- Data Types
- Data Definition Language
- Referential Constraint
- SQL Query
- Data Manipulation Language

Data Types

- Untuk setiap kolom/field yang terdapat pada sebuah tabel, harus ditentukan pula tipe datanya yang menentukan jangkauan nilai yang bisa diisikan
- Masing-masing DBMS memiliki jenis dan nama tipe data tersendiri. Bisa jadi ada yang khusus (tidak ada pada yang lain), atau diberi dengan nama lain, atau bahkan tidak memiliki tipe data yang standar

Data Types

1. CHAR = mendefinisikan string sepanjang n karakter.
   Bila n tidak disertakan, maka panjang karakter adalah 1.

2, VARCHAR = mendefinisikan string yang panjangnya bisa berubah-ubah sesuai dengan kebutuhan, namun string tersebut dibatasi sebanyak n karakter. Oracle merekomendasikan varchar2.

3. VARCHAR2(n) = mendefinisikan string yang panjangnya bisa berubah-ubah sesuai dengan kebutuhan, namun string tersebut dibatasi sebanyak n karakter.
   Maksimun karakter pada varchar2 adalah 2000 karakter.

4. LONG = mendefinisikan tipe data binary, maksimum 2 Giga Byte disimpan dalam format internal Oracle.

Data Types

5. LONG RAW = sama dengan Long yaitu mendefinisikan tipe data binary maksimum 2 Giga Byte, tidak dikonversi oleh Oracle(data mentah apa adanya).

6. DATE = mendefinisikan tanggal, menyimpan tahun, bulan, hari, jam, menit, dan detik.

7. NUMBER = mendefinisikan angka pecahan, baik fixed decimal ataupun floating point. Nilai n adalah jumlah bytes total dan p adalah presisi angka debelakang koma.

- Data Types
- Data Definition Language
- Referential Constraint
- SQL Query
- Data Manipulation Language

DDL

Data Definition Language (DDL) berkaitan dengan perintah-perintah untuk pendefinisian obyek-obyek basis data. Diantaranya digunakan untuk Table, View, dan Trigger :

Perintah yang digunakan antara lain :

- Create : untuk membuat objek
- Alter : untuk melakukan perubahan struktur
- Drop : untuk menghapus objek
- Constraint : untuk memberikan batasan

DDL Untuk Tabel (1)

Pembuatan tabel, syntax:

```
CREATE TABLE <namatabel> (
      <kolom1>  <tipedata1>  [<aturan1>],
      <kolom2>  <tipedata2>  [<aturan2>],
      ...
      <kolomn>  <tipedatan> [<aturann>],
      [<aturanTabel>]
      )
```

- **[aturan$_n$] berisi aturan untuk field ybs, bersifat opsional. Biasanya berupa:**

  NOT NULL → artinya field tersebut harus ada isinya

  DEFAULT <*nilai*> → artinya field tersebut jika tidak diisi nilainya, maka nilai defaultnya adalah <*nilai*>

DDL Untuk Tabel (2)

- [aturanTabel] berisi aturan-aturan yang berlaku untuk tabel tersebut. Jika aturan lebih dari satu, maka dibatasi oleh tanda koma. Aturan tabel biasanya:

  PRIMARY KEY (<DaftarKolomKey>)

  FOREIGN KEY (<daftarKolomForeignKey>) REFERENCES <namaTabel> (<daftarKolom>)

  [ON DELETE <aturanDelete>] [ON UPDATE <aturanUpdate>])

- [AturanTabel] bisa diberi nama. Jika diberi nama, maka syntaxnya sbb:

  CONSTRAINT <namaAturan> <AturanTabel>

DDL Untuk Tabel (3)

Pembuatan tabel pasok dengan primary key ‘kode_pasok’ dan constraint foreign key ‘kode_suplier’ pada tabel suplier. Default menyatakan nilai pada field jika tidak diisi oleh user. Pada field keterangan jika tidak diisi maka datanya adalah ‘Barang Masuk’. Contoh:

```
CREATE TABLE pasok(
kode_pasok char(5),
kode_suplier char(5),
tanggal_pasok date,
jumlah_pasok number(5),
keterangan varchar2(30) DEFAULT 'Barang Masuk',
CONSTRAINT PK_pasok PRIMARY KEY (kode_pasok),
CONSTRAINT FK_pasok_suplier FOREIGN KEY (kode_suplier)
REFERENCES suplier(kode_suplier) ON DELETE CASCADE )
```

DDL Untuk Tabel (4)

Menghapus tabel, syntax:

```
DROP TABLE <namaTabel>
```

Modifikasi tabel:

1. Menambahkan kolom baru:

```
ALTER TABLE <namaTabel>    add <namakolom>
<tipedata> <aturan>
```

2. Menghapus kolom:

```
ALTER TABLE <namaTabel>    drop <namakolom>
```

DDL Untuk View (1)

View adalah tabel bayangan. Tidak menyimpan data secara fisik. Biasanya berupa hasil query dari tabel-tabel dalam sebuah database

Syntax:

```
CREATE VIEW <namaTabel>  AS
<SQLQuery>
```

DDL Untuk View (2)

Contoh

Membuat View dengan nama Mahasiswa Pria:

View

```
CREATE VIEW MahasiswaPria AS
SELECT * FROM Mahasiswa WHERE jeniskel="L"
```

DDL Untuk TRIGGER (1)

- Trigger adalah sebuah obyek dalam database yang berupa prosedur yang merespon setiap kali terdapat proses modifikasi pada tabel
- Proses modifikasi berupa: Insert, Update dan delete

Syntax:

```
CREATE TRIGGER <namaTrigger> ON TABLE
<namaTabel> FOR [DELETE] [,] [INSERT] [,]
[UPDATE] AS <perintahSQL>
```

DDL Untuk TRIGGER (2)

Contoh

Membuat trigger dg nama *tLogUbahNilai* untuk setiap penambahan / update data pada tabel Pesertakul, dilakukan penambahan data pada tabel LogHistoris

Trigger

```
CREATE TRIGGER tLogUbahNilai ON TABLE pesertakul
FOR UPDATE, INSERT
AS
INSERT INTO LogHistoris (tanggal, proses) VALUES
(getDate(), 'Terjadi proses perubahan data nilai')
```

- Data Types
- Data Definition Language
- Referential Constraint
- SQL Query
- Data Manipulation Language

Referential Integrity Constraint

Aturan untuk Update → berlaku pada proses **modifikasi di parent table**

- Cascade → Pembaruan sebuah baris data diikuti dengan pembaruan baris data pada child table yang terelasikan.
- Restrict → mencegah proses pembaruan data jika terdapat baris data di *child table* yang terelasikan.
- Ignore → mengabaikan referensi. Boleh memperbarui data pada *parent*, tapi tidak memperbarui data yang berelasi pada *child table*.

Referential Integrity Constraint

Aturan untuk Delete → berlaku pada proses **modifikasi di parent table**

- Cascade → Menghapus seluruh baris data pada child table yg terelasikan.
- Restrict → mencegah penghapusan jika terdapat baris data yang berelasi pada child table.
- Ignore → mengabaikan referensi. Boleh menghapus data, dan tidak ada efeknya bagi child table.

Referential Integrity Constraint

Aturan untuk Insert

- Restrict → Tidak boleh menambah data pada child table, jika nilai yang dimasukkan pada kolom yang berelasi tidak terdapat pada parent tabelnya.
- Ignore → mengabaikan referensi. Boleh menambah data pada child, walaupun nilai yang dimasukkan pada kolom yang berelasi tidak terdapat pada parent tabel.

Referential Integrity Constraint

Contoh

```
CREATE TABLE Mahasiswa
(nim CHAR(10),
nama CHAR(20),
nip  CHAR(10)
PRIMARY KEY  (nim),
FOREIGN KEY (nip)
REFERENCES Dosen
ON DELETE CASCADE
)
```

- Data Types
- Data Definition Language
- Referential Constraint
- SQL Query
- Data Manipulation Language

SQL Query

Berikut adalah syntax dari SQL-SELECT

```
SELECT [DISTINCT] select_list
FROM table_source
[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC |
DESC] ]
```

select

Adalah perintah untuk menampilkan data atau hasil dari proses query .

a. Menampilkan keseluruhan field

Untuk menampilkan keseluruhan field dari tabel-tabel yang didefinisikan, digunakan *. Misalnya, untuk menampilkan seluruh field dari tabel Anggota, perintahnya:

```
SELECT * FROM Anggota
```

select

b. Menampilkan kolom-kolom tertentu

Kolom-kolom yang dipilih berupa **ekspresi,** yang mana ekspresi tersebut bisa berupa:

- field tabel
- operasi dan fungsi

Antara kolom satu dengan lainnya **dipisahkan dengan tanda koma (,).**

select

Contoh

- menampilkan nama dan alamat Anggota

```
SELECT nama, alamat FROM Anggota
```

- menampilkan nama dalam bentuk huruf kapital:

```
SELECT UPPER(nama), alamat FROM Anggota
```

select

c. Menampilkan data tertentu dengan klausa "where"

```
SELECT id, nama, alamat FROM pegawai
WHERE id = 002
```

- Data Types
- Data Definition Language
- Referential Constraint
- SQL Query
- Data Manipulation Language

MANIPULASI DATA

- Data Manipulation Language (DML) merupakan bahasa basis data yang berguna untuk melakukan modifikasi dan pengambilan data pada suatu basis data

- Modifikasi data terdiri dari: penambahan (insert), pembaruan (update) dan penghapusan (delete).

Penambahan Data (insert)

- Instruksi SQL untuk melakukan penambahan data adalah menggunakan syntax:

```
INSERT INTO <namaTabel> [(field1, field2, …)]
VALUES (field1 [,field2, …]) | SQL-SELECT
```

Keterangan

- **<namaTabel>** → nama tabel yang akan ditambahkan datanya
- **[(field1, field2, …)]** →field-field di dalam tabel yang akan diisikan nilainya
- **VALUES** (nilai1 [,nilai2, …]) **| SQL-SELECT** → nilai yang diisikan

Jika mengisikan sebuah data tunggal saja yang tidak diambil dari tabel lain, gunakan:

```
VALUES (nilai1 [,nilai2, …])
```

Penambahan Data (insert)

Contoh

Untuk mengisikan data pada tabel pegawai (id, nama, alamat, telp):

```
INSERT INTO pegawai
VALUES (001, 'sugeng','Jl tlogomas 245','0341-51234')
```

Contoh di atas tidak menyertakan klausa [(field1, field2, ...)], sehingga pengisiannya harus seluruh field dan urutannya harus benar sesuai dengan urutan field pada struktur tabel.

Penambahan Data (insert)

Contoh

Untuk mengisikan data pada tabel pegawai:

```
INSERT INTO pegawai(id,nama)
VALUES (001, 'sugeng')
```

Contoh di atas menyebutkan field-field yang diisikan pada tabel pegawai , sehingga nilai-nilai yang ditulis setelah klausa VALUES juga harus mengikuti field-field tersebut.

Mengubah Data (update)

- Instruksi SQL untuk melakukan perubahan data adalah menggunakan syntax:

```
UPDATE <namaTabel>
SET <field1>=<nilai1> [ , <field2> = <nilai2>, …]
[WHERE <kondisi>]
```

Keterangan

- **<namaTabel>** → nama tabel yang akan ditambahkan datanya
- **SET <field1>=<nilai1>** [,**<field2>=<nilai2>,...**] → nilai baru yang akan diisikan pada field tertentu
- [**WHERE** <kondisi>] → filter yang berlaku untuk menentukan data mana saja yang diupdate

Mengubah Data (update)

Contoh

- Untuk melakukan update massal (berlaku untuk seluruh field), yakni menaikkan semua gaji sebesar 50% pada pegawai:

```
UPDATE pegawai SET gaji= gaji + (gaji*0.5)
```

- Untuk melakukan update tertentu, misal menaikkan gaji 50% hanya untuk pimpinan, dimana id pimpinan = 001

```
UPDATE pegawai SET gaji= gaji + (gaji*0.5)
WHERE id = 001
```

Menghapus Data (delete)

- Instruksi SQL untuk menghapus data adalah menggunakan syntax:

```
DELETE FROM <namaTabel>
[WHERE <kondisi>]
```

Keterangan

- **<namaTabel>** → nama tabel yang akan ditambahkan datanya
- [**WHERE** <kondisi>] → filter yang berlaku untuk menentukan data mana saja yang dihapus

Menghapus Data (delete)

Contoh

- Untuk menghapus seluruh data pegawai:

```
DELETE FROM pegawai
```

- Untuk menghapus seluruh pegawai yang memiliki umur >60 tahun

```
DELETE FROM pegawai WHERE umur > 60
```