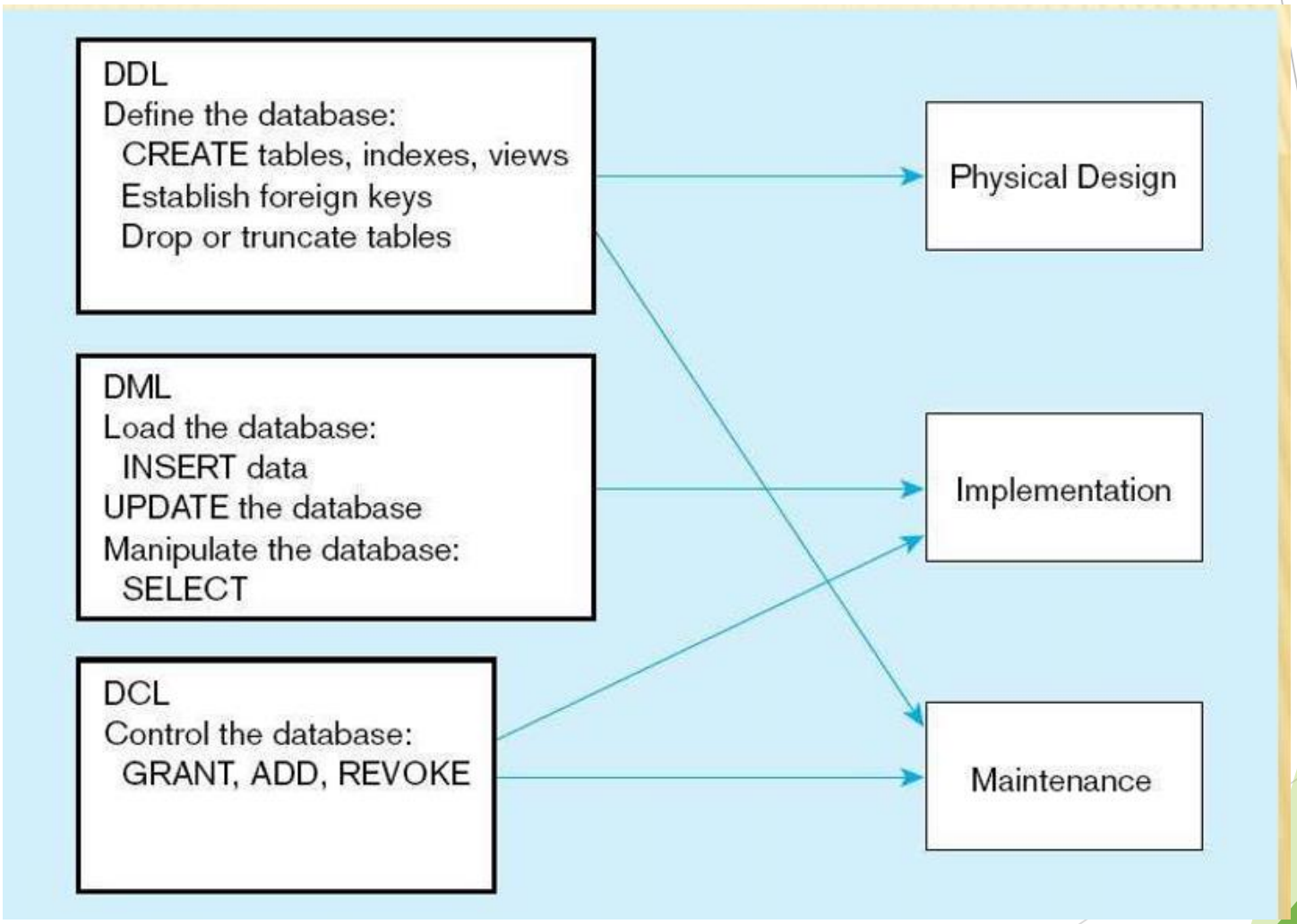
# STRUCTURE QUERY LANGUAGE

DDL & DML

# OUTLINE

- Data Definiton Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

# Proses Pembangunan Database



# Perintah SQL

**DDL - Data Definition Language**

| Command | Description |
|---|---|
| CREATE | Creates a new table, a view of a table, or other object in the database. |
| ALTER | Modifies an existing database object, such as a table. |
| DROP | Deletes an entire table, a view of a table or other objects in the database. |

**DML - Data Manipulation Language**

| Command | Description |
|---|---|
| SELECT | Retrieves certain records from one or more tables. |
| INSERT | Creates a record. |
| UPDATE | Modifies records. |
| DELETE | Deletes records. |

**DCL - Data Control Language**

| Command | Description |
|---|---|
| GRANT | Gives a privilege to user. |
| REVOKE | Takes back privileges granted from user. |

# DATA DEFINITION LANGUAGE (DDL)

# Overview

- DDL merupakan bagian dari perintah SQL untuk membuat, memodifikasi atau menghapus struktur basis data

# Objects

- Table
- View → merupakan sebuah **tabel semu / tabel lojik**, dimana datanya berasal dari satu/ lebih tabel lain sebagai tabel sumber. View biasa dibuat untuk memudahkan user menampilkan data
- User digunakan untuk **mendefinisikan user beserta passwordnya**.
- Index digunakan untuk **mempercepat pengaksesan data** pada suatu tabel. Index dapat diberikan pada satu atau lebih kolom.

# Jenis DDL

Secara Umum, ada beberapa jenis DDL :

- Create
- Modifikasi (Alter)
- Drop
- Rename
- Truncate
- Comment

# Constraints

Beberapa constraints yang sering digunakan :

- Primary Key
- Foreign Key
- Unique Key
- Null / Not Null

# CONSTRAINT

- **Constraint** adalah pemaksaan aturan yang dilakukan pada level tabel. Terdapat beberapa constraint yang dapat digunakan yaitu:
  - **Primary key**
  - **Foreign Key**
  - **Not Null**
  - **Unique**

# PRIMARY KEY

- Secara Implisit Primary key membentuk keunikan dan **NOT NULL.**
- **Hanya ada satu primary key yang diperbolehkan** untuk setiap tabel.
- Pada sebuah Primary Key tidak boleh diberikan constraint Unique tetapi diperbolehkan menggunakan NOT NULL.

```
CREATE TABLE SUPERVISOR(
        nip char(6) not null CONSTRAINT supervisor_pk PRIMARY KEY,
        nmsupervisor varchar2(20),
        thn_masuk number
    );
CREATE TABLE SUPERVISOR(
        nip char(6) not null enable,
        nmsupervisor varchar2(20),
        thn_masuk number,
        CONSTRAINT supervisor_pk PRIMARY KEY (nip)
    );
```

# FOREIGN KEY

- Biasanya sebuah **foreign key pada suatu tabel merupakan primary key tabel yang lain.**
- Constraint ini bertujuan untuk menetapkan suatu kolom atau kombinasi dari beberapa kolom menjadi foreign key dari sebuah tabel.
- Foreign key sering disebut sebagai ***referential integrity constraint***.

```
CREATE TABLE PRODUK  (
        idproduk char(6) not null,
        nmproduk varchar2(20),
        stok number,
        harga number,
        detail varchar2(50),
        spesifikasi varchar2(50),
        idkategori char(6),
        CONSTRAINT produk_pk PRIMARY KEY (idproduk) ENABLE,
        CONSTRAINT produk_fk FOREIGN KEY (idkategori) REFERENCES kategori
(idkategori)
        ON DELETE CASCADE
  );
```

Nama tabel referensi

# NOT NULL

- Jika sebuah kolom pada database tidak boleh kosong, maka constraint **NOT NULL** diberikan pada kolom tersebut.

```
CREATE TABLE SUPERVISOR(
        nip  char(6)  NOT  NULL  CONSTRAINT  supervisor_pk  primary
    key,
        nmsupervisor varchar2(20) NOT NULL,
        thn_masuk number
    );
```

# UNIQUE

- Merupakan constraint yang memiliki sifat yang hampir sama dengan **primary key**, yaitu harus memiliki **nilai yang berbeda untuk setiap baris pada satu kolom yang sama.**
- Constraint unique mengijinkan adanya nilai **NULL(kosong).**

```
CREATE TABLE MEMBER
(
        idmember char(6),
        idmembership char(6),
        username varchar2(20),
        password varchar2(20),
        nmmember varchar2(20),
        telp varchar2(13),
        email varchar2(20),
        alamat varchar2(30),
        kota varchar2(20),
        provinsi varchar2(20),
        kodepos varchar2(6),
        nokartu char(7),
        tgldaftar date,
        tglexpired date,
        CONSTRAINT member_pk PRIMARY KEY (idmember),
        CONSTRAINT  member_fk1  FOREIGN  KEY  (idmembership)
references  membership (idmembership) on delete cascade,
        CONSTRAINT member_uq UNIQUE(nokartu)
);
```

# CREATE

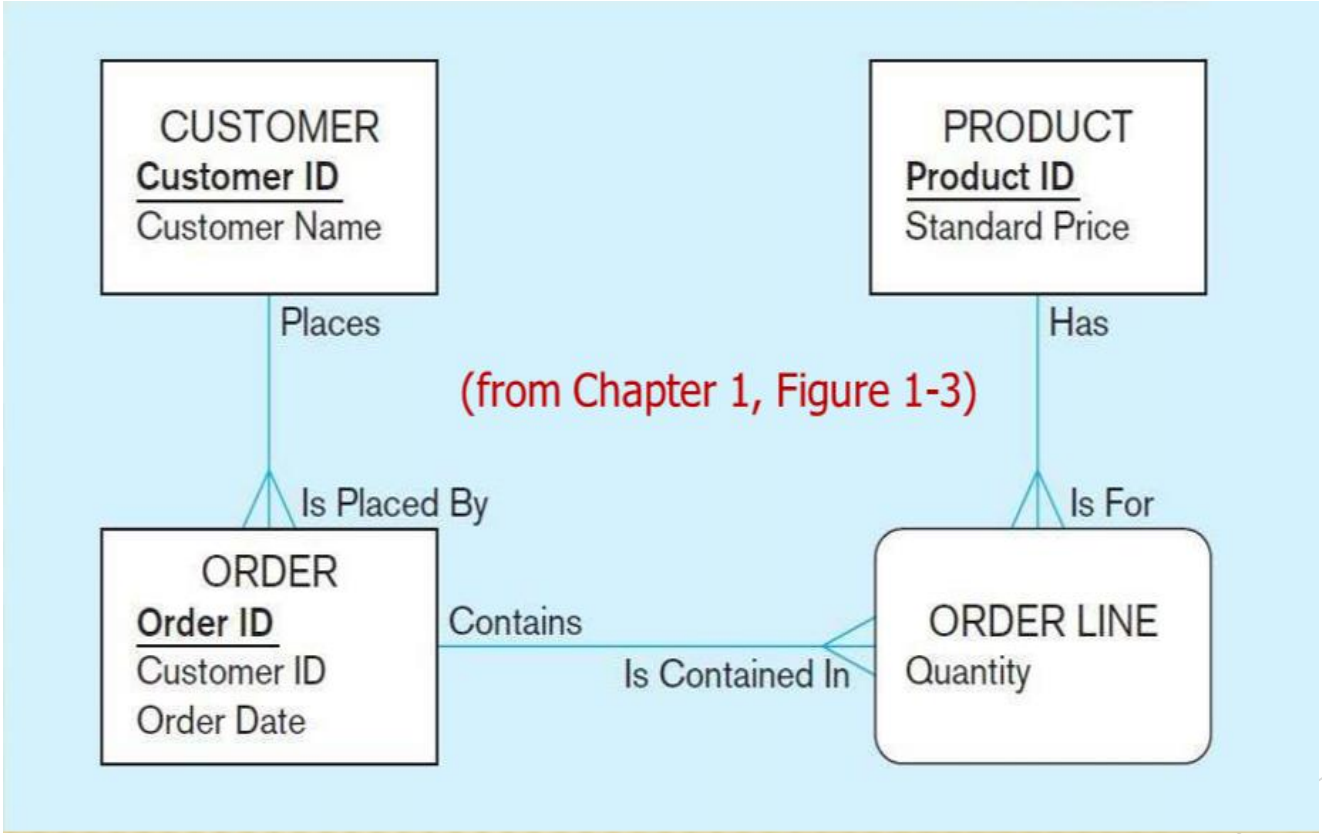
- Table
- View
- User
- index

```
CREATE TABLE tablename
( {column definition   [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
        {domain name | datatype [(size)] }
        [column_constraint_clause . . .]
        [default value]
        [collate clause]

and table constraint ::=
        [CONSTRAINT constraint_name]
        Constraint_type [constraint_attributes]
```

# Contoh Skema Relasi



# Penggunaan Perintah Create

```
CREATE TABLE Customer_T
                (CustomerID                 NUMBER(11,0)      NOT NULL,
                 CustomerName               VARCHAR2(25)      NOT NULL,
                 CustomerAddress            VARCHAR2(30),
                 CustomerCity               VARCHAR2(20),
                 CustomerState              CHAR(2),
                 CustomerPostalCode         VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
```

```
CREATE TABLE Order_T
                (OrderID                    NUMBER(11,0)      NOT NULL,
                 OrderDate                  DATE DEFAULT SYSDATE,
                 CustomerID                 NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));
```

```
CREATE TABLE Product_T
                (ProductID                  NUMBER(11,0)      NOT NULL,
                 ProductDescription         VARCHAR2(50),
                 ProductFinish              VARCHAR2(20)
                        CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                                 'Red Oak', 'Natural Oak', 'Walnut')),
                 ProductStandardPrice       DECIMAL(6,2),
                 ProductLineID              INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

```
CREATE TABLE OrderLine_T
                (OrderID                    NUMBER(11,0)      NOT NULL,
                 ProductID                  INTEGER           NOT NULL,
                 OrderedQuantity            NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T(OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T(ProductID));
```

Overall table definitions

# ALTER

- Table (add, modify,drop, enable/disable)
- View (untuk mengkompilasi ulang)
- Index (modifikasi alokasi penyimpanan)
- User ( mengubah password)

# Penggunaan Perintah Alter

```
ALTER TABLE table_name alter_table_action;
```

Table action :

```
ADD [COLUMN] column_definition
ALTER [COLUMN] column_name SET DEFAULT default-value
ALTER [COLUMN] column_name DROP DEFAULT
DROP [COLUMN] column_name [RESTRICT] [CASCADE]
ADD table_constraint
```

```
ALTER TABLE CUSTOMER_T
ADD COLUMN CustomerType VARCHAR2 (2) DEFAULT "Commercial";
```

```
ALTER TABLE member
DROP COLUMN telp;
```

# DROP

- Table
- View
- Index
- User

```
DROP TABLE member CASCADE CONSTRAINT;

DROP VIEW view_data_transaksi;

DROP INDEX nokartu;
```

# Penggunaan Perintah Lainnya

- RENAME

  →mengubah nama tabel, view, sequence dan synonym

  **RENAME** nama lama **TO** nama baru;

  **RENAME** member **TO** pelanggan;

- TRUNCATE

  →menghapus seluruh isi tabel

  **TRUNCATE TABLE** [schema.] nama table;

  **TRUNCATE TABLE** member;

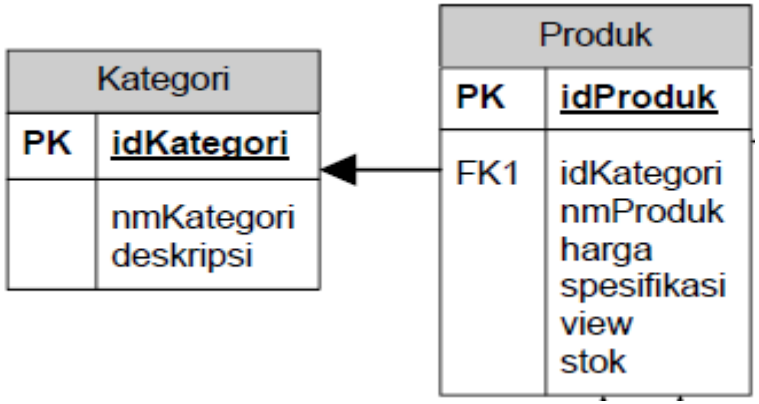
- COMMENT

  → menambahkan komentar pada tabel/view

  **COMMENT ON** TABLE [schema.] nama table **IS** 'isi komentar'

  **COMMENT ON TABLE** membership **IS** 'berisi jenis jenis keanggotaan yang tersedia dengan besaran diskon tertentu';

# LATIHAN




| idKategori | nmKategori | deskripsi |
|---|---|---|
| KAT001 | Kategori 1 | Minuman |
| KAT002 | Kategori 2 | Makanan |
| KAT003 | Kategori 3 | ATK |
| KAT004 | Kategori 4 | Tools |

Buat tabelnya menggunakan script DDL!!

| idProduk | idKategori | nmProduk | harga | spesifikasi | view | stok |
|---|---|---|---|---|---|---|
| P001 | KAT001 | Fanta Red | 6000 | 600ml | Fanta Red | 5 |
| P002 | KAT001 | Le Minera | 5000 | | Le Minera | 4 |
| P003 | KAT002 | Silverque | 7500 | | Silverque | 6 |
| P004 | KAT002 | Indomie | 4500 | | Indomie | 8 |
| P005 | KAT002 | beng beng | 2000 | small | beng beng | 2 |
| P006 | KAT002 | top | 2000 | | top | 6 |
| P007 | KAT003 | Pensil | 2500 | 2B | Pensil | 4 |
| P008 | KAT003 | Pulpen | 3500 | | Pulpen | 7 |
| P009 | KAT003 | Spidol | 5000 | BBG | Spidol | 5 |
| P010 | KAT004 | Gunting | 7000 | | Gunting | 3 |

# DATA MANIPULATION LANGUAGE (DML)

# Overview

- **Data Manipulation Language** merupakan bahasa yang memungkinkan pengguna untuk **mengakses** dan **mengubah data** yang sesuai dengan model datanya
- DML juga merupakan konsep yang menerangkan bagaimana menambah, mengubah dan menghapus baris tabel

# Perintah-perintah DML

- **Insert**

  **Sintak : Insert Into Nama table Values ( , )**

  **Contoh : INSERT INTO** penyewa **values** ('82900','22-APR-2009','meeting');

- **Update**

  **Sintak : Update Nama Table**

  **Set Atribut = IsiAtributBaru (Value baru)**

  **Where Kondisi**

  **Contoh : UPDATE** reservasi

  **Set** kegunaan='resepsi pernikahan'

  **Where** id_reservasi='82900';

- **Delete**

  **DELETE FROM** reservasi **WHERE** id_reservasi='82900';

- **TRUNCATE TABLE** reservasi;

# Perintah Dasar Transaksi

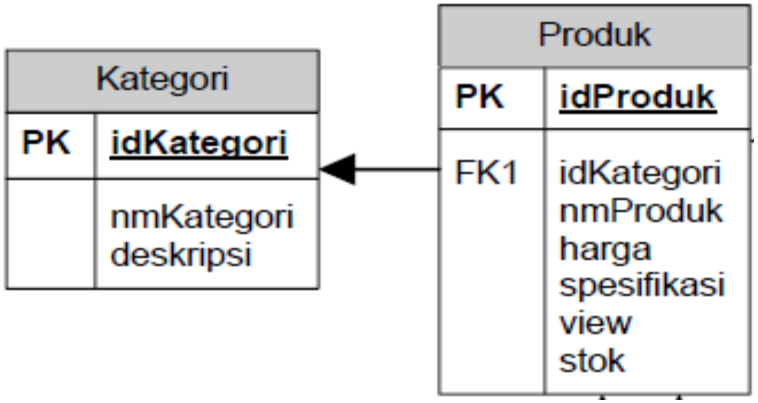
- **Commit**

  Perintah ini berfungsi untuk mengakhiri suatu transaksi yang telah dirubah menggunakan perintah DML. Data-data akan bersifat permanen setelah menggunakan perintah “commit”. Jika pengguna tidak menggunakan perintah ini, maka masih dimungkinkan untuk mengembalikan (undo) semua modifikasi terakhir kali.

- **Rollback**

  Data-data yang telah dirubah dengan perintah DML masih bisa dikembalikan ke kondisi awal transaksi. Pengembalian tersebut dapat menggunakan perintah rollback.

# LATIHAN




| idKategori | nmKategori | deskripsi |
|---|---|---|
| KAT001 | Kategori 1 | Minuman |
| KAT002 | Kategori 2 | Makanan |
| KAT003 | Kategori 3 | ATK |
| KAT004 | Kategori 4 | Tools |

| idProduk | idKategori | nmProduk | harga | spesifikasi | view | stok |
|---|---|---|---|---|---|---|
| P001 | KAT001 | Fanta Red | 6000 | 600ml | Fanta Red | 5 |
| P002 | KAT001 | Le Minera | 5000 | | Le Minera | 4 |
| P003 | KAT002 | Silverque | 7500 | | Silverque | 6 |
| P004 | KAT002 | Indomie | 4500 | | Indomie | 8 |
| P005 | KAT002 | beng beng | 2000 | small | beng beng | 2 |
| P006 | KAT002 | top | 2000 | | top | 6 |
| P007 | KAT003 | Pensil | 2500 | 2B | Pensil | 4 |
| P008 | KAT003 | Pulpen | 3500 | | Pulpen | 7 |
| P009 | KAT003 | Spidol | 5000 | BBG | Spidol | 5 |
| P010 | KAT004 | Gunting | 7000 | | Gunting | 3 |

Isi tabelnya menggunakan script DML!

# Basic Query

# Basic SELECT Statement

```
SELECT    *|{[DISTINCT] column|expression [alias],...}
FROM      table;
```

- SELECT digunakan untuk mengidentifikasi kolom / atribut
- FROM digunakan untuk mengidentifikasi table / relasi

# Menampilkan Seluruh Kolom/Atribut

```
SELECT *
FROM   departments;
```

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 50 | Shipping | 124 | 1500 |
| 60 | IT | 103 | 1400 |
| 80 | Sales | 149 | 2500 |
| 90 | Executive | 100 | 1700 |
| 110 | Accounting | 205 | 1700 |
| 190 | Contracting | | 1700 |

8 rows selected.

# Menampilkan Kolom/Atribut tertentu

```
SELECT department_id, location_id
FROM   departments;
```

| DEPARTMENT_ID | LOCATION_ID |
|---|---|
| 10 | 1700 |
| 20 | 1800 |
| 50 | 1500 |
| 60 | 1400 |
| 80 | 2500 |
| 90 | 1700 |
| 110 | 1700 |
| 190 | 1700 |

8 rows selected.

# Arithmetic Expressions

Operator yang dapat digunakan adalah sebagai berikut

| Operator | Description |
|---|---|
| + | Add |
| - | Subtract |
| * | Multiply |
| / | Divide |

# Contoh Penggunaan Operator Aritmatika

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

| LAST_NAME | SALARY | SALARY+300 |
|---|---|---|
| King | 24000 | 24300 |
| Kochhar | 17000 | 17300 |
| De Haan | 17000 | 17300 |
| Hunold | 9000 | 9300 |
| Ernst | 6000 | 6300 |
| ... | | |
| Hartstein | 13000 | 13300 |
| Fay | 6000 | 6300 |
| Higgins | 12000 | 12300 |
| Gietz | 8300 | 8600 |

20 rows selected.

# Duplikasi Records

Tampilan awal query adalah seruh record yang memenuhi kondisi, termasuk record yang sama atau terduplikasi untuk mengeliminasinya dapat menggunakan perintah **DISTINCT dalam klausa SELECT**

```
SELECT department_id
FROM   employees;
```

| DEPARTMENT_ID |
|---|
| 90 |
| 90 |
| 90 |
| 60 |
| 60 |
| 60 |
| 50 |
| 50 |
| 50 |

...

```
SELECT DISTINCT department_id
FROM   employees;
```

| DEPARTMENT_ID |
|---|
| 10 |
| 20 |
| 50 |
| 60 |
| 80 |
| 90 |
| 110 |

# Menampilkan Record Tertentu

- Untuk dapat menampilkan record sesuai dengan kondisi tertentu dapat menggunakan klausa WHERE

- Klausa WHERE mengikuti setelah klausa FROM

```
SELECT    *|{[DISTINCT] column|expression [alias],...}
FROM      table
[WHERE    condition(s)];
```

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | DEPARTMENT_ID |
|---|---|---|---|
| 100 | King | AD_PRES | 90 |
| 101 | Kochhar | AD_VP | 90 |
| 102 | De Haan | AD_VP | 90 |

# Operator Pembanding

| Operator | Meaning |
|---|---|
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| <> | Not equal to |

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000;
```

| LAST_NAME | SALARY |
|---|---|
| Matos | 2600 |
| Vargas | 2500 |

# Operator Pembanding Lainnya

| Operator | Meaning |
|---|---|
| `BETWEEN ...AND...` | Between two values (inclusive), |
| `IN(set)` | Match any of a list of values |
| `LIKE` | Match a character pattern |
| `IS NULL` | Is a null value |

# Penggunaan BETWEEN

BETWEEN digunakan untuk menampilkan record yang berada dalam jangkauan suatu nilai. Contohnya sebagai berikut :

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500;
```

**Lower limit** **Upper limit**

| LAST_NAME | SALARY |
|---|---|
| Rajs | 3500 |
| Davies | 3100 |
| Matos | 2600 |
| Vargas | 2500 |

# Penggunaan IN

IN digunakan untuk menampilkan keanggotaan suatu himpunan nilai

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

| EMPLOYEE_ID | LAST_NAME | SALARY | MANAGER_ID |
|---|---|---|---|
| 202 | Fay | 6000 | 201 |
| 200 | Whalen | 4400 | 101 |
| 205 | Higgins | 12000 | 101 |
| 101 | Kochhar | 17000 | 100 |
| 102 | De Haan | 17000 | 100 |
| 124 | Mourgos | 5800 | 100 |
| 149 | Zlotkey | 10500 | 100 |
| 201 | Hartstein | 13000 | 100 |

8 rows selected.

# Penggunaan LIKE

- Operator LIKE dan NOT LIKE digunakan untuk mencari suatu nilai bertype string dengan membandingkan susunan karakternya.
- Bentuk umum :

  WHERE nama_kolom LIKE nilai_pembanding

  WHERE nama_kolom NOT LIKE nilai_pembanding

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%';
```

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%';
```

# Penggunaan NULL

Untuk menampilkan record bernilai kosong

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```

| LAST_NAME | MANAGER_ID |
|---|---|
| King | |

# Operator Logika

| Operator | Meaning |
|---|---|
| `AND` | Returns `TRUE` if *both* component conditions are true |
| `OR` | Returns `TRUE` if *either* component condition is true |
| `NOT` | Returns `TRUE` if the following condition is false |

# Penggunaan Operator AND

**AND membutuhkan kedua kondisi yang benilai benar**

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|---|---|---|---|
| 149 | Zlotkey | SA_MAN | 10500 |
| 201 | Hartstein | MK_MAN | 13000 |

# Penggunaan Operator OR

**OR membutuhkan salah satu kondisi yang benilai benar**

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|---|---|---|---|
| 100 | King | AD_PRES | 24000 |
| 101 | Kochhar | AD_VP | 17000 |
| 102 | De Haan | AD_VP | 17000 |
| 124 | Mourgos | ST_MAN | 5800 |
| 149 | Zlotkey | SA_MAN | 10500 |
| 174 | Abel | SA_REP | 11000 |
| 201 | Hartstein | MK_MAN | 13000 |
| 205 | Higgins | AC_MGR | 12000 |

8 rows selected.

# Penggunaan Operator NOT

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

| LAST_NAME | JOB_ID |
|---|---|
| King | AD_PRES |
| Kochhar | AD_VP |
| De Haan | AD_VP |
| Mourgos | ST_MAN |
| Zlotkey | SA_MAN |
| Whalen | AD_ASST |
| Hartstein | MK_MAN |
| Fay | MK_REP |
| Higgins | AC_MGR |
| Gietz | AC_ACCOUNT |

10 rows selected.

# Klausa ORDER BY

- Mengurutkan record dapat menggunakan klausa ORDER BY
  - ASC: ascending order, default
  - DESC: descending order
- Klausa ORDER BY diletakkan terakhir setelah pernyataan SELECT

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

| LAST_NAME | JOB_ID | DEPARTMENT_ID | HIRE_DATE |
|---|---|---|---|
| King | AD_PRES | 90 | 17-JUN-87 |
| Whalen | AD_ASST | 10 | 17-SEP-87 |
| Kochhar | AD_VP | 90 | 21-SEP-89 |
| Hunold | IT_PROG | 60 | 03-JAN-90 |
| Ernst | IT_PROG | 60 | 21-MAY-91 |

...

20 rows selected.

# Perintah SQL yang berkaitan dengan operasi pada aljabar relasional

- Kartesian Produk

  Contoh : Menampilkan data dari table  borrower x loan

  SQL  : Select * from borrower, loan

- Join

  Contoh : “Tampilkan nama, loan number dan loan amount dari semua customer”.

  SQL  : select borrower.customer-name, loan.loan-number, loan.loan-amount
  from borrower, loan
  where borrower.loan-number = loan.loannumber

# Perintah SQL yang berkaitan dengan operasi pada aljabar relasional (lanj.)

- Operasi Himpunan
- Operasi himpunan pada SQL meliputi : union, intersect, dan except. Union identik dengan U, intersect identik dengan ∩ dan except identik dengan – pada aljabar relasional.
- Setiap operasi tersebut secara otomatis menghilangkan duplikasi
- Untuk mempertahankan duplikasi gunakan union all, intersect all dan except all
  a. Operator UNION atau UNION ALL digunakan untuk menggabungkan hasil dua buah Query atau lebih dalam satu tampilan. Contoh : "Untuk mendapatkan semua customer yang mempunyai pinjaman, rekening atau keduanya pada bank"
     SQL : (SELECT customer name FROM depositor)UNION (SELECT customer-name FROM borrower
  b. Operator INTERSECT digunakan untuk memperoleh data hasil irisan dari dua buah Query atau lebih. Contoh : "Untuk mendapatkan semua customer yang memiliki pinjaman dan rekening pada bank"
     SQL : (SELECT distinct customer-name FROM depositor) INTERSECT (SELECT distinct customer-name FROM borrower)

# Perintah SQL yang berkaitan dengan operasi pada aljabar relasional (lanj.)

c. Operator EXCEPT

Digunakan untuk memperoleh data hasil Query kiri yang tidak terdapat pada hasil Query kanan. Contoh : “Untuk mendapatkan semua customer yang mempunyai sebuah rekening tetapi tidak memiliki pinjaman pada bank”

SQL : (SELECT distinct customer-name FROM depositor) EXCEPT (SELECT customer-name FROM borrower)

# Selesai