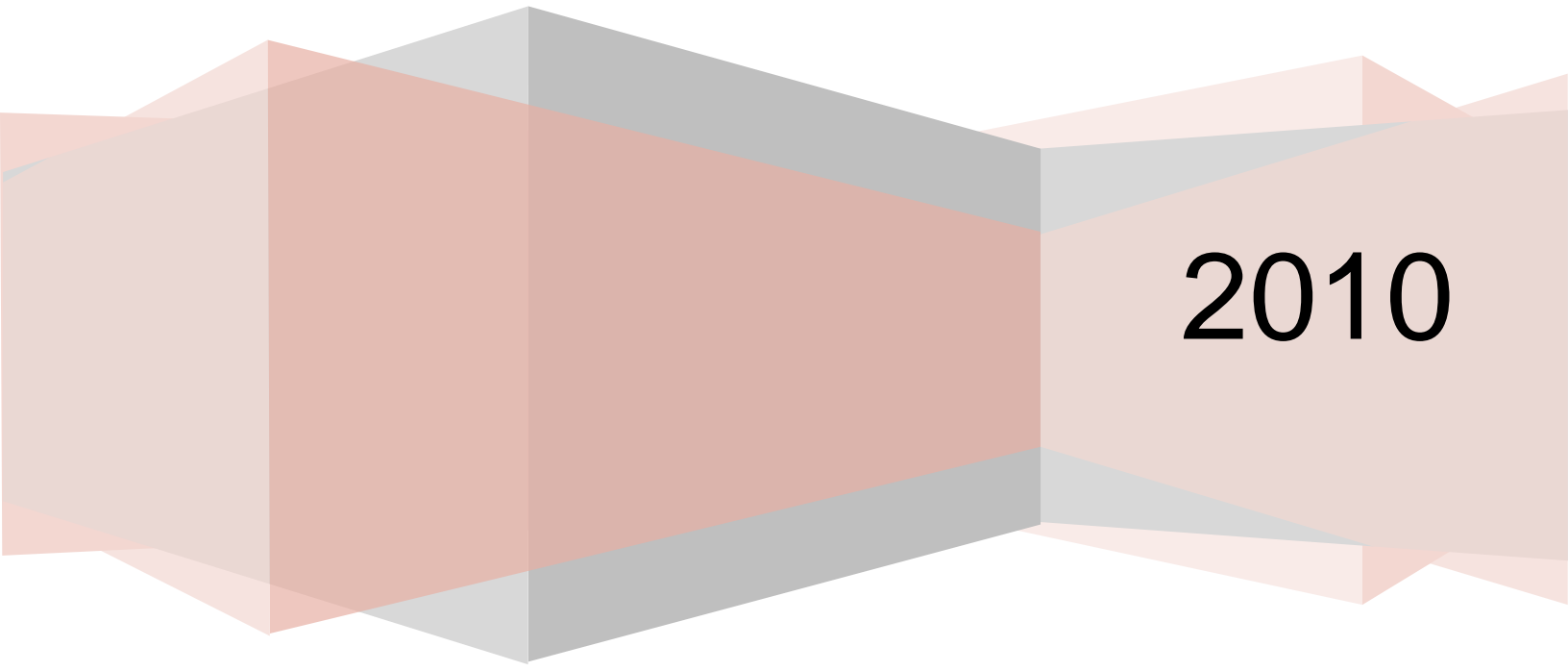


# BASIS DATA

The Oracle logo, consisting of the word "ORACLE" in a bold, red, sans-serif font, with a registered trademark symbol (®) to the upper right of the "E". It is centered within a light gray rectangular box.

**ORACLE®**

An abstract geometric design at the bottom of the page, featuring overlapping translucent shapes in shades of red, orange, and gray, creating a complex, crystalline structure.

**2010**

# Sesion I (Pengenalan Oracle)

## I. Pengenalan Database Oracle XE

Oracle Database Express Edition (Oracle Database XE) adalah produk database server yang bersifat Freeware dari Oracle Corp. Dengan produk ini, para pemakai Oracle XE dapat mempergunakannya tidak hanya untuk percobaan, tapi juga dapat digunakan untuk pengembangan deployment system.

Dengan Oracle XE, Anda dapat menggunakan interface browser untuk :

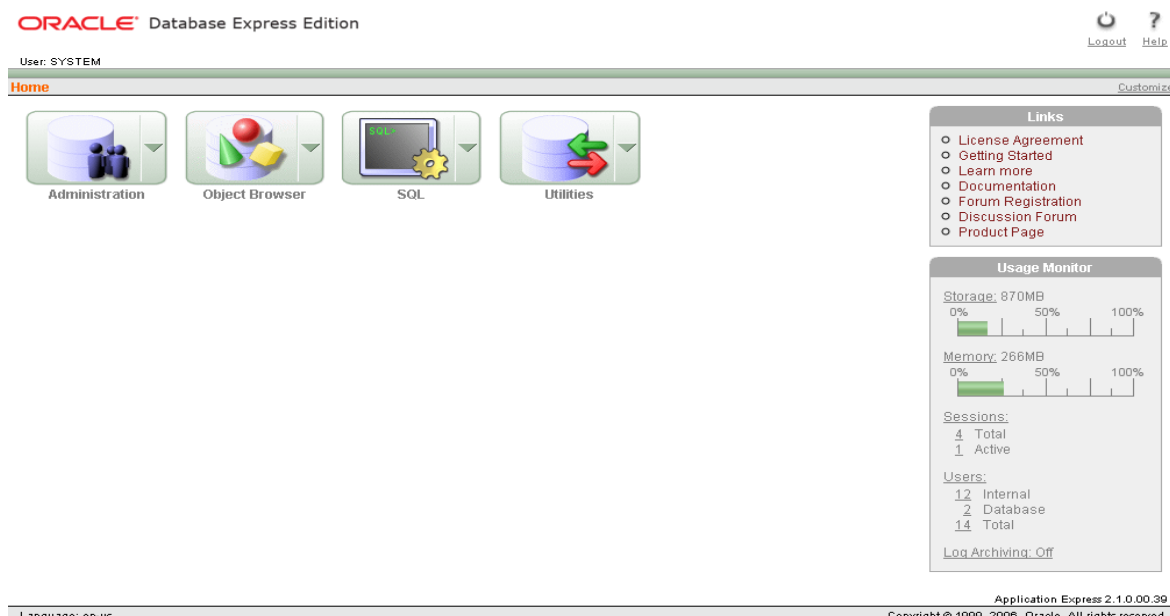
- Administrasi Database
- Membuat table, view, dan object database lainnya
- Menjalankan Query dan SQL Scripts
- Membuat Report

### I.I Login Sebagai Database Administrator

Pertama – tama Anda perlu login sebagai Oracle Database XE Administrator. Ikuti langkah – langkah berikut ini :

1. Buka Database Homepage pada Browser kalian.
  - Pada Start menu pilih → All Programs → Oracle Database 10g Express Edition → Go To Database Home Page.
  - Atau dapat juga dengan Menuliskan Alamat <http://127.0.0.1:8080/apex/> pada url browser kalian
2. Pada menu login isi informasi berikut :
  - Username : **system**
  - Password : (password yang di buat di saat Oracle XE pertama kali di install atau tanyakan kepada Asisten Lab)
3. Klik Login

Tampilan Oracle Database XE home page



## I.II Unlocking Akun Sample User

Untuk membuat application, kamu perlu untuk login sebagai User database. Oracle Database XE menyediakansample atau contoh user database yang di sebut **HR**. User ini memiliki beberapa skema table database dengan yag bisa di gunakan untuk membuat aplikasi untuk Human Resource Department (HRD). Namun pada kondisi default akun ini tidak aktif atau terkunci. Untuk menggunakannya kita perlu meng unlocknya terlebih dahulu

Untuk unlock sample user account **HR** :

1. Pastikan Anda login sebagai Oracle Database XE Administrator pada sesi sebelumnya.
2. Klik **Administration** icon, dan kemudian klik **Database Users**.
3. Klik **HR** schema icon untuk menampilkan informasi tentang **HR**.



HR

4. Isi dengan Informasi berikut :
  - **Password** dan **Confirm Password**: masukan **hr** sebagai password.
  - **Account Status**: pilih **Unlocked**.
  - **Roles**: pastikan kedua pilhan **CONNECT** dan **RESOURCE** di enable..
5. klik **Alter User**.

Sekarang Anda telah siap membuat aplikasi pertama kalian dengan Oracle Database XE

## II. Membuat Aplikasi Sederhana

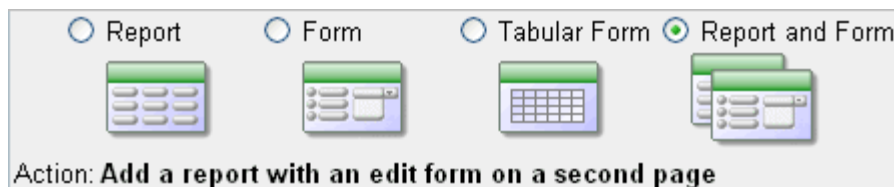
Membuat aplikasi adalah cara untuk mempermudah untuk menampilkan dan mengedit data Anda. Untuk membuat Aplikasi kali ini berdasarkan table EMPLOYEES, yang merupakan bagian dari skema **HR**.

Untuk membuat aplikasi berdasarkan table EMPLOYEES :

1. Pada Databse Homepage, klik **Application Builder**
2. Klik **Create** button.
3. Di bawah Create Application, pilih **Create Application** dan klik **Next**.
4. Di bawah Create Application:
  - a. **Name**: (nama aplikasi saya).
  - b. Biarkan yang lainnya pada kondisi default..
  - c. Klik **Next**.

Berikutnya, add pages untuk aplikasi kamu.

5. Di bawah Add Page:
  - a. Untuk Select Page Type, pilih **Report and Form**.



Perlu di ingat **Action** menggambarkan tipe halaman yang Anda buat.

- b. Berikutnya pada **Table Name** field, klik gambar panah ke atas, kemudian pilih **EMPLOYEES** dari jendela pencarian
- c. klik **Add Page**.

Dua halaman akan di buat dan di tampilkan di halaman dan di bawah Create Application seperti ini .:

Create Application					Cancel	< Previous	Next >
Page	Page Name	Page Type	Source Type	Source			
1	EMPLOYEES	Report	Table	EMPLOYEES			
2	EMPLOYEES	Form	Table	EMPLOYEES			

- d. klik **Next**.
6. Pada tab panel , pada kondisi default (**One Level of Tabs**) dan klik **Next**.
7. Pada Shared Components panel, pada kondisi default (**No**) dan klik **Next**.  
Ops ini memungkinkan kamu untuk mengimport shared component dari aplikasi lain. Shared Component adalah element umum yang bisa di tampilkan atau di pakai pada setiaopo halaman aplikasi.
8. Untuk Authentication Scheme, Language, dan User Language Preference Derived From, biarkan pada kondisi default dan klik **Next**.
9. Untuk theme, pilih **Theme 2** dan klik **Next**.  
Theme adalah koleksi berbagai macam template yang bisa di gunakan untuk mendesain layout dan style seluruh aplikasi
10. Setelah yakin klik **Create** atau pilih **Previous** unuk melakukan perubahan.  
Setelah mengklik **Create**, akan ada konfirmasi seperti berikut jika berhasil :  
Application created successfully.

## II.I Menjalankan Aplikasi baru Anda.

Untuk menjalankan aplikasi :

1. klik **Run Application** icon.



2. Pada halaman login, Masukan **hr** untuk **User Name** and **Password**.  
Aplikasi akan di tampilkan, yang menunjukan table EMPLOYEES .
3. Explore aplikasi Anda.  
Kamu bisa melakukan query table EMPLOYEES . Untuk mengatur aplikasi , gunakan Developer Toolbar pada di bawah .

Edit Application	Edit Page 1	Create	Session	Debug	Show Edit Links
------------------	-------------	--------	---------	-------	-----------------

Developer toolbar menawarkan cara cepat untuk mengedit halaman yang sedang aktif, mebuat halaman baru, control , atau komponen, debugging dan lain -lain

4. Untuk keluar dari aplikasi dan kembali ke Application Builder , klik **Edit Page1** pada Developer toolbar.
5. Untuk kembali ke Database Home Page, pilih **Home** pada halaman yang paling atas.

Home > Application Builder > Application 108 > Page Definition

### III. Membuat Table

#### A. Membuat Table baru dengan Object Browser

1. Pada Database Home Page, klik **Object Browser** icon. Menu Object browser tampil
2. Klik **Create**.
3. Dari type list object , pilih **Table**.
4. Input Nama table. Nama table harus sesuai dengan penamaan system oracle. Jangan memulai dengan spasi, angka atau underscore.
5. Untuk memiliki nama tabel terakhir sesuai dengan kasus yang dimasukkan dalam bidang Nama Tabel, klik **Preserve Case**.
6. Masukan detail untuk setiap kolom. Untuk setiap kolom :
  - a. Masukan nama kolom
  - b. Pilih tipe kolom .Select the column type. Tipe data yang tersedia :NUMBER, VARCHAR2, DATE, TIMESTAMP, CHAR, CLOB, BLOB, NVARCHAR2, BINARY\_FLOAT, dan BINARY\_DOUBLE
  - c. Masukan informasi berikut yang sesuai :
    1. Untuk kolom yang tidak boleh berisi NULL, checklist checkbox **NOT NULL**.
    2. Untuk mengubah urutan kolom klik **UP** tau **Down** . Untuk menambahkan kolom baru klik **Add Colom**.

The screenshot shows the 'Create Table' dialog box. On the left, there is a sidebar with buttons: 'Columns', 'Primary Key', 'Foreign Key', 'Constraints', and 'Confirm'. The main area is titled 'Create Table' and contains a 'Table Name' field, a 'Preserve Case' checkbox, and a table with columns: 'Column Name', 'Type', 'Precision', 'Scale', 'Not Null', and 'Move'. The 'Type' column has a dropdown menu open, showing options like 'NUMBER', 'VARCHAR2', 'DATE', 'TIMESTAMP', 'CHAR', 'CLOB', 'BLOB', 'NVARCHAR2', 'BINARY\_FLOAT', and 'BINARY\_DOUBLE'. The 'Not Null' column has checkboxes. The 'Move' column has up and down arrows. At the bottom right, there is an 'Add Column' button. At the top right, there are 'Cancel' and 'Next >' buttons.

7. Click **Next**.

Selanjutnya , adalah mendefinisikan Primary key untuk table ini (bersifat Optional)  
Pilih dari salah satu berikut :

  - **No Primary Key -**
  - **Populated from a new sequence –**
  - **Populated from an existing sequence –**
  - **Not populated –**

8. Klik Next,

Selanjutnya adalah membuat foreign keys (bersifat optional).

To add a foreign key:

- a) Masukkan nama untuk Constrain Foreign
- b) Pilih **Select Key Column(s)** – pilih kolom yang akan di jadikan Foreign key Select the columns that are part of the foreign key. Once selected, click the **Add** icon to move them to Key Column(s).
- c) References Table – Pilih table yang di refrence oleh foreign key.
- d) Kemudian pilh kolom yang di refrence pada **Select Reference Column(s)** kemudian klik **Add** icon

Pilih salah satu Integritas refrential berikut:

- **Disallow Delete** -.
- **Cascade Delete** -.
- **Set to Null on Delete** - .

Click **Add**.

Click **Next**.

9. Selanjutnya adalah halaman constrain, berfungsi untuk mebuat constrain (bersifat optional) . Anda bisa membuat multiple constrain, tapi haru membuatnya secara terpisah.

10. Klik Finish

## B. Membuat Table dengan SQL Comand

### I. To create a new table Using SQL Comand :

1. Pada Database Home Page, klik **SQL** icon, kemudian pilih **SQL COMMANDS**.
2. Tulis SQL syntax.

Contoh syntax untuk membuat table mhs :

```
CREATE TABLE mhs(  
  nim varchar(12) NOT NULL,  
  nama varchar(50) NOT NULL,  
  sex CHAR NOT NULL  
      CHECK (sex IN ('M', 'F')),  
  alamat varchar(50) DEFAULT NULL,  
  jurusan NOT NULL,  
  PRIMARY KEY(nim),  
);
```

3. Kemudian klik **Run**
4. Dan table pun di buat.

### II. Show description Table :

```
DESC table_name;
```

### III. Delete table Using SQL Comand :

```
DROP TABLE table_name;
```

### IV. Change table structure Using SQL Comand :

```
ALTER TABLE table_name alter_option;
```

*Alter\_option* adalah perintah spesifik untuk mengubah struktur table :

- **ADD** new\_field;
- **ADD INDEX** index\_name;
- **ADD PRIMARY KEY**(Key\_Field)
- **CHANGE** Field\_target New\_definition\_Field
- **MODIFY** Field\_definition
- **DROP** Field\_name\_to\_delete;
- **RENAME TO** New\_Table\_Name;

Example :

```
ALTER TABLE mhs ADD religi VARCHAR(15) NOT NULL;  
ALTER TABLE mhs ADD PRIMARY KEY(nim);  
ALTER TABLE mhs CHANGE religi religi VARCHAR(2) NOT NULL;  
ALTER TABLE mhs MODIFY religi CHAR(2) NOT NULL;  
ALTER TABLE mhs DROP religi;  
ALTER TABLE mhs RENAME TO mahasiswa;
```

### V. Untuk mengubah struktur table dengan object browser akan di bahas pada pertemuan berikut.

1. Apa yang kalian ketahui tentang Oracle XE ?
2. Bagaimanakah cara membackup Database di Oracle XE ?
3. Apa perbedaan Oracle XE dengan Oracle SE atau Oracle EE ?
4. Apa perbedaan / kelebihan dan kekurangan pembuatan table metode SQL dengan metode Object Browser ?
5. Analisa Skema Database **HR** , deskripsikanlah entitas dari setiap table yang ada!



# Sesion II (Operasi Terhadap Data)

## I. Insert, Update, dan Delete Record record dengan Object Browser

- Klik **Object Browser** → **Browse** → **Table**
- Pilih nama table yang akan di ubah, funtuk contoh table *region* pada database **HR** .  
Dan menu ini akan tampil

REGIONS

Create ▼

Table

Data

Indexes

Model

Constraints

Grants

Statistics

UI Defaults

Triggers

Dependencies

SQL

Add Column

Modify Column

Rename Column

Drop Column

Rename

Copy





Drop

Truncate

Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
REGION_ID	NUMBER	No	-	1
REGION_NAME	VARCHAR2(25)	Yes	-	-
1 - 2				

1. Pada Tab **Table** terdapat beberapa operasi seperti menambahkan kolom baru pada table, modifikasi kolom,menghapus, mengganti nama table dan lain – lain
2. Pada Tab **Data** kita bisa melakukan Query pada record, jumlah baris, memasukan data dan mengedit hasil record table

Table	Data	Indexes	Model	Constraints
Query	Count Rows	Insert Row		
EDIT	REGION_ID	REGION_NAME		
	1	Europe		
	2	Americas		
	3	Asia		
	4	Middle East and Africa		
row(s) 1 - 4 of 4				

Download

[Download](#)

## II. Insert, Update, and Delete Record dengan SQL Command

### A. Insert Record

Untuk memasukan nilai atau insert record ke table teradapat tiga cara :

```
INSERT INTO table_name VALUES ('value 1', 'value 2', '.....');
```

```
INSERT INTO table_name (field 1, field 2, field 3, ....) VALUES ('value 1', 'value 2',
```

```
INSERT INTO table_name SET (field 1='value 1', field 2= 'value 2', '.....');
```

Example :

```
INSERT INTO mhs VALUES('41507010080','JOKO','M', 'Jakarta',");
```

### B. Update Record

Untuk mengupdate nilai record dapat di lakukan dengan perintah berikut :

```
UPDATE table_name SET Field 1 = 'New_Value 1', Field 2 = 'New_Value 2',  
Field .... = 'New_Value ....' [WHERE CONDITION_FIELD];
```

Example

```
UPDATE mhs SET alamat='Bogor', sex='F' WHERE nim='41507010069';
```

### C. Delete Record

Untuk menghapus nilai dari suatu record dapat di lakukan dengan perintah berikut :

```
DELETE FROM table_name [WHERE CONDITION_FIELD];
```

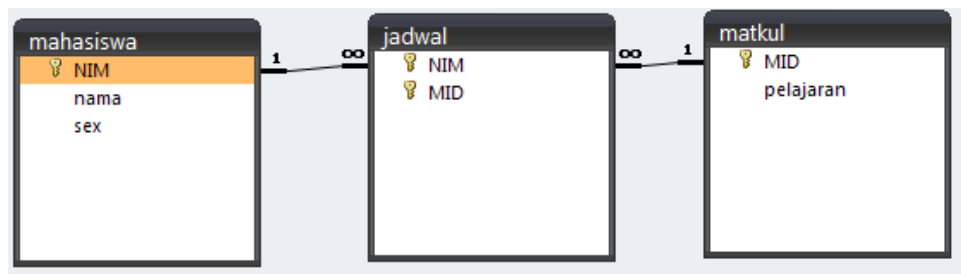
Example

```
DELETE FROM mhs WHERE nim='41507010069';
```

### III. Membuat Table dengan Primer dan Komposit Key

- **Primary Key**  
Primary key dari suatu tabel harus berisi nilai yang unik, dan non-null untuk setiap barisnya
- **Foreign Key**  
Foreign Key adalah kolom atau himpunan kolom yang menghubungkan setiap baris dalam child table yang berisi Foreign Key dengan baris dari parent table yang berisi Primary Key yang sesuai/match.
- **Candidate Key**
  - Superkey (K) dalam relasi
  - Untuk setiap relasi R, nilai K akan mengidentifikasi secara unik tuplenya.Jika Candidate key terdiri dari beberapa atribut, disebut **composite key**.

Sebagai contoh perhatikanlah Skema **Kampus** berikut ini :



Skema tersebut menggambarkan setiap mahasiswa dapat mengambil banyak mata kuliah, dan setiap mata kuliah dapat di ambil oleh banyak mahasiswa.

Table Mahasiswa

```
CREATE TABLE "MAHASISWA" (
  "NIM" VARCHAR2(12) NOT NULL ENABLE,
  "NAMA" VARCHAR2(50) NOT NULL ENABLE,
  "SEX" CHAR(1) NOT NULL ENABLE,
  CHECK (sex IN ('M', 'F')) ENABLE,
  PRIMARY KEY ("NIM") ENABLE );
```

Table Matkul

```
CREATE TABLE "MATKUL"
(
  "MID" VARCHAR2(12) NOT NULL ENABLE,
  "PELAJARAN" VARCHAR2(50),
  PRIMARY KEY ("MID") ENABLE );
```

Untuk membuat foreign key serta cara membuat composite key pada table **jadwal** dapat menggunakan perintah SQL berikut :

```
CREATE TABLE jadwal(
  nim varchar(12) NOT NULL,
  mid varchar(12) NOT NULL,
  FOREIGN KEY (mid) REFERENCES matkul , FOREIGN KEY (nim) REFERENCES mahasiswa
  PRIMARY KEY("NIM","MID") );
```

## Latihan Sesion 2

---

1. Apa kegunaan primary key dan foreign key ?
2. Buatalah Skema **Dreamhome Database** ( lihat table di halaman paling terkahir )!
3. Masukkan data untuk setiap tabel yang di buat !
4. Analisa Entitas dari setiap table **Dreamhome Database** ?

# Sesion III (Kekangan pada Data)

## A. Contoh tipe data:

Tipe Data	Penjelasan
char(n)	String sepanjang n karakter. Bila n tidak disertakan, maka panjang karakter adalah 1. Contoh : char = 1, char(3) = 3
varchar(n)	String yang panjangnya bisa berubah-ubah sesuai kebutuhan, namun string tersebut dibatasi sebanyak n karakter. ( <i>Oracle merekomendasikan varchar2. Bila diketikan varchar pada oracle maka otomatis berubah menjadi varchar2</i> ). Contoh : varchar(5) = batas varchar adalah 5
varchar2(n)	String yang panjangnya bisa berubah –ubah sesuai dengan kebutuhan, namun string tersebut dibatasi sebanyak n karakter. Maksimum karakter pada varchar2 adalah 2000 karakter. Contoh : varchar2(10) = batas varchar2 adalah 10
long	Tipe data binary, maksimum 2 Giga Byte, disimpan dalam format internal Oracle.
long raw	Sama dengan long yaitu tipe data binary, maksimum 2 Giga Byte, tidak dikonversi oleh Oracle (data mentah apa adanya).
date	Tanggal, menyimpan tahun, bulan, hari, jam, menit, dan detik.
number(n,p)	Angka pecahan, baik fixed decimal ataupun floating point. Nilai n adalah jumlah bytes total dan p adalah presisi angka di belakang koma. Contoh : number(4,2) = 4 (jumlah batas), 2 (presisi angka di belakang koma)

Contoh :

```
CREATE TABLE "MAHASISWA" (  
  "NIM" VARCHAR2(12) NOT NULL ENABLE,  
  "NAMA" VARCHAR2(50) NOT NULL ENABLE,  
  "SEX" CHAR(1) NOT NULL ENABLE,  
  CHECK (sex IN ('M', 'F')) ENABLE,  
  PRIMARY KEY ("NIM") ENABLE  
);
```

Dilihat dari kolom NIM dan NAMA memakai varchar2 karena bila terjadi perubahan pembatasan pada kolom NIM maka tidak perlu di update, dan bila kolom NAMA tidak sampai batasnya maka tidak memakan memory yang ada.

Dilihat dari kolom SEX maka hanya memakai char yang batasnya 1, karena hanya memakai 1 karakter saja bisa 'M' atau 'F'.

Contoh di login HR.

Tabel countries

Column Name	Data Type	Nullable	Default	Primary Key
COUNTRY_ID	CHAR(2)	No	-	1
COUNTRY_NAME	VARCHAR2(40)	Yes	-	-
REGION_ID	NUMBER	Yes	-	-

Jika insert untuk country\_id dengan nilai char lebih dari 2.

insert into countries (country\_id) values ('abcd');

```
ORA-12899: value too large for column "HR"."COUNTRIES"."COUNTRY_ID" (actual: 4, maximum: 2)
```

Jika insert untuk region\_id dengan nilai char.

insert into countries (region\_id) values ('abcd');

```
ORA-01722: invalid number
```

## B. KEKANGAN DALAM BASIS DATA

Dalam perancangan dan penyusunan basis data dikenal adanya beberapa kekangan atau aturan yang harus ditaati/dipatuhi dalam file-file basis data.

Kekangan tersebut berhubungan dengan aspek-aspek penting dalam basis data, yaitu:

1. Kerangkapan data
2. Inkonsistensi data
3. Data terisolasi
4. Keamanan data
5. Integritas data

### 1. Kerangkapan Data (Data Redundancy)

Kerangkapan data adalah munculnya data-data yang secara berlimpah/berulang kali pada file basis data yang semestinya tidak diperlukan.

#### a. Alasan menghindari kerangkapan data

1. Pemborosan media penyimpanan data
2. Biaya penyimpanan data yang semakin besar
3. Kesulitan/inefisiensi dalam pengolahan data
4. Pemborosan waktu dalam pengolahan data
5. Semakin besar kemungkinan muncul data tidak konsisten

#### b. Kejadian kerangkapan data dapat terjadi pada dua kemungkinan

1. Kerangkapan data dalam satu file
2. Kerangkapan data dalam beberapa file

#### c. Kerangkapan data dalam 1 file

Kerangkapan data dalam 1 file terjadi jika muncul kerangkapan nilai-nilai rinci data dalam 1 file tersebut.

Contoh 1  
File Karyawan

NIK	Nama_Karyawan	Alamat	Gol_Gaji	Gaji_Pokok
K001	Rita	Yogyakarta	IIIA	500.000
K002	Rina	Semarang	IVA	750.000
K003	Rini	Jakarta	IIIA	500.000
K004	Rani	Yogyakarta	IIIB	550.000
K005	Rika	Surabaya	IVA	750.000

d. Menghindari kerangkapan data

Untuk menghindari kerangkapan data di dalam file, dapat dilakukan dengan cara mengubah struktur file, yaitu dengan cara memecah file tersebut menjadi dua buah file baru.

Contoh pada file karyawan sebelumnya dapat dipecah menjadi file Karyawan\_1 dan file Golongan.

Contoh  
File Karyawan\_1

NIK	Nama_Karyawan	Alamat	Gol_Gaji
K001	Rita	Yogyakarta	IIIA
K002	Rina	Semarang	IVA
K003	Rini	Jakarta	IIIA
K004	Rani	Yogyakarta	IIIB
K005	Rika	Surabaya	IVA

Contoh  
File Golongan

Gol_Gaji	Gaji_Pokok
IA	100.000
IIA	300.000
IIIA	500.000
IIIB	550.000
IVA	750.000

e. Kesimpulan

Kerangkapan data dalam 1 file dapat diatasi dengan cara memecah file tersebut, menjadi file-file baru yang mempunyai struktur lebih sederhana.

Banyaknya file baru yang terbentuk adalah bergantung pada banyaknya kerangkapan data yang terjadi.

2. Data Tidak Konsisten (Data Inconsistency)

Data tidak konsisten adalah munculnya data yang tidak konsisten pada medan/kolom yang sama dalam satu atau beberapa file data yang dihubungkan/direlasikan.

Data tidak konsisten dapat terjadi akibat:

1. Proses pemasukan data (data entry) yang tidak benar
2. Proses pembaharuan data (update) yang tidak benar
3. Pengendalian sistem yang tidak baik/terkontrol

Kejadian data tidak konsisten juga dapat terjadi pada:

1. Data tidak konsisten dalam 1 file
2. Data tidak konsisten dalam beberapa file

Data tidak konsisten dalam 1 file

- Data tidak konsisten dalam 1 file terjadi jika kemunculan data yang tidak konsisten terjadi pada 1 file (yang mengalami kerangkapan data)

Contoh 1:

File Karyawan

NIK	Nama_Karyawan	Alamat	Gol_Gaji	Gaji_Pokok
K001	Rita	Yogyakarta	IIIA	500.000
K002	Rina	Semarang	IVA	750.000
K003	Rini	Jakarta	IIIA	500.000
K004	Rani	Yogyakarta	IIIB	550.000
K005	Rika	Surabaya	IVA	750.000

Akibat

- Inkonsistensi data tersebut akan mengakibatkan kesalahan informasi pada hasil pengolahan data, misal:
  1. Kesalahan pada saat mencetak struk daftar perolehan gaji karyawan
  2. Kesalahan jumlah total pengeluaran uang yang dikeluarkan untuk gaji karyawan

Pemecahan

- Inkonsistensi data dalam 1 file dapat dihindari dengan cara yang sama sebagaimana permasalahan kerangkapan data dalam 1 file, yaitu dengan memecah file menjadi file-file baru yang lebih sederhana dan tetap saling berhubungan.

### 3. Data Terisolasi

Data terisolasi disebabkan oleh pemakaian beberapa file basis data dimana program aplikasi tidak dapat mengakses data-data dari file tertentu, kecuali jika program aplikasi file tertentu, kecuali jika program aplikasi diubah/ditambah, sehingga seolah-olah ada file yang terpisah/terisolasi terhadap file yang lain dalam basis data.

Penyebab data terisolasi

1. Tidak adanya kemungkinan untuk menghubungkan antar data dalam file
2. Tidak adanya standarisasi data (berkaitan dengan domain/format data, meliputi tipe dengan domain/format data, meliputi tipe dan ukuran)

Contoh

File Mahasiswa

NIM	Nama_Mahasiswa
02050001	Rita
02050002	Rina
02050003	Rini
02050004	Rani
02050005	Rika



#### File Minat\_Mahasiswa\_1

NIM	Minat
02050001	Pemrograman
02050002	Jaringan
02050003	Web
02050004	Basis Data
02050005	Multimedia

#### File Pembimbing\_Minat

Kode_Pembimbing	Nama_Pembimbing
P001	Dani
P002	Dina
P003	Dino
P004	Dion
P005	Doni

#### Permasalahan

• Seandainya diperlukan informasi mengenai siapa pembimbing minat mahasiswa bernama Rita dengan NIM 02050001 yang mempunyai minat Pemrograman, maka file-file tersebut minat Pemrograman, maka file-file tersebut tidak memenuhi kebutuhan informasi tersebut. Hal ini karena tidak ada hubungan antara file Pembimbing\_Minat dengan file Mahasiswa dan file Minat\_Mahasiswa\_1

#### Cara mengatasi?

• Untuk mengatasinya maka perlu dirancang sebuah file baru yang berfungsi untuk menghubungkan antara dua minat dengan data pembimbing

#### File Membimbing

Kode_Pembimbing	Minat
P001	Pemrograman
P002	Jaringan
P003	Web
P004	Basis Data
P005	Multimedia

Cara pertama seperti ini baik dilakukan apabila ada kemungkinan bahwa seorang pembimbing dapat membimbing lebih dari 1 minat.

#### Jawaban :

#### File Minat\_Mahasiswa

NIM	Kode_Minat
02050001	M001
02050002	M002
02050003	M003
02050004	M004
02050005	M005

#### File Pembimbing\_Minat

Kode_Pembimbing	Nama_Pembimbing
P001	Dani
P002	Dina
P003	Dino
P004	Dion
P005	Doni

#### File Membimbing

Kode_Pembimbing	Kode_Minat
P001	M001
P002	M002
P003	M003
P004	M004
P005	M005

#### Penyebab Lain

- Dalam hal ini data terisolasi muncul akibat domain/format data yang tidak standar, maka permasalahan ini hanya dapat diselesaikan dengan cara merubah/menyesuaikan format dengan cara merubah/menyesuaikan format data dalam file basis data sehingga data-data di dalamnya dapat saling dihubungkan.

#### Cara mengatasi Data Terisolasi

1. Menambah file baru bertipe transaksi yang berfungsi sebagai penghubung antar data dalam file-file lain yang telah ada.
2. Menambah kolom yang berfungsi sebagai penghubung dengan file-file lain yang telah ada
3. Menyesuaikan domain kolom yang berfungsi untuk menghubungkan antar file

#### 4. Keamanan Data (Data Security)

Prinsip dasar dari keamanan data dalam basis data adalah bahwa data-data dalam basis data merupakan sumber informasi yang bersifat sangat penting dan rahasia. sangat penting dan rahasia.

#### Aspek Keamanan Data

1. Recovery
2. Integrity
3. Concurrency
4. Privacy 4. Privacy
5. Security

#### 5. Integritas Data (Data Integrity)

Integritas data berhubungan dengan kinerja sistem agar dapat melakukan kendali/control pada semua bagian system. Integritas dimaksudkan sebagai suatu sarana. Integritas dimaksudkan sebagai suatu sarana untuk meyakinkan bahwa data-data yang tersimpan dalam basis data selalu berada dalam kondisi yang benar (tipe dan ukuran datanya), up to date (sesuai dengan kondisi aktual), konsisten, dan selalu tersedia (current).

#### Menjaga Integritas Data

- Salah satu cara terbaik menjaga integritas data adalah meyakinkan bahwa nilai-nilai data adalah benar sejak masuk pertama kali.

- Hal ini ditempuh dengan beberapa metode, misalnya mengeset secara seksama prosedur penangkapan data yang dilakukan secara manual, atau dengan membuat modul dalam program aplikasi untuk mengecek validitas nilai data pada saat diinput.

Aspek hubungan Integritas data

- Integritas domain
- Key constraints, berkaitan dengan dua hal, yaitu integritas entitas pada kunci primer relasi dan integritas referensial pada kunci relasi dan integritas referensial pada kunci penghubung relasi.

### Soal bab 3:

Terdapat ilustrasi sebagai berikut :

Direktur menjelaskan deskripsi bagaimana perusahaannya berjalan :

1. Setiap barang mempunyai supplier yang berbeda-beda dalam menyuplai barang ke gudang perusahaan. Kemudian dicatat setiap tanggalnya dan stok pada saat barang disuplai ke gudang. Lalu setiap terjadi penyuplaian barang, maka data supplier dicatat.
2. Setiap barang mempunyai satuan untuk membedakan ukuran barang dan jumlah total stok yang dihitung didalam gudang dari supplier yang berbeda-beda tiap barangnya.
3. Setiap terjadi transaksi dengan pembeli atau customer akan dicatat tiap tanggal pembelian dan jumlah total pembelian tiap barang yang dibeli. Kemudian setiap terjadi pembelian, data customer dicatat.

Direktur memberikan beberapa contoh tipe query basis data untuk membantu perusahaannya :

1. Nama dan nomor telp supplier yang terdapat di Bandung.
2. Nama dan nomor telp customer yang terdapat di Solo.
3. Total jumlah pasok berdasarkan barang dan supplier pada bulan Januari 2010.
4. Total jumlah pembelian berdasarkan barang dan customer pada bulan Januari 2010.
5. Tanggal pembelian yang jumlahnya diatas rata-rata

Soalnya :

- a. Identifikasi entitas yang ada
- b. Identifikasi atribut dari setiap entitas
- c. Tentukan primary key dari setiap entitas
- d. Tentukan tipe data setiap atributnya

## Sesion IV (Integritas Referensial)

- Foreign Key adalah kolom atau himpunan kolom yang menghubungkan setiap baris dalam *child table* yang berisi Foreign Key dengan baris dari *parent table* yang berisi Primary Key yang sesuai/match.
- Integritas referential berarti, jika FK berisi suatu nilai, maka nilai tersebut harus mengacu kesuatu baris dalam *parent table*.
- Standard ISO menyediakan pendefinisian untuk FK dengan clause FOREIGN KEY dalam CREATE dan ALTER TABLE:

Contoh :

```
CREATE TABLE jadwal(  
  nim varchar(12) NOT NULL,  
  mid varchar(12) NOT NULL,  
  FOREIGN KEY (mid) REFERENCES matkul  
  FOREIGN KEY (nim) REFERENCES mahasiswa  
);
```

- Operasi INSERT/UPDATE yang berusaha untuk membuat nilai FK dalam *child table* tanpa nilai *candidate key* yang sesuai dalam *parent table*.
- Aksi yang dilakukan yang berusaha untuk merubah / menghapus (update/delete) nilai *candidate key* dalam *parent table* yang memiliki baris yang sesuai dalam *child table* tergantung pada **referential action** yang ditetapkan dengan subclause ON UPDATE dan ON DELETE. Terdapat 4 pilihan aksi, yaitu :
  - **CASCADE**, menghapus baris dari *parent table* dan secara otomatis menghapus baris yang sesuai dalam *child table*, jika baris yang dihapus tadi merupakan *candidate key* yang digunakan sebagai *foreign key* pada tabel lainnya, maka aturan *foreign key* untuk tabel ini dihilangkan.
  - **SET NULL**, menghapus baris pada *parent table* dan menetapkan nilai *foreign key* dalam *child table* menjadi NULL. Berlaku jika kolom *foreign key* mempunyai qualifier NOT NULL.
  - **SET DEFAULT**, menghapus baris dari *parent table* dan menetapkan setiap komponen foreign key dari *child table* menjadi default yang telah ditetapkan. Berlaku jika kolom foreign key memiliki nilai DEFAULT.
  - **NO ACTION**, menolak operasi penghapusan dari *parent table*. Merupakan default jika aturan ON DELETE dihilangkan

Perhatikan Skema Database **HR**

**Contoh 1 :**

Pada tabel **COUNTRIES**, filed **REGION\_ID** merupakan foreign key yang mengacu ke tabel **REGIONS**. Untuk menetapkan aturan penghapusan, jika salah satu record **REGION\_ID** dihapus dari tabel **REGIONS**, maka nilai **REGION\_ID** yang ada pada **COUNTRIES** akan diganti menjadi NULL. Sehingga dapat dituliskan :

```
FOREIGN KEY (REGION_ID) REFERENCES COUNTRIES ON DELETE SET NULL
```

## Contoh 2 :

Pada tabel **COUNTRIES**, filed **REGION\_ID** merupakan foreign key yang mengacu ke tabel **REGIONS**. Untuk mendefinisikan aturan peng-update-an, jika **REGION\_ID** dalam **REGIONS** di-update, maka kolom yang terkait dalam tabel **COUNTRIES** akan diganti dengan nilai baru, sehingga dapat dituliskan :

**FOREIGN KEY (REGION\_ID) REFERENCES COUNTRIES ON UPDATE CASCADE**

Perhatikan Skema Database **Kampus** Pada Bab 2

Untuk membuat Table Jadwal dengan object Browser, ikuti langkah berikut :

- Buat 2 kolom MID dan NIM yang scalenya sama dengan table induk

Table

Create Table

\* Table Name:  ☐ Preserve Case

Column Name	Type	Precision	Scale	Not Null	Move
<input type="text" value="MID"/>	<input type="text" value="VARCHAR2"/>		<input type="text" value="12"/>	<input type="checkbox"/>	▼ ▲
<input type="text" value="NIM"/>	<input type="text" value="VARCHAR2"/>		<input type="text" value="12"/>	<input type="checkbox"/>	▼ ▲

- Pilih **Not populated** Klik Next

•

Table

Primary Key

Table name: **JADWAL**

Primary Key: ☐ No Primary Key  
☐ Populated from a new sequence  
☐ Populated from an existing sequence  
☒ Not populated

\* Primary Key Constraint Name:

\* Primary Key:

Composite Primary Key:

- Pada Foreign Key pertama pilih MID dan pilih refrence tabelnya , lalu klik Add

Foreign Key  
 Constraints  
 Confirm

Add Foreign Key Add

\* Name

☐ Disallow Delete  
☒ Cascade Delete  
☐ Set Null on Delete

Select Key Column(s)  

NIM

MID

\* Key Column(s)  

MID

\* References Table

Select Reference Column(s)  

PELAJARAN

MID

\* Referenced Column(s)  

MID

- Pada Foreign key Ke dua Pilih NIM dan pilih refrence tabelnya, lalu klik Add

•

Foreign Key  
 Constraints  
 Confirm

JADWAL\_FK2   MID   MATKUL   MID   cascade   ✗

Add Foreign Key Add

\* Name

☐ Disallow Delete  
☒ Cascade Delete  
☐ Set Null on Delete

Select Key Column(s)  

MID

NIM

\* Key Column(s)  

NIM

\* References Table

Select Reference Column(s)  

NAMA  
SEX  
ALAMAT  
BIRTH

NIM

\* Referenced Column(s)  

NIM

- Selanjutnya klik Next

Columns  
 Primary Key  
 Foreign Key  
 Constraints

Foreign Keys

Foreign Key	Columns	Referenced Table	Referenced Columns	Action
JADWAL_FK2	MID	MATKUL	MID	cascade <span style="color: red;">✗</span>
JADWAL_FK3	NIM	MAHASISWA	NIM	cascade <span style="color: red;">✗</span>

ORACLE®

Page 21

- Dan yang terakhir klik Finish.
- Atau Untuk SQL Comandnya dapat menggunakan perintah di bawah ini :

```
CREATE TABLE jadwal(
  nim varchar(12) NOT NULL,
  mid varchar(12) NOT NULL,
  FOREIGN KEY (mid) REFERENCES matkul ON DELETE SET NULL,
  FOREIGN KEY (nim) REFERENCES mahasiswa ON DELETE SET NULL
);
```

- Kita juga dapat mengubah Integritas referential yang sudah ada dengan menggunakan Menu **Constraint**, yang terdapat pada Menu Object Browser → Table  
Example :

JOBS

Create

Table

Data

Indexes

Model

Constraints

Grants

Statistics

UI Defaults

Triggers

Dependencies

SQL

Create

Drop

Enable

Disable

Constraint	Type	Table	Search Condition	Delete Rule	Status	Last Change	Index	Invalid
JOB_TITLE_NN	C	JOBS	"JOB_TITLE" IS NOT NULL	-	ENABLED	07-02-2006	-	-
JOB_ID_PK	P	JOBS	-	-	ENABLED	07-02-2006	JOB_ID_PK	-
								1 - 2

- Pada Menu **Create** kita dapat membuat Constraint dengan Type :
  - Check : Melakukan cheking, misalkan menetapkan peraturan Field *SEX* hanya dapat berisi dengan nilai M/F, atau tidak boleh bernilai NULL
  - Primary Key : Kita dapat membuat Komposite Key dan hal lainnya
  - Foreign Key : Pada menu ini kita dapat membuat dan mengatur Refrensial table dari setiap foreign key
  - Unique
- Pada Menu **Drop** kita dapat menghapus Constraint yang Ada
- Menu **Enable** dan **Disble** untuk mengaktifkan dan menonaktifkan Constraint yang telah di buat

## Latihan Bab 4

1. Modifikasilah skema **Dreamhome Database** sehingga skema tersebut memiliki Integritas Referensial !
2. Analisa Skema Database **HR** , Integritas apa saja yang terdapat dalam skema tersebut ?

# Sesion V (Mengenal Dasar Query)

---

## Tujuan dari SQL

- Idealnya, *database language* dapat memungkinkan user untuk:
  - Membuat struktur relasi dan database;
  - Melakukan operasi penyisipan (insertion), perubahan (modification) dan penghapusan (deletion) data dari relasi;
  - Melakukan query sederhana dan rumit.Database language harus melaksanakan operasi-operasi tersebut dengan usaha minimal yang dilakukan user dan sintaks/struktur instruksi harus mudah dipahami/dipelajari.
- Harus portable sehingga memungkinkan untuk pindah dari satu DBMS ke DBMS lainnya.
- SQL merupakan *transform-oriented language* dengan 2 komponen utama:
  - DDL untuk definisi struktur database.
  - DML untuk pengambilan (retrieving) dan perubahan (updating) data.
- SQL cukup mudah dipelajari, karena:
  - Merupakan bahasa non-procedural – cukup menspesifikasikan informasi apa yang dibutuhkan daripada bagaimana mendapatkannya.
  - Pada dasarnya mempunyai format yang bebas.
- Terdiri dari bahasa Inggris standard, seperti :
  - CREATE TABLE Staff(staffNo VARCHAR(5), IName ARCHAR(15), salary DECIMAL(7,2));
  - INSERT INTO Staff VALUES ('SG16', 'Brown', 8300);
  - SELECT staffNo, IName, salary FROM Staff WHERE salary > 10000;
- Dapat digunakan oleh bermacam-macam user, termasuk DBA, manajemen, pembuat aplikasi dan user lainnya.

## Pentingnya SQL

- SQL telah menjadi bagian dari arsitektur aplikasi seperti arsitektur aplikasi sistem IBM.
- Merupakan pilihan yang strategis untuk organisasi besar dan berpengaruh (co : X/OPEN).
- SQL digunakan untuk standar lainnya dan mempengaruhi pembuatan standar lainnya sebagai *definitional tool*. Contoh :
  - Standar ISO Information Resource Directory System (IRDS).
  - Standar Remote Data Access (RDA).

## Penulisan perintah SQL

- Statemen SQL terdiri dari *reserved word* dan *user-defined word*.
  - Reserved word adalah bagian yang telah ditetapkan pada SQL dan penulisannya harus sesuai dan tidak bisa dipisah-pisahkan.
  - User-defined word dibuat oleh user dan merepresentasikan nama-nama berbagai objek database seperti relasi, kolom dan view.
- Kebanyakan komponen dari perintah SQL bersifat *case insensitive*, kecuali untuk data literal karakter.
- Mudah dibaca dengan pengaturan baris dan spasi :
  - Setiap clause dimulai pada baris baru.
  - Awal dari suatu clause harus berurutan dengan clause lainnya.
  - Jika clause mempunyai beberapa bagian, harus ditampilkan pada baris yang berbeda dan diberi spasi pada awal clause.
- Menggunakan bentuk notasi Backus Naur Form (BNF) :
  - Menggunakan huruf besar untuk merepresentasikan *reserved word*.
  - Menggunakan huruf kecil untuk merepresentasikan *user-defined word*.
  - Kurung kurawal mengindikasikan *required element*.
  - Kurung siku mengindikasikan *optional element*.



### **SQL Reserved Words**

<b>Begins with:</b>	<b>Reserved Words</b>
A	ACCESS, ADD*, ALL*, ALTER*, AND*, ANY*, AS*, ASC*, AUDIT
B	BETWEEN*, BY*
C	CHAR*, CHECK*, CLUSTER, COLUMN, COMMENT, COMPRESS, CONNECT*, CREATE*, CURRENT*
D	DATE*, DECIMAL*, DEFAULT*, DELETE*, DESC*, DISTINCT*, DROP*
E	ELSE*, EXCLUSIVE, EXISTS
F	FILE, FLOAT*, FOR*, FROM*
G	GRANT*, GROUP*
H	HAVING*
I	IDENTIFIED, IMMEDIATE*, IN*, INCREMENT, INDEX, INITIAL, INSERT*, INTERSECT*, INTO*, IS*
L	LEVEL*, LIKE*, LOCK, LONG
M	MAXEXTENTS, MINUS, MLSLABEL, MODE, MODIFY
N	NOAUDIT, NOCOMPRESS, NOT*, NOWAIT, NULL*, NUMBER
O	OF*, OFFLINE, ON*, ONLINE, OPTION*, OR*, ORDER*
P	PCTREE, PRIOR*, PRIVILEGES*, PUBLIC*
R	RAW, RENAME, RESOURCE, REVOKE*, ROW, ROWID, ROWNUM, ROWS*
S	SELECT*, SESSION*, SET*, SHARE, SIZE*, SMALLINT*, START, SUCCESSFUL, SYNONYM, SYSDATE
T	TABLE*, THEN*, TO*, TRIGGER
U	UID, UNION*, UNIQUE*, UPDATE*, USER*
V	VALIDATE, VALUES*, VARCHAR*, VARCHAR2, VIEW*
W	WHENEVER*, WHERE, WITH*

### **PL/SQL Reserved Words**

<b>Begins with:</b>	<b>Reserved Words</b>
A	ALL, ALTER, AND, ANY, ARRAY, ARROW, AS, ASC, AT
B	BEGIN, BETWEEN, BY
C	CASE, CHECK, CLUSTERS, CLUSTER, COLAUTH, COLUMNS, COMPRESS, CONNECT, CRASH, CREATE, CURRENT
D	DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DROP
E	ELSE, END, EXCEPTION, EXCLUSIVE, EXISTS
F	FETCH, FORM, FOR, FROM
G	GOTO, GRANT, GROUP
H	HAVING
I	IDENTIFIED, IF, IN, INDEXES, INDEX, INSERT, INTERSECT, INTO, IS
L	LIKE, LOCK
M	MINUS, MODE
N	NOCOMPRESS, NOT, NOWAIT, NULL
O	OF, ON, OPTION, OR, ORDER, OVERLAPS
P	PRIOR, PROCEDURE, PUBLIC
R	RANGE, RECORD, RESOURCE, REVOKE
S	SELECT, SHARE, SIZE, SQL, START, SUBTYPE
T	TABAUTH, TABLE, THEN, TO, TYPE
U	UNION, UNIQUE, UPDATE, USE
V	VALUES, VIEW, VIEWS
W	WHEN, WHERE, WITH

## I. Menampilkan Record

- Kita dapat Menjalankan Perintah SQL pada Menu **Home→SQL→SQL Commands**.
- Untuk menampilkan data kita dapat menggunakan perintah sebagai berikut :

```
SELECT [TableName.FieldName] FROM TableName [WHERE Condition];
```

Example :

- Menampilkan Seluruh isi table **Regions** pada Database **HR**

```
SELECT * FROM REGIONS
```

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

- Menampilkan First name, salary dan last name pada table **EMPLOYEE** pada database **HR** yang berpendapatan < 2500

```
SELECT "FIRST_NAME", "SALARY", "LAST_NAME"  
FROM "EMPLOYEES" WHERE "SALARY" <2500
```

FIRST_NAME	SALARY	LAST_NAME
James	2400	Landry
Steven	2200	Markle
TJ	2100	Olson
Ki	2400	Gee
Hazel	2200	Philtanker

Atau Anda bisa menggunakan **AS** untuk mengubah nama kolom

```
SELECT "EMPLOYEES"."FIRST_NAME" AS "Nama_Depan",  
       "EMPLOYEES"."SALARY" AS "Pendapatan",  
       "EMPLOYEES"."LAST_NAME" AS "Nama_Belakang"  
FROM "EMPLOYEES" WHERE "EMPLOYEES"."SALARY" <2500
```

Nama_Depan	Pendapatan	Nama_Belakang
James	2400	Landry
Steven	2200	Markle
TJ	2100	Olson
Ki	2400	Gee
Hazel	2200	Philtanker

- Menampilkan ALAMAT dan CITY dari table **LOCATIONS** yang berinisial IT atau JP

```
SELECT      "LOCATIONS"."STREET_ADDRESS" AS "STREET_ADDRESS",
            "LOCATIONS"."CITY" AS "CITY",
            "LOCATIONS"."COUNTRY_ID" AS "COUNTRY_ID"
FROM        "LOCATIONS" "LOCATIONS"
WHERE       "LOCATIONS"."COUNTRY_ID" ='IT' OR COUNTRY_ID='JP'
```

STREET_ADDRESS	CITY	COUNTRY_ID
1297 Via Cola di Rie	Roma	IT
93091 Calle della Testa	Venice	IT
2017 Shinjuku-ku	Tokyo	JP
9450 Kamiya-cho	Hiroshima	JP

- Menampilkan Nama pegawai dan pendapatannya yang berkisar antara 7000 hingga 7500

```
select  "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",
        "EMPLOYEES"."LAST_NAME" as "LAST_NAME",
        "EMPLOYEES"."SALARY" as "SALARY"
from    "EMPLOYEES" "EMPLOYEES"
where   "EMPLOYEES"."SALARY" >7000 AND SALARY<7500
```

FIRST_NAME	LAST_NAME	SALARY
Mattea	Marvins	7200
William	Smith	7400
Elizabeth	Bates	7300

- Menampilkan Nama Kota yang berawalan T dari table **Location**

```
select  "LOCATIONS"."CITY" as "CITY"
from    "LOCATIONS" "LOCATIONS"
where   "LOCATIONS"."CITY" like 'T%'
```

CITY
Tokyo
Toronto

## Latihan Bab 5

---

1. Apa Fungsi dari perintah SQL di bawah ini ?
  - LIKE
  - AND
  - OR
  - NOT
2. Buatlah contoh penggunaan dengan perintah SQL di atas dengan menggunakan skema Database **HR** (Soal No. 3) !
3. Dari tabel employees, siapakah employee yang salary-nya = 13000 ?
4. Apakah Jonathan memiliki employee\_id ?
5. Siapakah employee yang employee\_id = 200 dan department\_id = 10?
6. Berapakah department\_id dari Executive?

# Sesion VI (Mengenal Dasar Query)

---

Beberapa Operator Pembandingan yang bisa dipakai :

- = (sama dengan)
- < (kurang dari)
- <= (kurang dari sama dengan)
- > (lebih dari)
- >= (lebih dari sama dengan)
- != (tidak sama dengan)

## Operator Boolean :

AND, OR, NOT

Contoh :

Tampilkan nama pegawai yang gajinya lebih dari 15000

```
select first_name, last_name, salary
from employees
where salary > 15000;
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Neena	Kochhar	17000
Lex	De Haan	17000

Tampilkan nama pegawai yang gajinya 4800 atau 17000

```
select first_name, last_name, salary
from employees
where salary=4800 or salary=17000;
```

FIRST_NAME	LAST_NAME	SALARY
Neena	Kochhar	17000
Lex	De Haan	17000
David	Austin	4800
Valli	Pataballa	4800

## Operator Presedence

Perkalian dan pembagian memiliki prioritas (precedence) lebih tinggi daripada penambahan dan pengurangan.

```
select last_name, salary, 12*salary+500
```

```
from employees;
```

LAST_NAME	SALARY	12*SALARY+500
King	24000	288500
Kochhar	17000	204500
De Haan	17000	204500
Hunold	9000	108500
Ernst	6000	72500
Austin	4800	58100
Pataballa	4800	58100
Lorentz	4200	50900
Greenberg	12000	144500
Faviet	9000	108500

## Penggunaan Tanda Kurung

Penggunaan tanda kurung memiliki prioritas paling tinggi disbanding presedensi operator yang lain.

```
select last_name, salary, 12*(salary+500)
```

```
from employees;
```

LAST_NAME	SALARY	12*(SALARY+500)
King	24000	294000
Kochhar	17000	210000
De Haan	17000	210000
Hunold	9000	114000
Ernst	6000	78000
Austin	4800	63600
Pataballa	4800	63600
Lorentz	4200	56400
Greenberg	12000	150000
Faviet	9000	114000

## Operator Penyambungan

Nilai dari dua kolom atau lebih dapat digabungkan dengan menggunakan operator penyambungan ( || ).

```
select last_name || job_id
```

```
from employees;
```

LAST_NAME  JOB_ID
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
AustinIT_PROG
PataballaIT_PROG
LorentzIT_PROG
GreenbergFI_MGR
FavietFI_ACCOUNT
More than 10 rows available. Increase rows selector to view more rows.

## Soal Latihan Bab 6 :

1. Tampilkan nama pegawai yang gajinya lebih dari sama dengan 7500 dan kurang dari sama dengan 20000
2. Tampilkan nama pegawai yang gajinya tidak sama dengan 12000 dan bukan kurang dari sama dengan 10000
3. Tampilkan nomor location dan alamat yang berada di negara Jepang dan Amerika
4. Tampilkan nama pegawai digabung dengan pekerjaan dengan dipisah tanda koma, kemudian beri judul “Pegawai dan Pekerjaan”

Pegawai dan Pekerjaan

```
-----
SMITH, CLERK
ALLEN, SALESMAN
WARD, SALESMAN
JONES, MANAGER
MARTIN, SALESMAN
BLAKE, MANAGER
CLARK, MANAGER
SCOTT, ANALYST
.....
```

5. Buat query untuk menampilkan semua kolom dari table EMPLOYEES. Semua kolom digabung jadi satu dengan tanda koma sebagai pemisah, kemudian beri judul “OUTPUT”

OUTPUT

```
-----
7369, SMITH, CLERK, 7902, 17-DEC-80, 800, , 20
7499, ALLEN, SALESMAN, 7698, 20-FEB-81, 1600, 300, 30
7521, WARD, SALESMAN, 7698, 22-FEB-81, 1250, 500, 30
7566, JONES, MANAGER, 7839, 02-APR-81, 2975, , 20
7654, MARTIN, SALESMAN, 7698, 28-SEP-81, 1250, 1400, 30
.....
```

6. Temukan 4 (empat) kesalahan pada statement SELECT berikut :

```
select empno, ename
      salary x 12
from emp;
```



# Sesion VII (Query Lanjutan)

Pada pertemuan kali ini kita akan membahas lebih dalam lagi mengenai Query

Perhatikan Skema Database HR

➤ **Contoh : Kondisi pencarian dengan batas (range)**

Tampilkan seluruh Pegawai dan pendapatanya yang berpenghasilan antara 4800 dan 6000 dari table EMPLOYEES

```
select "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",
       "EMPLOYEES"."LAST_NAME" as "LAST_NAME",
       "EMPLOYEES"."SALARY" as "SALARY"
from   "EMPLOYEES" "EMPLOYEES"
where  "EMPLOYEES"."SALARY" between 4800 and 6000
```

FIRST_NAME	LAST_NAME	SALARY
Bruce	Ernst	6000
David	Austin	4800
Valli	Pataballa	4800
Kevin	Mourgos	5800
Pat	Fay	6000

➤ **Contoh : Himpunan Anggota (Set Membership)**

Menampilkan seluruh nama pegawai yang memiliki bergaji 2100, 2200, 2400

```
select "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",
       "EMPLOYEES"."LAST_NAME" as "LAST_NAME",
       "EMPLOYEES"."SALARY" as "SALARY"
from   "EMPLOYEES" "EMPLOYEES"
where  "EMPLOYEES"."SALARY" IN(2100,2200,2400)
```

FIRST_NAME	LAST_NAME	SALARY
James	Landry	2400
Steven	Markle	2200
TJ	Olson	2100
Ki	Gee	2400
Hazel	Philtanker	2200

Tampilkan nama seluruh Pegawai yang jabatannya adalah President atau Administration Vice President dari table **EMPLOYEES**

```
select "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",
       "EMPLOYEES"."LAST_NAME" as "LAST_NAME",
       "EMPLOYEES"."JOB_ID" as "JOB_ID"
from   "EMPLOYEES" "EMPLOYEES"
where  "EMPLOYEES"."JOB_ID" IN (select "JOBS"."JOB_ID" from
"JOBS" "JOBS" where "JOBS"."JOB_TITLE" ='President' or
"JOBS"."JOB_TITLE" ='Administration Vice President')
```

FIRST_NAME	LAST_NAME	JOB_ID
Steven	King	AD_PRES
Neena	Kochhar	AD_VP
Lex	De Haan	AD_VP

➤ **Contoh : Penyesuaian bentuk (Pattern Matching)**

Menampilkan Nama Kota yang berawalan **T** dari table **Location**

```
select "LOCATIONS"."CITY" as "CITY"
from   "LOCATIONS" "LOCATIONS"
where  "LOCATIONS"."CITY" like 'T%'
```

CITY
Tokyo
Toronto

- SQL mempunyai dua simbol pattern matching :
- %: rangkaian dari nol atau lebih karakter.
- \_ (garis bawah): satu karakter tunggal.
- LIKE 'T%' berarti rangkaian karakter yang berawalan 'T', panjang string tidak ditentukan.

➤ **Ekspresi Reguler** memungkinkan Anda untuk mencari pola data dalam string dengan menggunakan sintaks standar konvensi.

Anda menentukan **ekspresi reguler** dengan metakarakter dan literal. Metakarakter adalah operator yang melakukan algoritma pencarian. Literal adalah karakter yang Anda cari.

Kalimat fungsi dan kondisi meliputi **REGEXP\_INSTR**, **REGEXP\_LIKE**, **REGEXP\_REPLACE**, dan **REGEXP\_SUBSTR**.

**Contoh 1.**

**REGEXP\_LIKE** : menyeleksi baris pada job\_id, yang diawali dengan *ac*, *fi*, *mk*, atau *st* kemudian diikuti dengan karakter *\_m* dan diakhiri dengan *an* atau *gr*, format '*i*' membuat tidak case-insensitive

```
SELECT employee_id, job_id FROM employees
WHERE REGEXP_LIKE (job_id, '[ac|fi|mk|st]_m[an|gr]', 'i');
```

EMPLOYEE_ID	JOB_ID
108	FI_MGR
120	ST_MAN
121	ST_MAN
122	ST_MAN
123	ST_MAN
124	ST_MAN
145	SA_MAN
146	SA_MAN
147	SA_MAN
148	SA_MAN

## Contoh 2

**REGEXP\_REPLACE** digunakan untuk menggantikan

- Nomor telepon dengan format "nnn.nnn.nnnn" dengan Tanda kurung, spasi, dan tanda hubung untuk menghasilkan format ini "(nnn) nnn-nnnn"
- Digit (0-9) dilambangkan dengan metacharacter `[[:digit:]]`
- metacharacter (n) menetapkan jumlah kejadian tetap
- lambang `'\'` adalah karakter yang digunakan untuk menjalankan fungsi sendiri sehingga metacharacter berikutnya
- Dalam pernyataan diperlakukan sebagai literal, seperti `\.`; Jika tidak,
- Metacharacter. melambangkan setiap karakter dalam ekspresi

```
SELECT phone_number, REGEXP_REPLACE( phone_number,
'([[:digit:]]{3})\.([[:digit:]]{3})\.([[:digit:]]{4})', '(1) 2-3')
"Phone Number" FROM employees;
```

PHONE_NUMBER	Phone Number
515.123.4567	(515) 123-4567
515.123.4568	(515) 123-4568
515.123.4569	(515) 123-4569
590.423.4567	(590) 423-4567
590.423.4568	(590) 423-4568
590.423.4569	(590) 423-4569
590.423.4560	(590) 423-4560
590.423.5567	(590) 423-5567
515.124.4569	(515) 124-4569
515.124.4169	(515) 124-4169

## Contoh 3

**REGEXP\_REPLACE** digunakan untuk menggantikan

- Nomor telepon format "nnn.nnn.nnnn.nnnnnn" Dengan format "+ nnn-nn-nnnn-nnnnnn"

```
SELECT phone_number, REGEXP_REPLACE( phone_number,
'([[:digit:]]{3})\.([[:digit:]]{3})\.([[:digit:]]{4})', '(1) 2-3')
"Phone Number" FROM employees;
```

PHONE_NUMBER	Phone Number
515.123.4567	515.123.4567
515.123.4568	515.123.4568
515.123.4569	515.123.4569
590.423.4567	590.423.4567
590.423.4568	590.423.4568
590.423.4569	590.423.4569
590.423.4560	590.423.4560
590.423.5567	590.423.5567
515.124.4569	515.124.4569
515.124.4169	515.124.4169

#### **Contoh 4**

**REGEXP\_SUBSTR** mengembalikan substring

- Terdiri dari satu atau lebih kemunculan angka dan garis
- metacharacter '+' menentukan beberapa kejadian dalam '[:digit:]-]+'

```
SELECT street_address, REGEXP_SUBSTR(street_address, '[:digit:]-]+' , 1, 1)
"Street numbers" FROM locations;
```

STREET_ADDRESS	Street numbers
1297 Via Cola di Rie	1297
93091 Calle della Testa	93091
2017 Shinjuku-ku	2017
9450 Kamiya-cho	9450
2014 Jabberwocky Rd	2014
2011 Interiors Blvd	2011
2007 Zagora St	2007
2004 Charade Rd	2004
147 Spadina Ave	147
6092 Boxwood St	6092

#### **Contoh 5**

**REGEXP\_INSTR** mulai mencari pada karakter pertama

- Dalam string dan mengembalikan posisi awal (default) dari kedua
- Terjadinya satu atau lebih karakter non-kosong
- REGEXP\_INSTR mengembalikan 0 jika tidak ditemukan
- metacharacter '^' menunjukkan TIDAK, seperti di ruang TIDAK [^]

```
SELECT street_address, REGEXP_INSTR(street_address, '[^ ]+', 1, 1)
"Position of 2nd block" FROM locations;
```

STREET_ADDRESS	Street numbers
1297 Via Cola di Rie	1297
93091 Calle della Testa	93091
2017 Shinjuku-ku	2017
9450 Kamiya-cho	9450
2014 Jabberwocky Rd	2014
2011 Interiors Blvd	2011
2007 Zagora St	2007
2004 Charade Rd	2004
147 Spadina Ave	147
6092 Boxwood St	6092

➤ **Contoh : Kondisi pencarian NULL**

Menampilkan nama jalan dan nama Kota yang provinsinya berisi data NULL

```
select  "LOCATIONS"."STREET_ADDRESS" as "STREET_ADDRESS",
        "LOCATIONS"."CITY" as "CITY"
from    "LOCATIONS" "LOCATIONS" where
"LOCATIONS"."STATE_PROVINCE" is NULL
```

STREET_ADDRESS	CITY
1297 Via Cola di Rie	Roma
93091 Calle della Testa	Venice
9450 Kamiya-cho	Hiroshima
40-5-12 Laogianggen	Beijing
198 Clementi North	Singapore
8204 Arthur St	London

➤ **Contoh : Ordering kolom tunggal**

Tampilkan semua informasi dari table **REGIONS** dengan urutan berdasarkan nama **REGION\_NAME** secara Ascending

```
select  "REGIONS"."REGION_ID" as "REGION_ID",
        "REGIONS"."REGION_NAME" as "REGION_NAME"
from    "REGIONS" "REGIONS"
order by REGIONS.REGION_NAME ASC
```

REGION_ID	REGION_NAME
2	Americas
3	Asia
1	Europe
4	Middle East and Africa

- Secara default Pengurutan bersifat Ascending
- Untuk mengubahnya secara Descending ubah **ASC** menjadi **DESC**

➤ Contoh : Ordering multiple kolom

Tampilkan Employee\_ID, JOB\_ID, tanggal mulai dan tanggal berakhir berdasarkan Employee\_ID

```
select  "JOB_HISTORY"."EMPLOYEE_ID" as "EMPLOYEE_ID",
        "JOB_HISTORY"."JOB_ID" as "JOB_ID",
        "JOB_HISTORY"."START_DATE" as "START_DATE",
        "JOB_HISTORY"."END_DATE" as "END_DATE"
from    "JOB_HISTORY" "JOB_HISTORY"
order by JOB_HISTORY.EMPLOYEE_ID
```

EMPLOYEE_ID	JOB_ID	START_DATE	END_DATE
101	AC_ACCOUNT	21-09-1989	27-10-1993
101	AC_MGR	28-10-1993	15-03-1997
102	IT_PROG	13-01-1993	24-07-1998
114	ST_CLERK	24-03-1998	31-12-1999
122	ST_CLERK	01-01-1999	31-12-1999
176	SA_REP	24-03-1998	31-12-1998
176	SA_MAN	01-01-1999	31-12-1999
200	AD_ASST	17-09-1987	17-06-1993
200	AC_ACCOUNT	01-07-1994	31-12-1998
201	MK_REP	17-02-1996	19-12-1999

- Dari contoh diatas terdapat empat field. Untuk menyusun Employee\_ID berdasarkan *Start\_Date* maka harus dispesifikasikan minor order sbb :

```
select  "JOB_HISTORY"."EMPLOYEE_ID" as "EMPLOYEE_ID",
        "JOB_HISTORY"."JOB_ID" as "JOB_ID",
        "JOB_HISTORY"."START_DATE" as "START_DATE",
        "JOB_HISTORY"."END_DATE" as "END_DATE"
from    "JOB_HISTORY" "JOB_HISTORY"
order by JOB_HISTORY.EMPLOYEE_ID , "JOB_HISTORY"."START_DATE" Desc
```

EMPLOYEE_ID	JOB_ID	START_DATE	END_DATE
101	AC_MGR	28-10-1993	15-03-1997
101	AC_ACCOUNT	21-09-1989	27-10-1993
102	IT_PROG	13-01-1993	24-07-1998
114	ST_CLERK	24-03-1998	31-12-1999
122	ST_CLERK	01-01-1999	31-12-1999
176	SA_MAN	01-01-1999	31-12-1999
176	SA_REP	24-03-1998	31-12-1998
200	AC_ACCOUNT	01-07-1994	31-12-1998
200	AD_ASST	17-09-1987	17-06-1993
201	MK_REP	17-02-1996	19-12-1999

Perahtikan perubahan susunan pada Employee\_ID yang sama

## Latihan Bab 7

---

1. Tampilkan nama dari EMPLOYEES yang berpendapatan di bawah 6000 dan berprofesi sebagai IT\_prog
2. Tampilkan nama dari EMPLOYEES yang berpendapatan antara 6900 hingga 7000
3. Tampilka urutan Pekerjaan dengan pendapatan terendah ke tertinggi dari table JOBS
4. Siapakah employee (first\_name) yang hire\_date nya diantara 01-JAN-00 dan 30-APR-00 dan first\_namenya diawali dengan huruf S ?
5. Department\_name apa sajakah yang department\_id nya diantara 40 dan 90 ?

## Sesion VIII (Query Lanjutan)

---

Fungsi agregat adalah fungsi-fungsi yang mengambil kumpulan (collection) suatu himpunan data atau beberapa himpunan data dan mengembalikan dalam bentuk nilai tunggal.

Terdapat 5 fungsi agregasi (agregat) baku, yaitu:

1. AVG : digunakan untuk mencari nilai rata-rata pada suatu kolom data
2. COUNT : digunakan untuk mencari jumlah record data row (jumlah baris data yang dihasilkan dari query/banyaknya data).
3. MAX : digunakan untuk mencari nilai data paling besar (Maximum).
4. MIN : digunakan untuk mencari nilai data paling kecil (minimum)
5. SUM : digunakan untuk mencari nilai jumlah total pada suatu Kolom data.

Contoh penggunaan Fungsi agregat

Menampilkan rata-rata gaji seluruh pegawai

```
select avg(salary) from employees;
```

AVG(SALARY)
6461.68224299065420560747663551401869159

Menampilkan jumlah gaji seluruh pegawai

```
select sum(salary) from employees;
```

SUM(SALARY)
691400

Menampilkan gaji terendah seluruh pegawai

```
select min(salary)from employees;
```

MIN(SALARY)
2100

Menampilkan gaji tertinggi seluruh pegawai

```
select max(salary) from employees;
```

MAX(SALARY)
24000



Menampilkan jumlah seluruh pegawai

```
select count(*)from employees;
```

<b>COUNT(*)</b>
107

## Membuat Group Data

Ada keadaan penggunaan fungsi agregat untuk menghasilkan beberapa record data sekaligus berdasarkan kondisi khusus atau group dari suatu kolom tertentu.

Maka dapat digunakan klausa GROUP BY.

Membagi rows (baris-baris) dalam tabel menjadi group-group data yang lebih kecil dengan klausa GROUP BY.

```
SELECT column, group_function(column)  
  
FROM table  
  
[WHERE condition]  
  
[GROUP BY group_by_expression];
```

Menampilkan nomor pekerjaan dan jumlah gaji berdasarkan nomor pekerjaan

```
select job_id, sum(salary)  
  
from employees  
  
group by job_id;
```

<b>JOB_ID</b>	<b>SUM(SALARY)</b>
IT_PROG	28800
AC_MGR	12000
AC_ACCOUNT	8300
ST_MAN	36400
PU_MAN	11000
AD_ASST	4400
AD_VP	34000
SH_CLERK	64300
FI_ACCOUNT	39600
FI_MGR	12000

## Menggunakan GROUP BY Clause pada Multiple Columns

Menampilkan nomor departemen, nomor pekerjaan dan jumlah gaji berdasarkan departemen dan nomor pekerjaan.

```
select department_id, job_id, sum(salary)
from employees
group by department_id, job_id;
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
110	AC_ACCOUNT	8300
90	AD_VP	34000
50	ST_CLERK	55700
80	SA_REP	243500
50	ST_MAN	36400
80	SA_MAN	61000
110	AC_MGR	12000
90	AD PRES	24000
60	IT_PROG	28800
100	FI_MGR	12000
.		

## Membuat Pembatasan Group Data

Fungsi GROUP BY dapat dibuat pembatasan dari data yang akan dihasilkan dengan menggunakan fungsi HAVING.

Dengan klausa HAVING dapat membatasi groups data:

1. Rows (baris-baris) akan di group.
2. Fungsi group dapat diaplikasikan.
3. Hasil group data yang match/sesuai dari klausa HAVING akan ditampilkan.

```
SELECT column, group_function
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
```

Contoh :

Menampilkan nomor pekerjaan yang mempunyai jumlah gaji lebih dari 15000.

```
select job_id, sum(salary)  
  
from employees  
  
group by job_id  
  
having sum(salary) > 15000;
```

JOB_ID	SUM(SALARY)
IT_PROG	28800
ST_MAN	36400
AD_VP	34000
SH_CLERK	64300
FI_ACCOUNT	39600
SA_MAN	61000
AD_PRES	24000
SA_REP	250500
ST_CLERK	55700

### Soal Latihan Bab 8 :

---

1. Tampilkan nomor departemen dan rata-rata gaji setahun pegawai untuk tiap-tiap department, dengan rata-rata gaji setahun tersebut antara \$10000 dan \$50000.
2. Tampilkan nomor departemen, rata-rata gaji dan total gaji setahun pegawai untuk tiap-tiap manager departemen yang memimpin, dan yang memiliki total gaji setahun tadi > 50 juta
3. Tampilkan nilai tertinggi, terendah, jumlah dan rata-rata gaji dari seluruh pegawai.
4. Tampilkan nilai tertinggi, terendah, jumlah dan rata-rata gaji pada tiap-tiap jenis pekerjaan yang ada.
5. Tampilkan nama pekerjaan dan jumlah pegawai yang bekerja pada tiap-tiap pekerjaan tersebut.

# Sesion IX (Ekspresi Query)

## Ekspresi Kondisional

Ekspresi Kondisional menggunakan logika IF-THEN-ELSE di dalam SQL Statement.

Digunakan dua cara, yaitu :

- Ekspresi CASE
- Fungsi DECODE

## Ekspresi CASE

Ekspresi CASE mempunyai bentuk umum sebagai berikut :

```
CASE expr WHEN comparison_expr1 THEN return_expr1
        [WHEN comparison_expr2 THEN return_expr2
        WHEN comparison_exprn THEN return_exprn
        ELSE else_expr]
END
```

Contoh :

```
select first_name, last_name, job_id, salary,
       case job_id when 'IT_PROG' then 1.10*salary
                  when 'ST_CLERK' then 1.20*salary
                  when 'SA_REP' then 1.30*salary
                  else salary end "revised_salary"
from employees;
```

FIRST_NAME	LAST_NAME	JOB_ID	SALARY	revised_salary
Steven	King	AD_PRES	24000	24000
Neena	Kochhar	AD_VP	17000	17000
Lex	De Haan	AD_VP	17000	17000
Alexander	Hunold	IT_PROG	9000	9900
Bruce	Ernst	IT_PROG	6000	6600
David	Austin	IT_PROG	4800	5280
Valli	Pataballa	IT_PROG	4800	5280
Diana	Lorentz	IT_PROG	4200	4620
Nancy	Greenberg	FI_MGR	12000	12000
Daniel	Faviet	FI_ACCOUNT	9000	9000
More than 10 rows available. Increase rows selector to view more rows.				

## Fungsi DECODE

Fungsi DECODE menyediakan fasilitas pencocokan seperti yang dikerjakan oleh CASE atau IF-THEN-ELSE.

Sintak (penulisan) fungsi DECODE :

```
DECODE(col/expression, search1, result1  
      [, search2, result2, ..., ]  
      [, default])
```

Contoh :

```
select first_name, last_name, job_id, salary,  
       decode (job_id, 'IT_PROG', 1.10*salary,  
               'ST_CLERK', 1.20*salary,  
               'SA_REP', 1.30*salary,  
               salary)  
       revised_salary  
from employees;
```

FIRST_NAME	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
Steven	King	AD_PRES	24000	24000
Neena	Kochhar	AD_VP	17000	17000
Lex	De Haan	AD_VP	17000	17000
Alexander	Hunold	IT_PROG	9000	9900
Bruce	Ernst	IT_PROG	6000	6600
David	Austin	IT_PROG	4800	5280
Valli	Pataballa	IT_PROG	4800	5280
Diana	Lorentz	IT_PROG	4200	4620
Nancy	Greenberg	FI_MGR	12000	12000
Daniel	Faviet	FI_ACCOUNT	9000	9000
More than 10 rows available. Increase rows selector to view more rows.				

## NAMA ALIAS

Memberikan nama alias pada keluran kolom menggunakan **AS**.

Contoh :

Menampilkan jumlah gaji seluruh pegawai yang di nama aliaskan sebagai total\_salary.

```
select sum(salary) as total_salary  
from employees;
```

TOTAL_SALARY
691400

Menampilkan jumlah gaji (di aliaskan sebagai total\_salary) per nama jabatan pegawai.

```
select j.job_title, sum(e.salary) as total_salary
from employees e, jobs j
where e.job_id = j.job_id
group by j.job_title;
```

JOB_TITLE	TOTAL_SALARY
Programmer	28800
Purchasing Clerk	13900
Sales Representative	250500
Public Relations Representative	10000
Accounting Manager	12000
Administration Vice President	34000
Stock Manager	36400
Marketing Representative	6000
President	24000
Finance Manager	12000

## Soal bab 9

1. Tampilkan jumlah gaji (di aliaskan sebagai total\_salary) per nama jabatan pegawai dan jumlah gajinya di atas 30000.
2. Tampilkan nama departemen dan jumlah departemen (di aliaskan sebagai total) yang berada di negara Inggris.
3. Buat query yang menampilkan nama dan jumlah komisi, jika komisi sama dengan NULL ganti dengan keterangan “Tidak ada Komisi” dan beri judul ‘Komisi’.

ENAME	KOMISI
SMITH	Tidak ada komisi
ALLEN	300
WARD	500
JONES	Tidak ada komisi
MARTIN	1400
BLAKE	Tidak ada komisi
CLARK	Tidak ada komisi

4. Tampilkan nama pegawai dan jumlah gaji yang ditampilkan dengan symbol ‘\*’, tiap satu ‘\*’ mewakili ratusan dollar.
5. Gunakan fungsi DECODE untuk menampilkan nilai grade semua pegawai berdasarkan pada nilai kolom JOB\_ID, dengan ketentuan sebagai berikut :

Job	Grade
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
Selain diatas	0

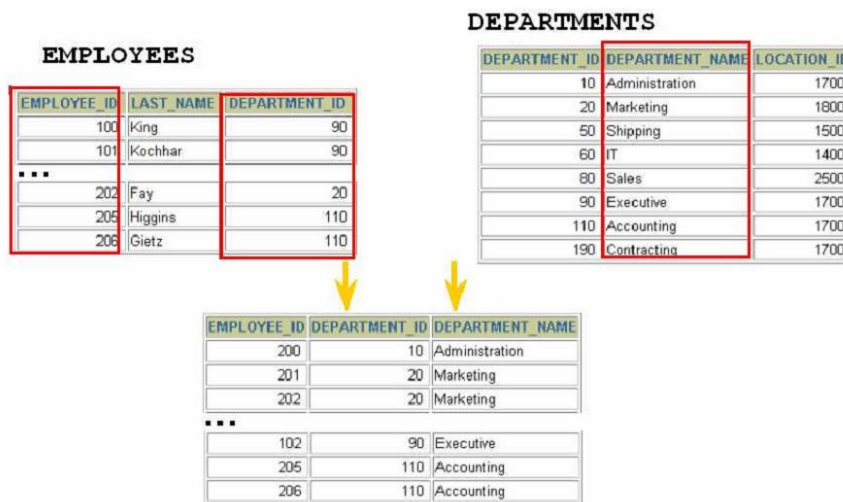
# Sesion X (Query Antar Tabel)

Query yang melibatkan lebih dari sebuah tabel.

## Perhatikan Skema Database HR

Seringkali kita perlu menggunakan data yang berasal dari banyak table, tidak hanya berasal dari satu table saja, semisal :

- Nomor pegawai (employee\_id) hanya ada di table pegawai (EMPLOYEES)
- Nomor department (department\_id) ada di table pegawai (EMPLOYEES) dan table department (DEPARTMENTS).
- Nama departemen (Department\_name) hanya ada di table department (DEPARTMENTS)



Contoh :

## Tanpa memakai alias

Menampilkan nama pegawai, pekerjaannya, batas gajinya.

```
select employees.first_name, employees.last_name, jobs.job_title, jobs.min_salary,
       jobs.max_salary
from employees, jobs
where employees.job_id = jobs.job_id;
```

FIRST_NAME	LAST_NAME	JOB_TITLE	MIN_SALARY	MAX_SALARY
Steven	King	President	20000	40000
Neena	Kochhar	Administration Vice President	15000	30000
Lex	De Haan	Administration Vice President	15000	30000
Alexander	Hunold	Programmer	4000	10000
Bruce	Ernst	Programmer	4000	10000
David	Austin	Programmer	4000	10000
Valli	Pataballa	Programmer	4000	10000
Diana	Lorentz	Programmer	4000	10000
Nancy	Greenberg	Finance Manager	8200	16000
Daniel	Faviet	Accountant	4200	9000

## Memakai alias

Menampilkan nama pegawai, pekerjaannya, batas gajinya.

```
select e.first_name, e.last_name, j.job_title, j.min_salary, j.max_salary  
from employees e, jobs j where e.job_id = j.job_id;
```

FIRST_NAME	LAST_NAME	JOB_TITLE	MIN_SALARY	MAX_SALARY
Steven	King	President	20000	40000
Neena	Kochhar	Administration Vice President	15000	30000
Lex	De Haan	Administration Vice President	15000	30000
Alexander	Hunold	Programmer	4000	10000
Bruce	Ernst	Programmer	4000	10000
David	Austin	Programmer	4000	10000
Valli	Pataballa	Programmer	4000	10000
Diana	Lorentz	Programmer	4000	10000
Nancy	Greenberg	Finance Manager	8200	16000
Daniel	Faviet	Accountant	4200	9000

More than 10 rows available. Increase rows selector to view more rows.

Soal Bab 10 :

- 1 . Tampilkan nama pegawai yang pekerjaannya sebagai programmer
- 2 . Tampilkan nama departemen yang berada di negara Amerika dan Itali
- 3 . Tampilkan nama pegawai yang pekerjaannya selesai pada bulan Desember 1999
- 4 . Tampilkan nama pegawai, nomor department dan nama department dari semua pegawai
- 5 . Tampilkan nama pegawai dan nama department untuk semua pegawai yang memiliki huruf 'A' pada namanya
- 6 . Buat query untuk menampilkan nama pegawai dan nomer pegawai, nama manager dan nomer pegawai dari manager
- 7 . Tampilkan nama pegawai dan nama department untuk semua pegawai yang memiliki huruf 'A' pada namanya.



# Sesion XI (Penggunaan Fungsi)

A. Pada SQL banyak sekali di sediakan fungsi – fungsi yang mebantukita dalam mengolah data waktu atau tanggal. Contoh :

- Menampilkan tanggal ,bulan, dan tahun dari sistem

```
SELECT EXTRACT(YEAR FROM SYSDATE) AS "tahun", EXTRACT(MONTH  
FROM SYSDATE) as "Bulan",  
EXTRACT(DAY FROM SYSDATE) as "tanggal sekarang" FROM DUAL;
```

tahun	Bulan	tanggal sekarang
2010	2	22

- Menmpilkan masa kerja pegawai yg bernama Steven king dari table **EMPLOYEE**

```
select "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",  
      "EMPLOYEES"."LAST_NAME" as "LAST_NAME" ,  
TRUNC(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)) "Months Employed"  
from "EMPLOYEES" "EMPLOYEES" WHERE  
"EMPLOYEES"."FIRST_NAME"='Steven' and  
"EMPLOYEES"."LAST_NAME"='King'
```

FIRST_NAME	LAST_NAME	Months Employed
Steven	King	272

- Menampilkan tahun kerja steven King

```
SELECT "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",  
      "EMPLOYEES"."LAST_NAME" as "LAST_NAME" ,  
EXTRACT(YEAR FROM hire_date) "Year Hired" FROM "EMPLOYEES"  
"EMPLOYEES" WHERE "EMPLOYEES"."FIRST_NAME"='Steven' and  
"EMPLOYEES"."LAST_NAME"='King'
```

FIRST_NAME	LAST_NAME	Year Hired
Steven	King	1987

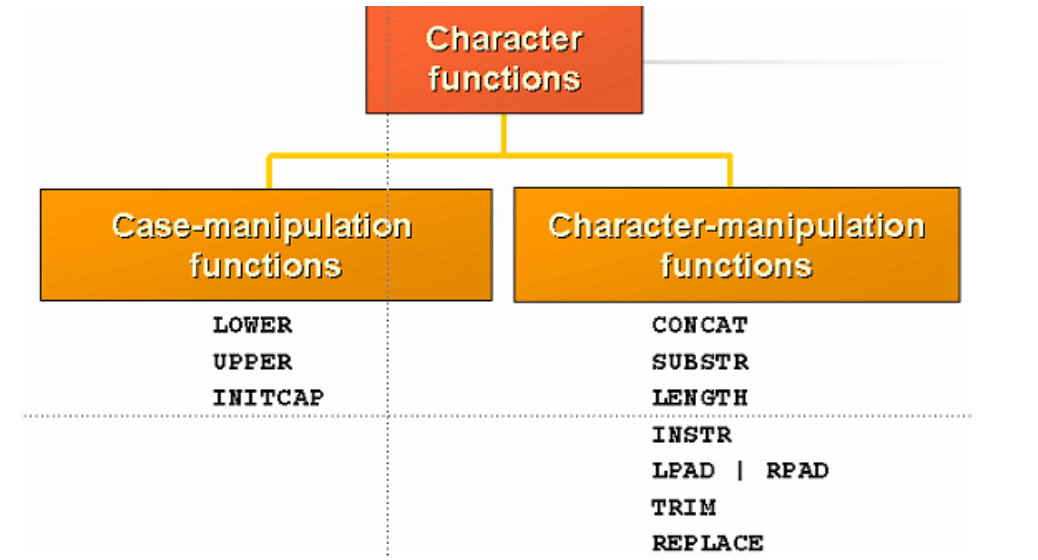
- Menambahkan 3 bulan masa kerja steven king

```
SELECT "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",  
      "EMPLOYEES"."LAST_NAME" as "LAST_NAME" ,  
      hire_date, ADD_MONTHS(hire_date, 3) from"EMPLOYEES" WHERE  
"EMPLOYEES"."FIRST_NAME"='Steven' and  
"EMPLOYEES"."LAST_NAME"='King';
```

FIRST_NAME	LAST_NAME	HIRE_DATE	ADD_MONTHS(HIRE_DATE,3)
Steven	King	17-JUN-87	17-SEP-87

B. Selanjutnya kita akan membahas Fungsi yang berhubungan dengan pengolahan String

### Fungsi Karakter



#### Fungsi karakter terbagi menjadi :

- Fungsi konversi, yaitu LOWER, UPPER, INITCAP
- Fungsi manipulasi : CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM, REPLACE.

#### Manipulasi Fungsi character

Function	Result
CONCAT('Hello' , 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
TRIM('H' FROM 'HelloWorld')	elloWorld

## Contoh 1

Menampilkan Email Stephen King Dengan huruf kecil semua

A. Sebelumnya

FIRST_NAME	LAST_NAME	EMAIL
Steven	King	SKING

```
SELECT "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",  
"EMPLOYEES"."LAST_NAME" as "LAST_NAME" ,lower("EMPLOYEES"."EMAIL") as  
"EMAIL" from"EMPLOYEES" WHERE "EMPLOYEES"."FIRST_NAME"='Steven' and  
"EMPLOYEES"."LAST_NAME"='King';
```

B. sesudah

FIRST_NAME	LAST_NAME	EMAIL
Steven	King	sking

## Contoh 2

```
SELECT employee_id , CONCAT(first_name, last_name) AS  
Nama,job_id,LENGTH(CONCAT(first_name, last_name)) as "Panjangnya nama",  
INSTR(last_name,'a') "Mengandung char 'a'?"  
from employees  
where SUBSTR(job_id,4)='REP';
```

EMPLOYEE_ID	NAMA	JOB_ID	PAnjangnya nama	Mengandung char 'a'?
150	PeterTucker	SA_REP	11	0
151	DavidBernstein	SA_REP	14	0
152	PeterHall	SA_REP	9	2
153	ChristopherOlsen	SA_REP	16	0
154	NanetteCambrault	SA_REP	16	2
155	OliverTuvault	SA_REP	13	4
156	JanetteKing	SA_REP	11	0
157	PatrickSully	SA_REP	12	0
158	AllanMcEwen	SA_REP	11	0
159	LindseySmith	SA_REP	12	0

## Latihan Bab 11

1. Tampilkan selisih masa kerja untuk setiap pegawai dari tabel JOB\_HISTORY
2. Kenapa Syntax : **select \* from "REGIONS" where "REGION\_NAME" ='asia'** tidak menghasilkan baris ?
3. Tampilkan nama pegawai yang di sewa pada bulan juni
4. Cari nama pegawai yang paling pertama di sewa
5. Cari id pegawai yang paling terlama di sewa

## Sesion XII (Query Multi-Table)

SQL tidak hanya menyediakan mekanisme query dan operasi modifikasi database saja, tetapi SQL juga menyediakan mekanisme untuk menggabungkan(join) relasi-relasi.

Saat data yang dibutuhkan berasal lebih dari satu table, maka kondisi join dibutuhkan. Umumnya dalam men-join table berdasarkan pada kolom yang bersesuaian Primary Key dari table-1 dengan Foreign Key dari table-2, atau yang disebut dengan join atau equi-join.

- Inner Join atau Simple Join : adalah bentuk kondisi join dimana nilai relasi yang terjadi antar dua table(binary relation) adalah sama (terdapat hubungan antara Primary Key dan Foreign Key)

```
SELECT employees.employee_id, employees.last_name,  
employees.department_id, departments.department_id,  
departments.location_id FROM employees, departments  
WHERE employees.department_id = departments.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
100	King	90	90	1700
101	Kochhar	90	90	1700
102	De Haan	90	90	1700
103	Hunold	60	60	1400
104	Ernst	60	60	1400
105	Austin	60	60	1400
106	Pataballa	60	60	1400
107	Lorentz	60	60	1400
108	Greenberg	100	100	1700
109	Faviet	100	100	1700

- Memilih data dari dua table dengan *NATURAL JOIN* Syntax

```
SELECT e.employee_id, e.last_name, e.first_name, e.manager_id, department_id,  
d.department_name, d.manager_id FROM employees e  
JOIN departments d USING (department_id);
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
101	Kochhar	Neena	90	Executive	100
102	De Haan	Lex	90	Executive	100
104	Ernst	Bruce	60	IT	103
105	Austin	David	60	IT	103
106	Pataballa	Valli	60	IT	103
107	Lorentz	Diana	60	IT	103
109	Faviet	Daniel	100	Finance	108
110	Chen	John	100	Finance	108
111	Sciarra	Ismael	100	Finance	108
112	Urman	Jose Manuel	100	Finance	108

- Memilih data dari beberapa table dengan **JOIN USING Syntax**

#### 1. dua table

```
SELECT e.employee_id, e.last_name, e.first_name, e.manager_id, department_id,
d.department_name, d.manager_id FROM employees e
JOIN departments d USING (department_id);
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
100	King	Steven	-	90	Executive	100
101	Kochhar	Neena	100	90	Executive	100
102	De Haan	Lex	100	90	Executive	100
103	Hunold	Alexander	102	60	IT	103
104	Ernst	Bruce	103	60	IT	103
105	Austin	David	103	60	IT	103
106	Pataballa	Valli	103	60	IT	103
107	Lorentz	Diana	103	60	IT	103
108	Greenberg	Nancy	101	100	Finance	108
109	Faviet	Daniel	108	100	Finance	108

#### 2. tiga table

```
SELECT e.employee_id, e.last_name, e.first_name, e.manager_id, department_id, d.department_name,
d.manager_id, location_id, l.country_id FROM employees e JOIN departments d USING
(department_id) JOIN locations l USING (location_id);
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID	COUNTRY_ID
100	King	Steven	-	90	Executive	100	1700	US
101	Kochhar	Neena	100	90	Executive	100	1700	US
102	De Haan	Lex	100	90	Executive	100	1700	US
103	Hunold	Alexander	102	60	IT	103	1400	US
104	Ernst	Bruce	103	60	IT	103	1400	US
105	Austin	David	103	60	IT	103	1400	US
106	Pataballa	Valli	103	60	IT	103	1400	US
107	Lorentz	Diana	103	60	IT	103	1400	US
108	Greenberg	Nancy	101	100	Finance	108	1700	US
109	Faviet	Daniel	108	100	Finance	108	1700	US

- Memilih data dari beberapa table dengan **JOIN ON Syntax**

#### a. 2 tabel

```
SELECT e.employee_id, e.last_name, e.first_name, e.department_id,
d.department_name, d.manager_id FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE e.manager_id = 122;
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
133	Mallin	Jason	50	Shipping	121
134	Rogers	Michael	50	Shipping	121
135	Gee	Ki	50	Shipping	121
136	Philtanker	Hazel	50	Shipping	121
188	Chung	Kelly	50	Shipping	121
189	Dilly	Jennifer	50	Shipping	121
190	Gates	Timothy	50	Shipping	121
191	Perkins	Randall	50	Shipping	121

**b. 3 tabel**

```
SELECT e.employee_id, e.last_name, e.first_name, e.department_id,
       d.department_name, d.manager_id, d.location_id, l.country_id FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id
WHERE l.location_id = 1700;
```

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID	COUNTRY_ID
100	King	Steven	90	Executive	100	1700	US
101	Kochhar	Neena	90	Executive	100	1700	US
102	De Haan	Lex	90	Executive	100	1700	US
108	Greenberg	Nancy	100	Finance	108	1700	US
109	Faviet	Daniel	100	Finance	108	1700	US
110	Chen	John	100	Finance	108	1700	US
111	Sciarra	Ismael	100	Finance	108	1700	US
112	Urman	Jose Manuel	100	Finance	108	1700	US
113	Popp	Luis	100	Finance	108	1700	US
114	Raphaely	Den	30	Purchasing	114	1700	US

**c. Menggabungkan data dari table itu sendiri**

Contoh berikut akan menampilkan id pegawai dan nama belakang pegawai dengan id manager dan nama belakang manager pada table EMPLOYEE

```
SELECT e.employee_id emp_id, e.last_name emp_lastname, m.employee_id mgr_id,
       m.last_name mgr_lastname
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;
```

EMP_ID	EMP_LASTNAME	MGR_ID	MGR_LASTNAME
101	Kochhar	100	King
102	De Haan	100	King
103	Hunold	102	De Haan
104	Ernst	103	Hunold
105	Austin	103	Hunold
106	Pataballa	103	Hunold
107	Lorentz	103	Hunold
108	Greenberg	101	Kochhar
109	Faviet	108	Greenberg
110	Chen	108	Greenberg

- Memilih data dari beberapa table dengan **Outer Joins Syntax**

#### a. LEFT OUTER JOIN

```
SELECT e.employee_id, e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
100	King	90	Executive
101	Kochhar	90	Executive
102	De Haan	90	Executive
103	Hunold	60	IT
104	Ernst	60	IT
105	Austin	60	IT
106	Pataballa	60	IT
107	Lorentz	60	IT
108	Greenberg	100	Finance
109	Faviet	100	Finance

#### b. RIGHT OUTER JOIN

```
SELECT e.employee_id, e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
200	Whalen	10	Administration
201	Hartstein	20	Marketing
202	Fay	20	Marketing
114	Raphaely	30	Purchasing
115	Khoo	30	Purchasing
116	Baida	30	Purchasing
117	Tobias	30	Purchasing
118	Himuro	30	Purchasing
119	Colmenares	30	Purchasing
203	Mavris	40	Human Resources

### c. FULL OUTER JOIN

```
SELECT e.employee_id, e.last_name, e.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d ON (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
100	King	90	Executive
101	Kochhar	90	Executive
102	De Haan	90	Executive
103	Hunold	60	IT
104	Ernst	60	IT
105	Austin	60	IT
106	Pataballa	60	IT
107	Lorentz	60	IT
108	Greenberg	100	Finance
109	Faviet	100	Finance

## Latihan Bab 12

1. BuatSQL Query untuk menampilkan last name, department number, dan department name untuk semua pegawai!
2. Tampilkan semua job (job id) pegawai secara unik yang berada pada department 80 termasuk nama lokasinya!
3. Buat query yang menampilkan last name, nama department, location id dan kota dari semua pegawai yang memiliki komisi.
4. Tampilkan last name pegawai dan nama department untuk semua pegawai yang memiliki huruf 'a' pada last name
5. Buat query yang menampilkan last name, department number, department name untuk semua pegawai yang bekerja di kota Toronto.



# Sesion XIII (Sub Query)

---

Subquery memakai operator : IN, EXISTS, ANY, dan ALL.

Contoh :

## IN

Menampilkan nomor pegawai, nama lengkap yang tercatat pekerjaannya di job history.

```
select employee_id, first_name, last_name
from employees
where employee_id in (select employee_id from job_history);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
101	Neena	Kochhar
102	Lex	De Haan
114	Den	Raphaely
122	Payam	Kaufling
176	Jonathon	Taylor
200	Jennifer	Whalen
201	Michael	Hartstein

## EXISTS

Menampilkan nomor departemen dan nama departemen yang ada pegawainya terdaftar di tabel employees

```
select department_id, department_name
from departments
where exists (select * from employees
              where department_id = departments.department_id);
```

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
110	Accounting

## ANY

Menampilkan nama pegawai dan gaji yang gajinya terendah dari seluruh pegawai yang ada.

```
select first_name, last_name, salary
from employees
where salary <= all (select salary
                    from employees);
```

FIRST_NAME	LAST_NAME	SALARY
TJ	Olson	2100

## ALL

Menampilkan nama pegawai dan gaji terkecuali pegawai yang gajinya paling rendah.

```
select first_name, last_name, salary
from employees
where salary > any (select salary
                   from employees);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Neena	Kochhar	17000
Lex	De Haan	17000
John	Russell	14000
Karen	Partners	13500
Michael	Hartstein	13000
Nancy	Greenberg	12000
Alberto	Errazuriz	12000
Shelley	Higgins	12000
Lisa	Ozer	11500
More than 10 rows available. Increase rows selector to view more rows.		

## Subquery Insert, Update, Delete

Subquery dapat digunakan dalam statement INSERT :

```
INSERT INTO (SELECT employee_id, last_name, email, hire_date, job_id, salary, department_id
FROM employees
WHERE department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR', TO_DATE('07-JUN-99', 'DD-MON-RR'), 'ST_CLERK', 5000,
50);
```

1 row(s) inserted.

### Mengupdate dua kolom dengan subquery

Query berikut ini akan mengupdate job dan salary yang dimiliki oleh employee 114 supaya sama dengan job dan salary yang dimiliki oleh employee 205

```
UPDATE employees
SET job_id = (SELECT job_id
FROM employees
WHERE employee_id = 205),
salary = (SELECT salary
FROM employees
WHERE employee_id = 205)
WHERE employee_id = 114;
1 row updated.
```

### Menghapus Berdasarkan Baris pada Tabel yang lain

Subquery dapat digunakan dalam statement DELETE untuk menghapus baris pada suatu table berdasarkan data yang ada di table yang lain.

```
DELETE FROM employees
WHERE department_id =
(SELECT department_id
FROM departments
WHERE department_name LIKE '%Public%');
1 row deleted.
```

## TRIGGER

### PENGERTIAN TRIGGER

Trigger adalah blok PL/SQL atau prosedur yang berhubungan dengan table, view, skema atau database yang dijalankan secara implicit pada saat terjadi sebuah event.

Tipe dari trigger adalah :

- Application trigger : diaktifkan pada saat terjadi event yang berhubungan dengan sebuah aplikasi

- Database trigger : diaktifkan pada saat terjadi event yang berhubungan dengan data (seperti operasi DML) atau event yang berhubungan dengan sistem (semisal logon atau shutdown) yang terjadi pada sebuah skema atau database.

## PENGUNAAN TRIGGER

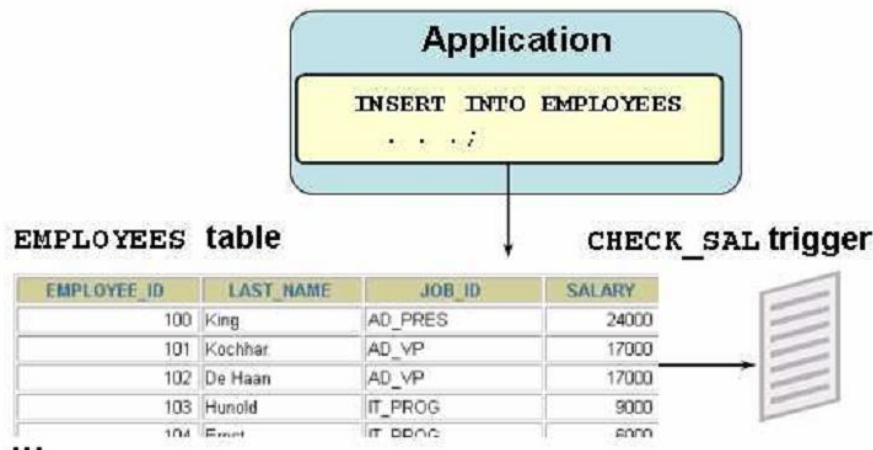
Trigger dibuat sesuai dengan keperluan. Ada kalanya trigger perlu dibuat, dan kadangkala tidak perlu dibuat.

Trigger perlu dibuat pada saat :

- Membentuk sebuah aksi tertentu terhadap suatu event
- Memusatkan operasi global Trigger tidak perlu dibuat, jika :
- Fungsionalitas yang diperlukan suatu ada pada Oracle server
- Duplikat atau sama dengan fungsi trigger yang lain.

Prosedur bisa dibuat dalam database, kemudian prosedur tersebut dipanggil pada trigger. Jika penggunaan trigger terlalu berlebihan, maka akan menyebabkan terjadi sifat ketidaktergantungan yang terlalu kompleks sehingga akan mempersulit pemeliharaan dari aplikasi yang besar.

Gambar berikut ini menunjukkan ilustrasi dari penggunaan trigger :



Pada gambar tersebut, database trigger CHECK\_SAL memeriksa nilai gaji pada saat suatu aplikasi mencoba untuk memasukkan baris baru ke dalam table EMPLOYEES. Nilai yang terletak pada jangkauan diluar kategori pekerjaan akan diabaikan.

Sintak penulisan dari database trigger, berisi komponen berikut :

1. Trigger timing :
  - a. Untuk tabel : BEFORE, AFTER
  - b. Untuk view : INSTEAD OF
2. Trigger event : INSERT, UPDATE atau DELETE
3. Nama tabel : yaitu nama tabel atau view yang berhubungan dengan trigger

4. Tipe trigger : Baris atau Pernyataan (statement)
5. klausa WHEN : untuk kondisi pembatasan
6. trigger body : bagian prosedur yang dituliskan pada trigger

## **KOMPONEN TRIGGER**

Komponen dari sebuah trigger ada 6 (enam), yaitu : trigger timing, trigger event, nama tabel, tipe trigger, klausa WHEN, dan trigger body. Berikut ini penjelasan komponen dari trigger.

Trigger timing adalah waktu kapan trigger diaktifkan. Ada tiga macam trigger timing, yaitu :

- BEFORE : trigger dijalankan sebelum DML event pada tabel
- AFTER : trigger dijalankan setelah DML event pada tabel
- INSTEAD OF : trigger dijalankan pada sebuah view.

Trigger event ada 3 kemungkinan : INSERT, UPDATE atau DELETE.

Pada saat trigger event UPDATE, kita dapat memasukkan daftar kolom untuk mengidentifikasi kolom mana yang berubah untuk mengaktifkan sebuah trigger (contoh : UPDATE OF salary ... ). Jika tidak ditentukan, maka perubahannya akan berlaku untuk semua kolom pada semua baris.

Tipe trigger ada 2 macam, yaitu :

- Statement : trigger dijalankan sekali saja pada saat terjadi sebuah event. Statement trigger juga dijalankan sekali, meskipun tidak ada satupun baris yang dipengaruhi oleh event yang terjadi.
- Row : trigger dijalankan pada setiap baris yang dipengaruhi oleh terjadinya sebuah event. Row trigger tidak dijalankan jika event dari trigger tidak berpengaruh pada satu baris pun.

Trigger body mendefinisikan tindakan yang perlu dikerjakan pada saat terjadinya event yang mengakibatkan sebuah trigger menjadi aktif.

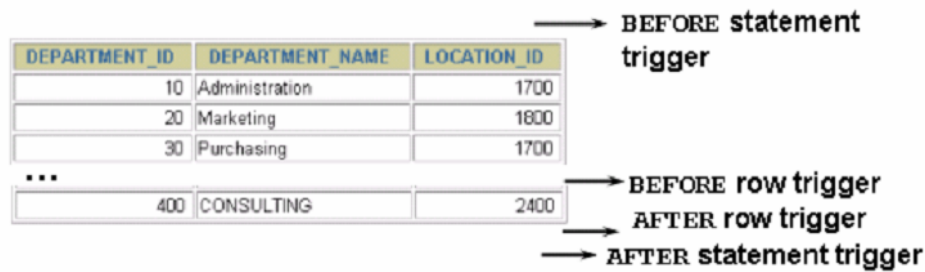
## **CONTOH PEMBUATAN TRIGGER**

Contoh berikut ini akan mengaktifkan sebuah trigger pada saat sebuah baris tunggal dimanipulasi pada tabel :

Misal diberikan perintah DML untuk menyisipkan baris baru ke dalam tabel sebagai berikut :

```
INSERT INTO departments (department_id, department_name, location_id)  
VALUES (400, 'CONSULTING', 2400);
```

Ilustrasi dari trigger timing untuk event tersebut adalah sebagai berikut :



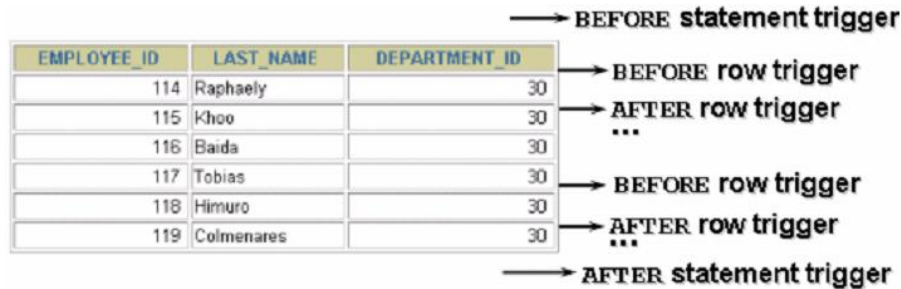
Jika DML statement berlaku untuk lebih dari satu baris yang ada pada tabel (multiple row), semisal :

### UPDATE employees

**SET salary = salary \* 1.1**

**WHERE department\_id = 30;**

Maka ilustrasi dari trigger timing untuk event tersebut adalah sebagai berikut :



### DML STATEMENT TRIGGER

Berikut ini sintak atau cara penulisan untuk pembuatan DML Statement trigger :

**CREATE [OR REPLACE] TRIGGER trigger\_name**

**timing**

**event1 [OR event2 OR event3]**

**ON table\_name**

**trigger\_body**

Berikut contoh pembuatan DML Statement trigger :

**CREATE OR REPLACE TRIGGER secure\_emp**

**BEFORE INSERT ON employees**

**BEGIN**

**IF (TO\_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR**

**(TO\_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN '08:00' AND '18:00')**

**THEN RAISE\_APPLICATION\_ERROR (-20500,'Penyisipan data pada table**

**EMPLOYEES hanya diperbolehkan selama jam kerja');**

**END IF;**

**END;**

Contoh trigger diatas akan membatasi penyisipan baris baru ke dalam table EMPLOYEES diperbolehkan hanya pada jam kerja mulai hari Senin sampai Jum'at. Jika user menyisipkan baris baru diluar ketentuan tersebut, misal pada hari Sabtu maka akan tampil pesan kesalahan.

Perintah berikut ini akan menguji trigger SECURE\_EMP dengan memberikan perintah SQL berikut ini pada jam diluar jam kerja, sebagai berikut :

```
INSERT INTO employees (employee_id, last_name,first_name, email, hire_date,  
job_id, salary, department_id)  
  
VALUES (300, 'Smith', 'Rob', 'RSMITH', SYSDATE,'IT_PROG', 4500, 60);
```

Perintah tersebut akan memberikan pesan kesalahan :

```
INSERT INTO employees (employee_id, last_name, first_name, email,  
*  
ERROR at line 1:  
ORA-20500: You may insert into EMPLOYEES table only during business hours.  
ORA-06512: at "PLSQL_SECURE_EMP", line 4  
ORA-04088: error during execution of trigger 'PLSQL_SECURE_EMP'
```

### **MENGGOMBINASIKAN EVENT PADA TRIGGER**

Beberapa event pada trigger bisa dikombinasikan dalam sebuah trigger dengan menggunakan predikat kondisional INSERTING, UPDATING dan DELETING.

Berikut ini akan dibuat trigger yang menggunakan predikat kondisional INSERTING, UPDATING dan DELETING untuk membatasi manipulasi data pada tabel EMPLOYEES hanya diperbolehkan pada setiap jam kerja mulai hari Senin sampai Jum'at.

```
BEFORE INSERT OR UPDATE OR DELETE ON employees  
BEGIN  
IF (TO_CHAR (SYSDATE,'DY') IN ('SAT','SUN')) OR  
   (TO_CHAR (SYSDATE, 'HH24') NOT BETWEEN '08' AND '18')  
THEN  
  IF DELETING THEN  
    RAISE_APPLICATION_ERROR (-20502,'You may delete from  
EMPLOYEES table only during business hours.');  
    ELSIF INSERTING THEN  
      RAISE_APPLICATION_ERROR (-20500,'You may insert into  
EMPLOYEES table only during business hours.');  
    ELSIF UPDATING ('SALARY') THEN  
      RAISE_APPLICATION_ERROR (-20503,'You may update  
SALARY only during business hours.');  
    ELSE  
      RAISE_APPLICATION_ERROR (-20504,'You may update  
EMPLOYEES table only during normal hours.');  
    END IF;  
  END IF;  
END;
```

## **ROW TRIGGER**

Berikut ini sintak atau cara penulisan untuk membuat Row Trigger :

**CREATE [OR REPLACE] TRIGGER trigger\_name**

**timing**

**event1 [OR event2 OR event3]**

**ON table\_name**

**[REFERENCING OLD AS old | NEW AS new]**

**FOR EACH ROW**

**[WHEN (condition)]**

**trigger\_body**

Contoh berikut ini akan dibuat row trigger dengan timing BEFORE untuk membatasi operasi DML pada table EMPLOYEES hanya diperbolehkan untuk pegawai yang memiliki kode pekerjaan 'AD\_PRES' dan 'AD\_VP' serta memiliki gaji kurang dari 15000.

**CREATE OR REPLACE TRIGGER restrict\_salary**

**BEFORE INSERT OR UPDATE OF salary ON employees**

**FOR EACH ROW**

**BEGIN**

**IF NOT (:NEW.job\_id IN ('AD\_PRES', 'AD\_VP'))**

**AND :NEW.salary > 15000**

**THEN**

**RAISE\_APPLICATION\_ERROR (-20202,'Employee**

**cannot earn this amount');**

**END IF;**

**END;**

Jika kita mencoba memberikan perintah SQL sebagai berikut, maka akan ditampilkan pesan kesalahan :

**UPDATE employees**

**SET salary = 15500**

**WHERE last\_name = ' Russell' ;**



## MENGGUNAKAN OLD DAN NEW QUALIFIERS

Pada Row Trigger, nilai dari kolom sebelum dan sesudah perubahan data dapat dirujuk dengan menggunakan OLD dan NEW qualifier. OLD dan NEW hanya digunakan pada Row Trigger. OLD dan NEW menggunakan prefiks (:) untuk pernyataan dalam perintah SQL. Jika qualifier ini terlibat dalam pembatasan kondisi pada klausa WHEN, maka tidak digunakan prefiks (:). Row triggers akan menurunkan unjuk kerja jika banyak dilakukan update pada table yang cukup besar.

Contoh Trigger berikut ini menggunakan OLD dan NEW qualifier pada Row Trigger :

```
CREATE OR REPLACE TRIGGER audit_emp_values  
AFTER DELETE OR INSERT OR UPDATE ON employees  
FOR EACH ROW  
BEGIN  
  
  INSERT INTO audit_emp_table (user_name, timestamp,  
    id, old_last_name, new_last_name, old_title,  
    new_title, old_salary, new_salary)  
  
  VALUES (USER, SYSDATE, :OLD.employee_id,  
    :OLD.last_name, :NEW.last_name, :OLD.job_id,  
    :NEW.job_id, :OLD.salary, :NEW.salary );  
  
END;
```

Untuk memeriksa hasil dari pembuatan trigger diatas, diberikan perintah SQL sebagai berikut :

```
INSERT INTO employees  
  
  (employee_id, last_name, job_id, salary, ...)  
  
VALUES (999, 'Temp emp', 'SA_REP', 1000, ...);
```

```
UPDATE employees  
  
SET salary = 2000, last_name = 'Smith'  
  
WHERE employee_id = 999;  
  
1 row created.  
1 row updated.
```

Hasil dari perintah SQL tersebut adalah akan disimpan record perubahan pada table AUDIT\_EMP\_TABLE sebagai hasil dari operasi Trigger :

**SELECT user\_name, timestamp, ... FROM audit\_emp\_table**

USER_NAME	TIMESTAMP	ID	OLD_LAST_N	NEW_LAST_N	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
PLSQL	28-SEP-01			Temp emp		SA_REP		1000
PLSQL	28-SEP-01	999	Temp emp	Smith	SA_REP	SA_REP	1000	2000

## **PENGUNAAN KLAUSA WHEN PADA TRIGGER**

Untuk membatasi operasi trigger hanya pada baris yang memenuhi kondisi tertentu, maka digunakan klausa WHEN. Berikut ini akan dibuat trigger pada tabel EMPLOYEES yang menghitung komisi yang diterima oleh seorang pegawai pada saat sebuah baris ditambahkan ke dalam tabel EMPLOYEES, atau pada saat dilakukan modifikasi pada gaji pegawai.

**CREATE OR REPLACE TRIGGER derive\_commission\_pct**

**BEFORE INSERT OR UPDATE OF salary ON employees**

**FOR EACH ROW**

**WHEN (NEW.job\_id = 'SA\_REP')**

**BEGIN**

**IF INSERTING**

**THEN :NEW.commission\_pct := 0;**

**ELSIF :OLD.commission\_pct IS NULL**

**THEN :NEW.commission\_pct := 0;**

**ELSE**

**:NEW.commission\_pct := :OLD.commission\_pct + 0.05;**

**END IF;**

**END;**

Pada klausa WHEN, penggunaan OLD dan NEW qualifier tidak dengan prefiks (:). Untuk menggunakan NEW qualifier, gunakan BEFORE Row Trigger, jika timing BEFORE pada trigger diatas diganti dengan AFTER, maka akan didapat pesan kesalahan :

**CREATE OR REPLACE TRIGGER derive\_commission\_pct\***

**ERROR at line 1:**

**ORA-04084: cannot change NEW values for this trigger type**

## **PERINTAH UMUM**

Berikut ini perintah-perintah umum yang digunakan pada trigger. Untuk mengaktifkan atau menonaktifkan database trigger, digunakan perintah :

**ALTER TRIGGER trigger\_name DISABLE | ENABLE**

Untuk mengaktifkan atau menonaktifkan semua trigger yang berlaku untuk sebuah tabel, digunakan perintah :

**ALTER TABLE table\_name DISABLE | ENABLE ALL**

Untuk melakukan kompilasi ulang sebuah trigger, digunakan perintah :

**ALTER TRIGGER trigger\_name COMPILE**

Untuk menghapus trigger dari database, digunakan perintah :

**DROP TRIGGER trigger\_name**

Catatan : Semua trigger yang berlaku pada sebuah tabel akan dihapus pada saat tabel tersebut dihapus dari database.

Soal bab 13 :

---

1. Tampilkan nama pegawai dan tanggal yang pekerjaannya paling awal bekerja dibandingkan dengan pegawai lainnya.
2. Tampilkan nama pegawai dan tanggal terkecuali yang mendapat pekerjaan paling akhir.
3. Tampilkan nomer dan nama pegawai untuk semua pegawai yang bekerja di department yang sama dengan pegawai yang memiliki nama yang mengandung huruf 'T', dan gaji yang dimiliki lebih besar daripada rata-rata gaji .
4. Perubahan pada data hanya diperbolehkan selama jam kerja dari jam 8:45 pagi sampai 17.30 , dari Senin hingga Jum'at. Buat stored procedure dengan nama SECURE\_DML untuk mencegah DML statement dijalankan diluar dari jam kerja, dengan menampilkan pesan "Perubahan pada data hanya diperbolehkan hanya pada jam kerja"
5. Buat statement trigger pada tabel JOBS untuk memanggil prosedur diatas.
6. Implementasikan trigger berikut pada table JOBS sehubungan dengan kenaikan gaji pegawai. Buat stored procedure dengan nama UPD\_EMP\_SAL untuk mengupdate jumlah gaji. Prosedur ini menerima dua parameter : job id dari gaji yang akan diubah dan nilai minimum salary yang baru. Prosedur ini dijalankan dari trigger yang dibuat pada table JOBS.
7. Lanjutan dari soal nomor 6, buat row trigger dengan nama UPDATE\_EMP\_SALARY pada table JOBS yang memanggil prosedur UPD\_EMP\_SAL, pada saat minimum gaji pada table JOBS diubah untuk suatu job ID tertentu.

# Sesion XIV (Aplikasi View)

## Views

- View merupakan hasil dinamik dari satu atau lebih operasi relasional yang dioperasikan pada relasi dasar untuk menghasilkan relasi lain.
- View merupakan relasi virtual yang tidak perlu ada dalam database tetapi dihasilkan dari permintaan – permintaan khusus para user pada saat itu.
- Isi dari view didefinisikan sebagai query pada satu atau lebih relasi dasar.
- Dengan view resolution (pemecahan), operasi apapun pada view secara otomatis di terjemahkan kedalam operasi pada relasi mana view tersebut dihasilkan.
- Dengan view materialization (perwujudan), view disimpan sebagai tabel sementara, yang diatur sebagai tabel dasar utama yang telah diubah.

## SQL – Membuat view (CREATE VIEW)

Format penulisan CREATE VIEW :

```
CREATE VIEW ViewName [ (newColumnName [...]) ]  
AS subselect [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Dapat menetapkan nama untuk setiap kolomnya. Jika nama kolom didefinisikan, akan memiliki jumlah item yang sama dengan jumlah kolom yang dihasilkan oleh *subselect*. Jika nama kolom tidak didefinisikan, maka setiap kolom akan memiliki nama sesuai dengan kolom dalam *subselect*. Nama tabel harus ditentukan jika terdapat nama kolom yang sama (ambiguity). *Subselect* dikenal juga sebagai defining query. Penggunaan WITH CHECK OPTION memastikan jika tidak ada baris yang memenuhi kondisi clause WHERE pada *defining query*, maka tidak akan ditambahkan pada tabel dasar yang ditetapkan. Memerlukan hak SELECT pada semua tabel yang ditunjuk dalam subselect dan hak USAGE pada domain yang digunakan dalam kolom yang ditunjuk

## Contoh – Membuat view horisontal (Create Horizontal View)

Buatlah view, sehingga manager dikantor cabang B003 hanya dapat melihat detail staff yang bekerja di kantor cabang tersebut.

```
CREATE VIEW Manager3Staff  
AS SELECT *  
FROM Staff  
WHERE branchNo = 'B003';
```

**Table 6.3** Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

### Contoh – Membuat view vertikal (Create Vertical View)

Buatlah view detail staff dikantor cabang B003 tidak termasuk gaji.

```
CREATE VIEW Staff3  
AS SELECT staffNo, fName, lName, position, sex  
FROM Staff  
WHERE branchNo = 'B003';
```

**Table 6.4** Data for view Staff3.

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

### Contoh – View gabungan dan dikelompokkan (Grouped and Joined Views)

Buatlah view dari staff yang mengatur properti untuk disewakan, termasuk nomor kantor cabang tempat mereka bekerja, nomor staff dan jumlah properti yang ditangani.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)  
AS SELECT s.branchNo, s.staffNo, COUNT(*)  
FROM Staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo  
GROUP BY s.branchNo, s.staffNo;
```

**Table 6.5** Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

### SQL – Menghapus View (DROP VIEW)

Format penulisan DROP VIEW :

```
DROP VIEW ViewName [RESTRICT | CASCADE]
```

Akan menyebabkan terhapusnya pendefinisian view dari database.

**Contoh :**

**DROP VIEW Manager3Staff;**

Jika CASCADE didefinisikan, maka semua objek yang terkait/terhubung akan dihapus. Misalkan, view yang didefinisikan dari view yang dihapus. Jika RESTRICT (Default) didefinisikan, maka jika terdapat objek lain yang bergantung pada view yang akan dihapus, perintah penghapusan view akan ditolak.

### **View Resolution**

Carilah jumlah properti yang ditangani oleh setiap anggota staff dari kantor cabang B003.

```
SELECT staffNo, cnt  
FROM StaffPropCnt  
WHERE branchNo = 'B003'  
ORDER BY staffNo;
```

View resolution menggabungkan query diatas dengan *defining query* dari view StaffPropCnt sbb :

- (a) Nama kolom view dalam daftar SELECT diterjemahkan kedalam nama kolom yang dimaksud dalam *defining query*:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

- (b) Nama view dalam FROM digantikan dengan daftar FROM yang ditunjukkan dalam defining query:

```
FROM Staff s, PropertyForRent p
```

- (c) WHERE dari query user dikombinasikan dengan WHERE dari defining query menggunakan AND:

```
WHERE s.staffNo = p.staffNo AND branchNo = 'B003'
```

- (d) Clause GROUP BY dan HAVING disalin dari defining query:

```
GROUP BY s.branchNo, s.staffNo
```

- (e) ORDER BY disalin dari query dengan nama kolom view diterjemahkan kedalam nama kolom defining query :

```
ORDER BY s.staffNo
```

- (f) Hasil akhir penggabungan query, dieksekusi untuk menampilkan hasil :

```
SELECT s.staffNo, COUNT(*)  
FROM staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo AND branchNo = 'B003'  
GROUP BY s.branchNo, s.staffNo  
ORDER BY s.staffNo;
```

### Pembatasan pada View (Restrictions on Views)

SQL menentukan beberapa batasan pada pembuatan dan penggunaan view :

(a) Jika kolom dalam view berdasarkan pada fungsi aggregate, maka :

- Kolom hanya boleh muncul dalam clause SELECT dan ORDER BY dari query yang mengakses view.
- Kolom tidak dapat digunakan dalam WHERE maupun argumen untuk fungsi aggregate dalam query yang berasal dari view.

- Contoh, query berikut adalah salah :

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

Dan

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

(b) View yang dikelompokkan tidak akan pernah digabungkan dengan tabel dasar atau view.

- **Contoh** : view StaffPropCnt merupakan view yang dikelompokkan, oleh sebab itu usaha untuk menggabungkan view ini atau tabel lainnya akan gagal.

### View Updatability

Seluruh update yang dilakukan pada tabel dasar akan terlihat dalam semua view yang mengandung tabel dasar tersebut. Maka dapat dikatakan, jika view di-update/diubah maka tabel dasar akan menggambarkan /menampilkan perubahannya. Perhatikan view StaffPropCnt, jika akan dimasukan record pada kantor cabang B003, SG5 mengatur 2 properti, dapat dituliskan :

```
INSERT INTO StaffPropCnt
VALUES ('B003', 'SG5', 2);
```

Maka harus memasukan 2 record ke tabel PropertyForRent, yang menampilkan properti mana yang diatur oleh SG5, tetapi tidak diketahui properti mana, yaitu tidak diketahui primary key dari property yang dimaksud.

Jika merubah pendefinisian view dan mengganti *count* dengan jumlah properti yang sebenarnya, sbb :

```
CREATE VIEW StaffPropList (branchNo, staffNo, propertyNo)
AS SELECT s.branchNo, s.staffNo, p.propertyNo
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo;
```

Lalu masukan record :

```
INSERT INTO StaffPropList
VALUES ('B003', 'SG5', 'PG19');
```

Maka masih akan tetap bermasalah, karena dalam tabel PropertyForRent, seluruh kolom kecuali postcode dan staffNo tidak memperbolehkan NULL, dan tidak dapat memasukan nilai pada kolom-kolom NON NULL. ISO menetapkan, suatu view harus dapat di-update dalam sistem yang sesuai dengan standar.

Sebuah view dapat di-update jika dan hanya jika :

- DISTINCT tidak ditetapkan/digunakan.
- Setiap elemen dalam daftar SELECT dari defining query merupakan nama kolom dan tidak ada kolom yang muncul lebih dari satu kali
- Clause FORM ditetapkan hanya 1 tabel, tidak termasuk view yang berasal dari join, union, intersection atau difference.

- Tidak terdapat nested SELECT yang mengacu ke outer tabel.
- Tidak terdapat clause GROUP BY atau HAVING.
- Setiap baris yang ditambahkan melalui view harus tidak melanggar batasan integritas dari tabel dasar.

### Updatable View

Untuk view yang dapat diubah, DBMS harus berkemampuan untuk menelusuri kebelakang setiap baris atau kolom pada baris atau kolom dalam tabel asal/sumber.

### WITH CHECK OPTION

- Suatu baris muncul dalam view karena memenuhi kondisi dalam clause WHERE dari defining query.
- Jika baris berubah dan tidak lagi memenuhi kondisi, maka akan dihilangkan dari view.
- Baris baru akan muncul dalam view jika insert/update pada view memenuhi kondisi WHERE.
- Baris yang masuk atau keluar dari view disebut baris migrasi (*migrating rows*).
- WITH CHECK OPTION menghalangi baris migrasi keluar dari view. Terdapat qualifier optional LOCAL/CASCADED.
- Jika ditetapkan WITH LOCAL CHECK OPTION, maka setiap baris yang di-insert/update pada view ini dan setiap view yang didefinisikan secara langsung maupun tidak langsung pada view ini harus tidak menyebabkan baris hilang dari view kecuali baris dihilangkan dari derived view/table.
- JIKA ditetapkan WITH CASCADED CHECK OPTION (Default), maka setiap baris yang di-insert/update pada view ini dan pada setiap view yang didefinisikan secara langsung ataupun tidak langsung pada view ini harus tidak menyebabkan baris hilang dari view.

*Contoh - WITH CHECK OPTION*

Perhatikan statemen dibawah ini:

```
CREATE VIEW Manager3Staff
AS SELECT *
FROM Staff
WHERE branchNo = 'B003'
WITH CHECK OPTION;
```

Misalkan salah satu baris akan di-update nomor cabangnya dari B003 menjadi B005, sbb :

```
UPDATE Manager3Staff
SET branchNo = 'B005'
WHERE staffNo = 'SG37';
```

Penulisan WITH CHECK OPTION pada definisi view akan menggagalkan perintah diatas, karena dapat menyebabkan baris bermigrasi.

Juga tidak dapat memasukan baris kedalam view jika nomor cabang tidak sama dengan B003, contoh :

```
INSERT INTO Manager3Staff
VALUES ('SL15', 'Mary', 'Black' 'Assistant', 'F', DATE'1967-06-21', 8000, 'B002');
```

Penulisan WITH CHECK OPTION pada definisi view akan mencegah pemasukkan data ke tabel Staff .

Jika Manager3Staff didefinisikan tidak pada tabel Staff langsung tetapi pada view lain dari tabel Staff :

```
CREATE VIEW LowSalary
AS SELECT * FROM Staff WHERE salary > 9000;
```



```

CREATE VIEW HighSalary

AS SELECT * FROM LowSalary

WHERE salary > 10000

WITH LOCAL CHECK OPTION;

```

```

CREATE VIEW Manager3Staff
AS SELECT * FROM HighSalary
WHERE branchNo = 'B003';

```

Jika akan dilaksanakan update sbb :

```

UPDATE Manager3Staff
SET salary = 9500
WHERE staffNo = 'SG37';

```

Maka perintah diatas akan gagal, walaupun update akan menyebabkan baris hilang dari HighSalary, tetapi tidak akan hilang dari LowSalary. Jika update mencoba untuk menetapkan salary = 8000, update akan dilaksanakan dan baris tersebut tidak lagi menjadi bagian dari LowSalary. Jika HighSalary ditetapkan WITH CASCADED CHECK OPTION, penetapan salary menjadi 9500 atau 8000 akan ditolak karena menyebabkan baris dihilangkan dari HighSalary. Untuk mengatasi penyimpangan seperti ini, setiap view harus dibuat menggunakan WITH CASCADED CHECK OPTION.

#### *Keuntungan dari Views*

- Kemandirian data (Data independence)
- Ketepatan (Currency)
- Meningkatkan keamanan (Improved security)
- Mengurangi kerumitan (Reduced complexity)
- Kenyamanan (Convenience)
- Customization
- Integritas data (Data integrity)

#### *Disadvantages of Views*

- Pembatasan update (Update restriction)
- Pembatasan struktur (Structure restriction)
- Performance

#### **View Materialization**

- View Materialization, yaitu menyimpan sebuah view sebagai tabel sementara (*temporary*) dalam database pada saat view di-query-kan pertama kali.
- Query yang berdasarkan materialized view akan lebih cepat dihasilkan daripada membuat ulang view setiap saat.
- Materialized view bermanfaat dalam aplikasi baru seperti *data warehousing*, *replication server*, *data visualization*, dan *mobile system*.
- Pemeriksaan batasan integritas dan optimisasi query juga merupakan manfaat dari materialized view.

- Kesulitan dari pendekatan ini adalah pemeliharaan ketepatan view ketika tabel dasar diubah.
- Proses perubahan (updating) suatu materialized view sebagai respon terhadap perubahan data sumber disebut View maintenance.
- Tujuan utama dari view maintenance adalah menampilkan hal-hal yang berubah agar ketepatan view selalu terjaga.
- Perhatikan view berikut :

```
CREATE VIEW StaffPropRent(staffNo)  
AS SELECT DISTINCT staffNo  
FROM PropertyForRent  
WHERE branchNo = 'B003' AND rent > 400;
```

- Hasil dari perintah diatas :

staffNo
SG37
SG14

- Jika akan dimasukan record baru ke tabel PropertyForRent dimana rent <= 400 maka view tidak akan berubah.
- Jika akan dimasukan data ('PG24', ... , 550, 'CO40', 'SG19', 'B003') ke tabel PropertyForRent, maka baris baru akan muncul pada materialized view.
- Tetapi jika data yang akan dimasukan adalah ('PG54', ..., 450, 'CO89', 'SG37', 'B003') maka tidak akan terjadi penambahan baris karena 'SG37' sudah ada dalam materialized view.

## Contoh memakai login HR.

### Memakai 1 tabel

```
create view v_employees as  
select first_name, last_name, salary  
from employees;
```

View created.

0.35 seconds

```
select * from v_employees;
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Neena	Kochhar	17000
Lex	De Haan	17000
Alexander	Hunold	9000
Bruce	Ernst	6000
David	Austin	4800
Valli	Pataballa	4800
Diana	Lorentz	4200
Nancy	Greenberg	12000
Daniel	Faviet	9000

## Memakai beberapa tabel

Contoh :

```
create view v_employees_jobs as  
  
select e.first_name, e.last_name, j.job_title  
  
from employees e, jobs j  
  
where e.job_id = j.job_id;
```

View created.

0.35 seconds

Menampilkan seluruh data dari view v\_employees\_job.

```
select * from v_employees_jobs;
```

FIRST_NAME	LAST_NAME	JOB_TITLE
Steven	King	President
Neena	Kochhar	Administration Vice President
Lex	De Haan	Administration Vice President
Alexander	Hunold	Programmer
Bruce	Ernst	Programmer
David	Austin	Programmer
Valli	Pataballa	Programmer
Diana	Lorentz	Programmer
Nancy	Greenberg	Finance Manager
Daniel	Faviet	Accountant
More than 10 rows available. Increase rows selector to view more rows.		

1. Tampilkan semua data view v\_employees yang mempunyai gaji / salary di antara 5000 dan 10000
2. Tampilkan semua data view v\_employees yang gaji / salary -nya di atas rata-rata
3. Buatlah view dengan nama v\_locations yang menampilkan location\_id, postal\_code, dan city.
4. Tampilkan semua data view v\_locations yang terdapat di kota berawalan 's'.
5. Tampilkan semua data view v\_employees\_job yang mempunyai pekerjaan seorang programmer.
6. Buatlah view dengan nama v\_departments\_locations\_countries\_regions yang menampilkan location\_id, street\_address, postal\_code, city, country\_name dan region\_name.
7. Tampilkan semua data view v\_departments\_locations\_countries\_regions yang terdapat di Asia.
8. Buat view SALARY\_VU yang berisi nama pegawai, nama department, gaji dan grade dari gaji untuk semua pegawai. Beri judul PEGAWAI, DEPARTMENT, GAJI, GRADE. Tampilkan data pada SALARY\_VU.

ENAME	DNAME	SAL	GRADE
JAMES	SALES	950	1
SMITH	RESEARCH	800	1
ADAMS	RESEARCH	1100	1
MARTIN	SALES	1250	2
WARD	SALES	1250	2
MILLER	ACCOUNTING	1300	2
ALLEN	SALES	1600	3
TURNER	SALES	1500	3
BLAKE	SALES	2850	4
CLARK	ACCOUNTING	2450	4
JONES	RESEARCH	2975	4
ENAME	DNAME	SAL	GRADE
FORD	RESEARCH	3000	4
SCOTT	RESEARCH	3000	4
KING	ACCOUNTING	5000	5

14 rows selected.

## Dreamhome Database

### *Registration*

<u>ClientNo</u>	<u>BranchNo</u>	<u>StaffNo</u>	<u>Data_joined</u>
CR76	B005	SL41	2-Jan-01
CR56	B003	SG37	11-Apr-00
CR74	B003	SG37	16-Nov-99
CR62	B007	SA9	7-Mar-00

### *Viewing*

<u>ClientNo</u>	<u>PropertyNo</u>	<u>View_Date</u>	<u>Comment</u>
CR56	PA14	24-May-01	Too Small
CR76	PG4	20-Apr-01	Too Remote
CR56	PG4	26-May-01	
CR62	PA14	14-May-01	No Dining Room
CR56	PG36	28-Apr-01	

### *Client*

<u>ClientNo</u>	<u>fName</u>	<u>lName</u>	<u>TelNo</u>	<u>PrefType</u>	<u>MaxRent</u>
CR76	John	Kay	0207-774-5632	Flat	425
CR56	Aline	Stewart	0141-848-1825	Flat	350
CR74	Mike	Ritchie	01475-392178	House	750
CR62	Mary	Tregear	01224-196720	Flat	600

### *Private Owner*

<u>Owner No</u>	<u>fName</u>	<u>lName</u>	<u>Address</u>	<u>TelNo</u>
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

### *PropertyForRent*

<u>PropertyNo</u>	<u>Street</u>	<u>City</u>	<u>PostCode</u>	<u>Type</u>	<u>Rooms</u>	<u>Rent</u>	<u>Owner No</u>	<u>Staff No</u>	<u>Branch No</u>
PA14	16 Holhead	Aberdeen	AB7 5SO	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	-	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

**Branch**

<b>BranchNo</b>	<b>Street</b>	<b>City</b>	<b>PostCode</b>
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

**Staff**

<b>StaffNo</b>	<b>fName</b>	<b>lName</b>	<b>Position</b>	<b>Sex</b>	<b>DOB</b>	<b>Salary</b>	<b>BranchNo</b>
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005