)

VALUES (123456, 'Fadhel Muhammad', 89);

Query Selection Structure

SELECT *A₁, A₂, ,A_{n-1}, A_n*

FROM *T₁, T₂, ,T_{n-1}, T_n*

WHERE *Criteria*

GROUP BY *A₁, A₂, ,A_{n-1}, A_n*

HAVING *Criteria_Agregate_function*

ORDER BY *Criteria_A*

Query Update Structure

UPDATE *tabel*

SET *assignments*

WHERE *Criteria*

CONTOH :

UPDATE *siswa*

SET *nilai* = 89

WHERE *NISN* ='123456';

Query Deletion Structure

DELETE FROM *Tabel*

WHERE *Criteria*

CONTOH :

DELETE FROM *siswa*

WHERE *NISN* = '123456';

Structure Query Language [SQL]

CDL (Control Definition Language)

- **GRANT**

Memberikan otoritas (hak akses) manipulasi data pada suatu tabel (database) kepada user

- **REVOKE**

Mencabut otoritas (hak akses) manipulasi data pada suatu tabel (database) dari user

Control Data Language

[CDL]

- **GRANT statement structure**

GRANT <otoritas> ON <nm_tabel> TO <user_name>

Grant Type :

insert , select,
update , delete , all

Control Data Language [CDL]

- **GRANT for insert**

GRANT *insert* **ON** <*nm_tabel*> **TO** <*user_name*>

Contoh :

GRANT *insert* **ON** *siswa* **TO** fadhel

GRANT *insert* **ON** *siswa* **TO** agung, fadhel, septi

GRANT *insert* **ON** *siswa* **TO** all

Control Data Language

[CDL]

- **GRANT for update**

GRANT *update* **ON** <*nm_tabel*> **TO** <*user_name*>

Contoh :

GRANT *update* **ON** *siswa* **TO** fadhel

GRANT *update* (*NIM*, *nmmhs*) **ON** *siswa* **TO** agung, septi

Control Data Language

[CDL]

- **GRANT for select**

GRANT *select* **ON** <nm_tabel> **TO** <user_name>

Contoh :

GRANT *select* **ON** siswa **TO** fadhel

GRANT *insert, select, update* **ON** siswa **TO** septi

GRANT *all* **ON** siswa **TO** ani

Control Data Language [CDL]

- **GRANT for all and public**

GRANT *all* **ON** <*nm_tabel*> **TO** <*user_name*>

GRANT <*otoritas*> **ON** <*nm_tabel*> **TO** *public*

Contoh :

GRANT *all* **ON** *siswa* **TO** *ani*

GRANT *select* **ON** *siswa* **TO** *public*

Control Data Language [CDL]

- REVOKE structure

REVOKE <otoritas> ON <nm_tabel> FROM <user_name>

Revoke Type :

insert , select,
update , delete , all

Control Data Language [CDL]

- **REVOKE for insert**

REVOKE *insert* ON <nm_tabel> FROM <user_name>

Contoh :

REVOKE insert ON siswa FROM ruben

REVOKE insert ON siswa FROM public

Control Data Language [CDL]

- **REVOKE for select**

REVOKE *select* **ON** <*nm_database*> **FROM** <*user_name*>

Contoh :

REVOKE *select* **ON** *siswa* **FROM** *septi*

REVOKE *insert, select* **ON** *siswa* **FROM** *fadhel*

REVOKE *select, delete* **ON** *siswa* **FROM** *public*

Advanced Query

- Complex Integrity Constraints
 - Constraints over single table
 - Domain constraints
- ICs over several tables
- IF conditional into query
- Aggregate function

Integrity Constraint

Model Relasional

Menjaga integritas atau satu kesatuan data dalam suatu database, gunakan kunci utama (Primary key) dan kunci tamu (Foreign key).

– Primary key :

Adalah atribut kunci yang dapat menunjukkan identitas informasi dari atribut yang bersangkutan.

– Foreign key :

Adalah atribut kunci milik relasi utama yang disisipkan pada relasi transaksi untuk menunjukkan relationship antara relasi transaksi dengan relasi utama.

Integrity Constraint

Model Relasional

Tujuan utama :

Primary key dan Foreign key digunakan untuk relationships antar relasi.

Untuk menjaga integritas data antar relasi keduanya.

Relationships

Model Relasional



NIM	NAMA	JURUSAN
12345	AGUS SANTOSO	DKP
12346	DIAN KURNIA	TKJ
12347	MARIMAR	TI

NIM	KODE+MK	NILAI
12345	TKJ-01	A
12346	TKJ-01	B
12347	TIF-03	B

Data Integrity

Model Relacional

- Introduction to Data Integrity
- Overview of Integrity Constraints
- Types of Integrity Constraints
- The Mechanisms of Constraint Checking
- Deferred Constraint Checking
- Constraint States

Data Integrity

Model Relasional

– Introduction to Data Integrity

Mendefinisikan himpunan aturan integritas data adalah penting, pendefinisiannya dilakukan oleh database administrator atau application developer. Sebagai contoh data integrity, dengan pertimbangan tables employees and departments dengan business rules untuk informasi pada setiap table sebagai ilustrasi sbb :

Data Integrity

Model Relasional

Figure 21-1 Examples of Data Integrity

EMPNO	EMPNAME	SALES	COMM	DEPNO
6666	MULDER	6500.00		20
7665	SMITH	9000.00		20
9876	ALLEN	7500.00	100.00	30
1234	WARDS	5000.00	200.00	30
1345	JONES	2975.00	400.00	30

DEPNO	DNAME	LOC
20	RESEARCH	DALAS
30	SALES	CHICAGO



Data Integrity

Model Relasional

Types of Data Integrity

Bagian ini menggambarkan aturan yang dapat diterapkan pada kolom tabel yang menekankan perbedaan tipe data pada integritas data.

- **Null Rule**

Aturan null adalah definisi aturan pada single column yang membolehkan atau tidak membolehkan inserts atau updates untuk pengisian rows kosong (the absence of a value) pada kolom ini.

- **Unique Column Values**

Aturan nilai unique didefinisikan pada sebuah column (or set of columns) yang membolehkan insert or update hanya pada row jika itu berisi sebuah nilai unique dalam sebuah kolom column (or set of columns).

Data Integrity

Model Relasional

Types of Data Integrity

- **Primary Key Values**

Aturan nilai primary key didefinisikan pada sebuah key (a column or set of columns) tertentu bahwa setiap each row dalam table dapat mengidentifikasi keunikan dengan nilai kunci tersebut

- **Referential Integrity Rules**

Aturan referential integrity adalah definsi aturan pada sebuah kunci key (a column or set of columns) dalam sebuah table yang menjamin bahwa data dalam kunci cocok dengan nilai dalam sebuah relasi table (the referenced value).

Data Integrity

Model Relacional

- Introduction to Data Integrity
- Overview of Integrity Constraints
- Types of Integrity Constraints
- The Mechanisms of Constraint Checking
- Deferred Constraint Checking
- Constraint States

Data Integrity

Model Relacional

- Introduction to Data Integrity
- Overview of Integrity Constraints
- Types of Integrity Constraints
- The Mechanisms of Constraint Checking
- Deferred Constraint Checking
- Constraint States

Data Integrity

Model Relacional

- Introduction to Data Integrity
- Overview of Integrity Constraints
- Types of Integrity Constraints
- The Mechanisms of Constraint Checking
- Deferred Constraint Checking
- Constraint States

Data Integrity

Model Relacional

- Introduction to Data Integrity
- Overview of Integrity Constraints
- Types of Integrity Constraints
- The Mechanisms of Constraint Checking
- Deferred Constraint Checking
- Constraint States

Integrity Constraint

Model Relacional

- Primary key

ALTER TABLE **tblCustomers**

ALTER COLUMN CustomerID INTEGER CONSTRAINT PK **tblCustomers**
PRIMARY KEY

ALTER TABLE **tblCustomers**

ALTER COLUMN CustomerID INTEGER PRIMARY KEY

Integrity Constraint

Model Relacional

- Primary key

```
ALTER TABLE tblCustomers  
ADD CONSTRAINT CustomerNames  
UNIQUE ([Last Name], [First Name])
```

```
ALTER TABLE tbllnvoices  
ADD CONSTRAINT CheckAmount  
CHECK (Amount > 0)
```

Integrity Constraint

Model Relacional

- Foreign key

```
ALTER TABLE tblShipping  
ADD CONSTRAINT FK_tblShipping  
FOREIGN KEY (CustomerID)  
REFERENCES tblCustomers (CustomerID)  
ON UPDATE CASCADE  
ON DELETE CASCADE
```

Advanced Query

- Complex Integrity Constraints
 - Constraints over single table
 - Domain constraints

Advanced Query

- ICs over several tables

Advanced Query

- IF conditional into query

Advanced Query

- Aggregate function

Query Optimization

- **What Is Optimization?**
- Optimization adalah pemilihan proses eksekusi SQL statement yang efficient. Langkah ini sangat penting di dalam processing untuk data manipulation language (DML) statement:
SELECT,
INSERT,
UPDATE,
or DELETE.

Query Optimization

- Merupakan Proses Penyeleksian yg paling optimum dan paling efisien dlm query evolution plan
- Tugas utama query optimization adalah mencari atau menemukan rencana yg baik untuk mengevaluasi ekspresi
- Terdapat beberapa cara untuk mengakses SQL statement yang sering ada, contoh, dengan variasi tahapan pengaksesan data dalam tabel atau indexes. Pada procedure Oracle menggunakan eksekusi sebuah statement sangat berpengaruh pada kesepatan eksekusi statement.

Query Optimization

- Bagian ini pada Oracle disebut perhitungan *optimizer* cara yang sangat efficient untuk sebuah SQL statement. Beberapa faktor evaluasi optimizer dapat dipilih dari beberapa access paths alternatif . Hal ini dapat menggunakan cost-based or rule-based approach

Query Optimization

- Rule based optimization adalah sebuah method lama bahwa optimizer dapat menggambarkan hasil bagaimana query and dieksekusi.
- The Cost based optimizer lebih optimizer "intelligent" yang dapat mencari dan menentukan bagaimana QUERY anda dieksekusi.

Query Optimization

- Rule based optimization

Rank	Access Path
1	Single row by rowid
2	Single row by cluster join
3	Single row by hash cluster key with unique or primary key
4	Single row by unique or primary key
5	Cluster join
6	Hash cluster key
7	Indexed cluster key
8	Composite key
9	Single-column indexes

Query Optimization

- Rule based optimization

Rank	Access Path
10	Bounded range search on indexed columns
12	Unbounded range search on indexed columns
13	Sort-merge join
14	MAX or MIN of indexed column
15	ORDER BY on indexed columns
16	Full table scan

Query Optimization

- Rule based optimization : Single row by rowid

This Access path ini hanya tersedia jika statement's WHERE clause dipilih identitas baris atau dengan menambahkan dukungan tulisan SQL CURRENT OF CURSOR oleh Oracle Precompilers. Menjalankan statement, Oracle accesses tabel perbaris.

Contoh :

diberikan statement untuk mengakses data sebagai berikut :

```
SELECT * FROM emp  
WHERE ROWID = 'AAAA7bAA5AAA1UAAA';
```

The EXPLAIN PLAN output for this statement might look like this:

OPERATION	OPTIONS	OBJECT_NAME
SELECT STATEMENT		
TABLE ACCESS	BY ROWID	EMP

Query Optimization

- Rule based optimization : Single Row by Cluster Join

This Access path ini hanya tersedia jika statement's WHERE clause dipilih identitas baris atau dengan menambahkan dukungan tulisan SQL CURRENT OF CURSOR oleh Oracle Precompilers. Menjalankan statement, Oracle accesses tabel perbaris.

Contoh :

diberikan statement untuk mengakses data sebagai berikut :

```
SELECT * FROM emp  
WHERE ROWID = 'AAAA7bAA5AAA1UAAA';
```

The EXPLAIN PLAN output for this statement might look like this:

OPERATION	OPTIONS	OBJECT_NAME
SELECT STATEMENT		
TABLE ACCESS	BY ROWID	EMP

Query Optimization

- Jika saat tujuan optimizer anda untuk dipilih, optimizer akan digunakan secara baku CBO tidak menghilangkan statistics anda. Bila berjalan dalam mode CBO, ini sangat penting untuk keakuratan statistik pada database (ANALYZE TABLE, or DBMS_STATS.Gather*).

Query Optimization

- Relational Query Optimization
- Optimization query Planning
- Catalog system in query optimization
- Indexes using for query optimization

Query Optimization

- Optimization query Planning

memperlihatkan representasi execution plan pada SQL statement berikut, untuk memilih name, job, salary, dan department name untuk semua employees yang tidak menerima gaji pada batas rentang gaji yang telah ditentukan :

```
SELECT ename, job, sal, dname  
FROM emp, dept  
WHERE emp.deptno = dept.deptno  
AND NOT EXISTS  
(SELECT * FROM salgrade  
WHERE emp.sal BETWEEN losal AND hisal);
```

Query Optimization

- Relational Query Optimization (mohon dilengkapi sendirii di fileke 2 ya...)

Query Optimization

- Optimization query Planning (mohon dilengkapi sendirii di fileke 2 ya)

Query Optimization

- Catalog system in query optimization (sama lengkapi dewe....

Query Optimization

- Indexes using for query optimization (sama lengkapi dewe....)

Database Trigger

- Trigger Type
- Using Database trigger
- The activate rule database trigger
- Deletion database trigger

Trigger Type

- **INSERT**

adalah event_trigger untuk penyisipan data record ke dalam tabel

- **UPDATE**

adalah event_trigger untuk perbaikan data record yang ada dalam tabel

- **DELETE**

adalah event_trigger untuk menhapus data record yang ada dalam tabel

Trigger Statement

```
CREATE OR REPLACE TRIGGER nm_trigger
    [BEFORE | AFTER] event_trigger ON nama_tabel
    FOR EACH ROW
    [DECLARE]
        Deklarasi_Variabel
    BEGIN
        statements list;
    END;
```

Database Trigger

- Using Database trigger

MENGUPDATE DATA BARANG :

```
CREATE OR REPLACE TRIGGER tr_barang
  AFTER update ON barang
  FOR EACH ROW
  BEGIN
    dbms_output.put_line('Tabel barang telah di-
update');
  END;
```

Database Trigger

MENAMBAH DATA STOK BARANG :

```
CREATE OR REPLACE TRIGGER tr_insert_pasok
  AFTER insert ON pasok
  FOR EACH ROW
  BEGIN
    update stok_barang
    set jum_stok= jum_stok + :new.jum_pasok ;
  END;
```

MENGHAPUS DATA STOK BARANG :

```
CREATE OR REPLACE TRIGGER tr_delete_pasok
  AFTER update ON pasok
  FOR EACH ROW
DECLARE
  Jml_lama integer;
  Jml_baru integer;
  Selisih integer;
BEGIN
  select:old.jum_stok,:new.jum_pasok into jml_lama,jml_baru from pasok;
  if (jml_lama>jml_baru) then
    selisih := jml_lama - jml_baru;
    update stok_barang
    set jum_stok = jum_stok - selisih;
  else
    selisih := jml_baru - jml_lama;
    update stok_barang
    set jum_stok = jum_stok + selisih;
  end if;
END;
```

Database Trigger

- The activate rule database trigger

*Tambah Contoh
Sendiri.....*

Rangkuman

- SQL menjadi faktor yang penting pada saat awal penerimaan model relasional, karena lebih natural dibandingkan bahasa-bahasa database sebelumnya yang bersifat prosedural (*procedural query languages*), seperti aljabar relasional.
- SQL lengkap secara rasional; bahkan dalam kenyataannya memiliki daya ekspresi yang signifikan dibandingkan aljabar relasional
- Bahkan, queries yang dapat dinyatakan dalam aljabar relasional seringkali dpt dinyatakan dengan lebih natural dalam SQL.

Tugas : ?