# Basis Data

Tessy Badriyah, PhD.

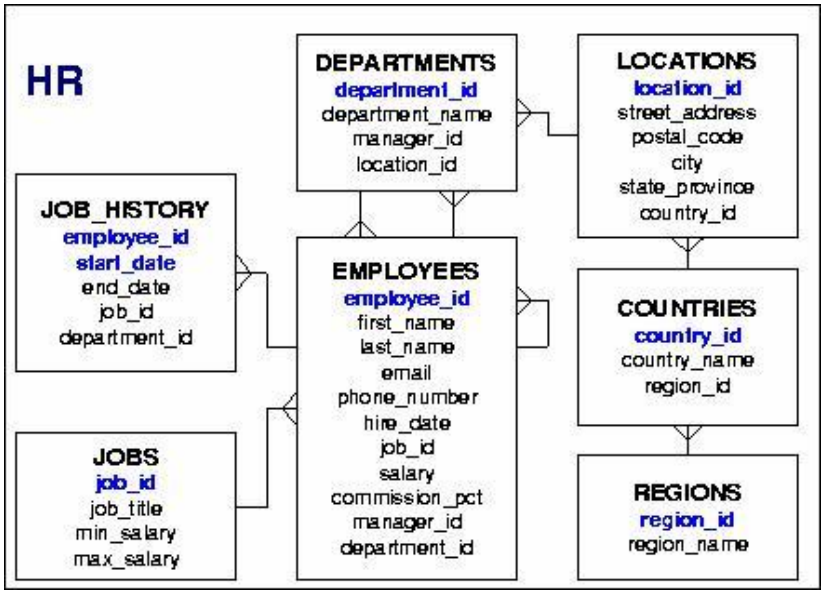
# Materi 1
# Dasar Perintah SQL

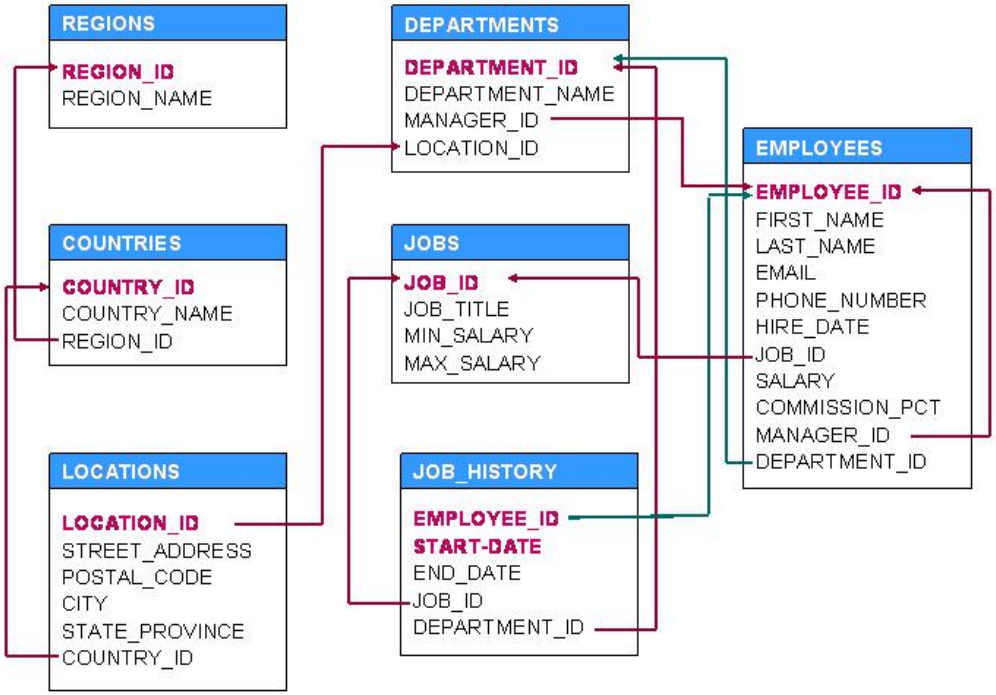
# Tujuan Pembelajaran

- Dasar penulisan SQL
- Pembatasan dan Pengurutan Data
- Fungsi Baris Tunggal

# Dasar penulisan SQL

# ER-D & Model Relasional skema HR



# Dasar Statement SELECT

- Sintak (cara penulisan) dari statement SELECT :

  SELECT [DISTINCT] {*, column [alias], … }

  FROM table;

  - SELECT digunakan untuk memilih kolom yang ingin ditampilkan.
  - FROM di
- Contoh:

```
SELECT *
FROM   departments;
```

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

# Memilih Kolom tertentu untuk ditampilkan

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

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

# Aturan Penulisan Statement SQL

- SQL Statement tidak case sensitive artinya tidak dibedakan antara penulisan huruf kecil dan huruf besar.
- SQL Statement dapat terdiri dari lebih dari satu baris.
- Keyword tidak bisa disingkat atau dipisah di baris yang berbeda.
- Klausa biasanya ditempatkan pada baris yang berbeda.
- Inden digunakan untuk memudahkan pembacaan.

# Ekspresi Aritmatik dalam SQL

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

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

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

| LAST_NAME | SALARY | 12*SALARY+100 |
|---|---|---|
| King | 24000 | 288100 |
| Kochhar | 17000 | 204100 |
| De Haan | 17000 | 204100 |
| Hunold | 9000 | 108100 |
| Ernst | 6000 | 72100 |
| ... | | |
| Hartstein | 13000 | 156100 |
| Fay | 6000 | 72100 |
| Higgins | 12000 | 144100 |
| Gietz | 8300 | 99700 |

# Penggunaan Tanda Kurung

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

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

| LAST_NAME | SALARY | 12*(SALARY+100) |
|---|---|---|
| King | 24000 | 289200 |
| Kochhar | 17000 | 205200 |
| De Haan | 17000 | 205200 |
| Hunold | 9000 | 109200 |
| Ernst | 6000 | 73200 |
| ... | | |
| Hartstein | 13000 | 157200 |
| Fay | 6000 | 73200 |
| Higgins | 12000 | 146200 |
| Gietz | 8300 | 100800 |

20 rows selected.

# Nilai NULL dalam SQL

- Nilai NULL (kosong) pada suatu kolom bisa berarti ada data yang tidak diisi, atau tidak diketahui nilainya.
- Nilai NULL tidak sama dengan NOL (zero).
- Nilai NULL juga tidak sama dengan spasi kosong.
- Supaya nilai NULL tidak ditampilkan, digunakan fungsi NVL untuk memeriksa keberadaan nilai NULL.
  - Formatnya :

    NVL(kolom yang di-evaluasi, diisi nilai ini jika kolom bernilai NULL)

# Nilai NULL dalam SQL

```
SELECT *
FROM EMPLOYEES
WHERE DEPARTMENT_ID IS NULL;
```

Execute | Load Script | Save Script | Cancel

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | COMMISSION_PCT | MANAGER_ID | DEPARTMENT_ID |
|---|---|---|---|---|---|---|---|---|---|---|
| 178 | Kimberely | Grant | KGRANT | 011.44.1644.429263 | 24-MAY-99 | SA_REP | 7000 | .15 | 149 | |

```
SELECT FIRST_NAME, LAST_NAME, NVL(DEPARTMENT_ID, 0)
FROM EMPLOYEES
WHERE DEPARTMENT_ID IS NULL;
```

Execute | Load Script | Save Script | Cancel

| FIRST_NAME | LAST_NAME | NVL(DEPARTMENT_ID,0) |
|---|---|---|
| Kimberely | Grant | 0 |

# Memberi nama lain untuk nama kolom
# (kolom alias)

- Judul (secara default) pada tiap kolom yang ditampilkan (heading) selalu sama dengan nama kolomnya dan ditulis dengan huruf besar.
- Penggantian judul kolom disebut kolom alias.

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

| NAME | COMM |
|---|---|
| King | |
| Kochhar | |
| De Haan | |

...

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM   employees;
```

| Name | Annual Salary |
|---|---|
| King | 288000 |
| Kochhar | 204000 |
| De Haan | 204000 |

...

20 rows selected.

# Operator Penyambungan (||)

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

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```

| Employees |
| --- |
| KingAD_PRES |
| KochharAD_VP |
| De HaanAD_VP |
| HunoldIT_PROG |
| ErnstIT_PROG |
| LorentzIT_PROG |
| MourgosST_MAN |
| RajsST_CLERK |

...

20 rows selected.

# Literal Character Strings

- Literal character strings adalah nilai yang ditampilkan bersama dengan nilai kolom.
- Literal dapat berupa karakter, ekspresi atau bilangan.
- Literal berupa tanggal (DATE) dan karakter harus diapit

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

| Employee Details |
|---|
| King is a AD_PRES |
| Kochhar is a AD_VP |
| De Haan is a AD_VP |
| Hunold is a IT_PROG |
| Ernst is a IT_PROG |
| Lorentz is a IT_PROG |
| Mourgos is a ST_MAN |
| Rajs is a ST_CLERK |

...

20 rows selected.

# Menghapus Baris yang Duplikat

- Untuk menghapus baris duplikat atau nilai yang sama muncul lebih dari satu kali, dapat digunakan keyword DISTINCT.

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

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

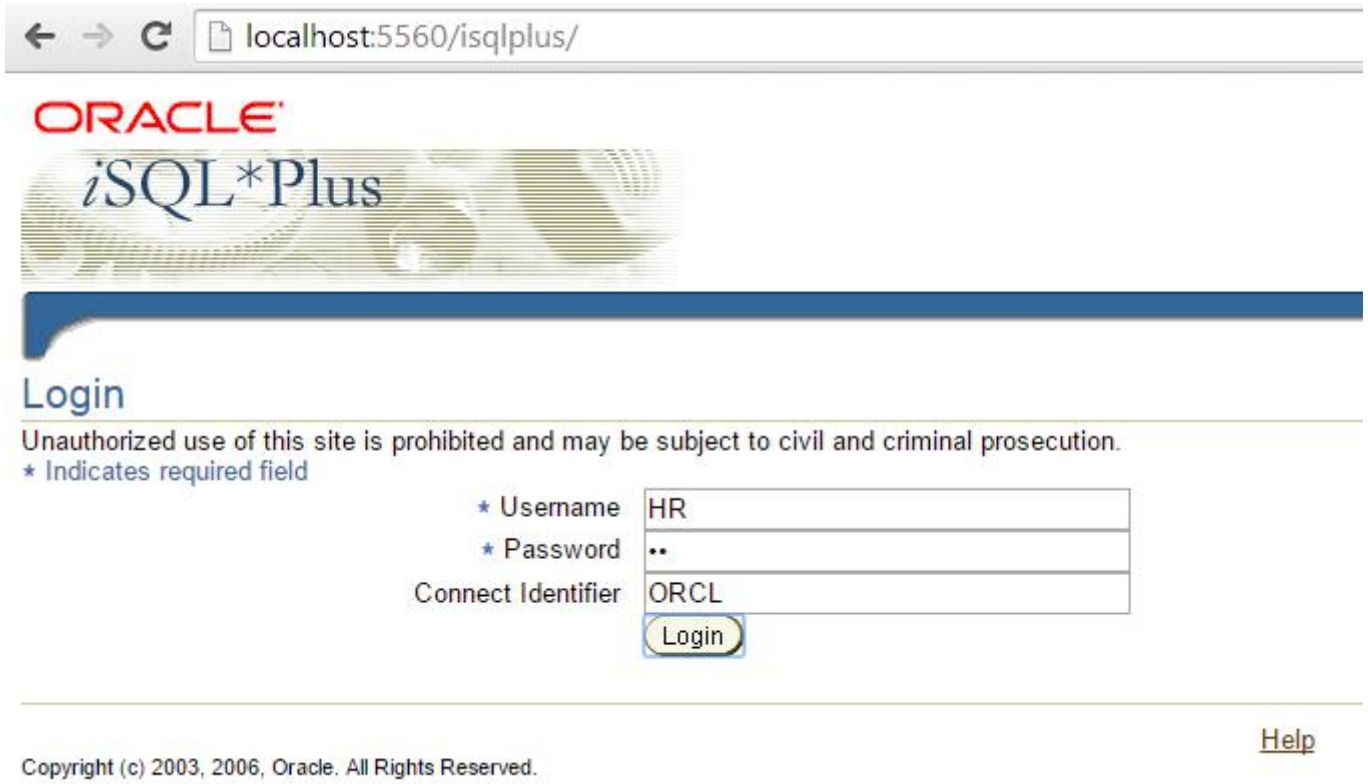
# iSQL*PLUS

- iSQL*PLUS  adalah salah satu tool yang dapat digunakan untuk menjalankan perintah SQL
- Untuk memanggil iSQL*PLUS, jalankan browser, pada jendela browser ketik alamat :

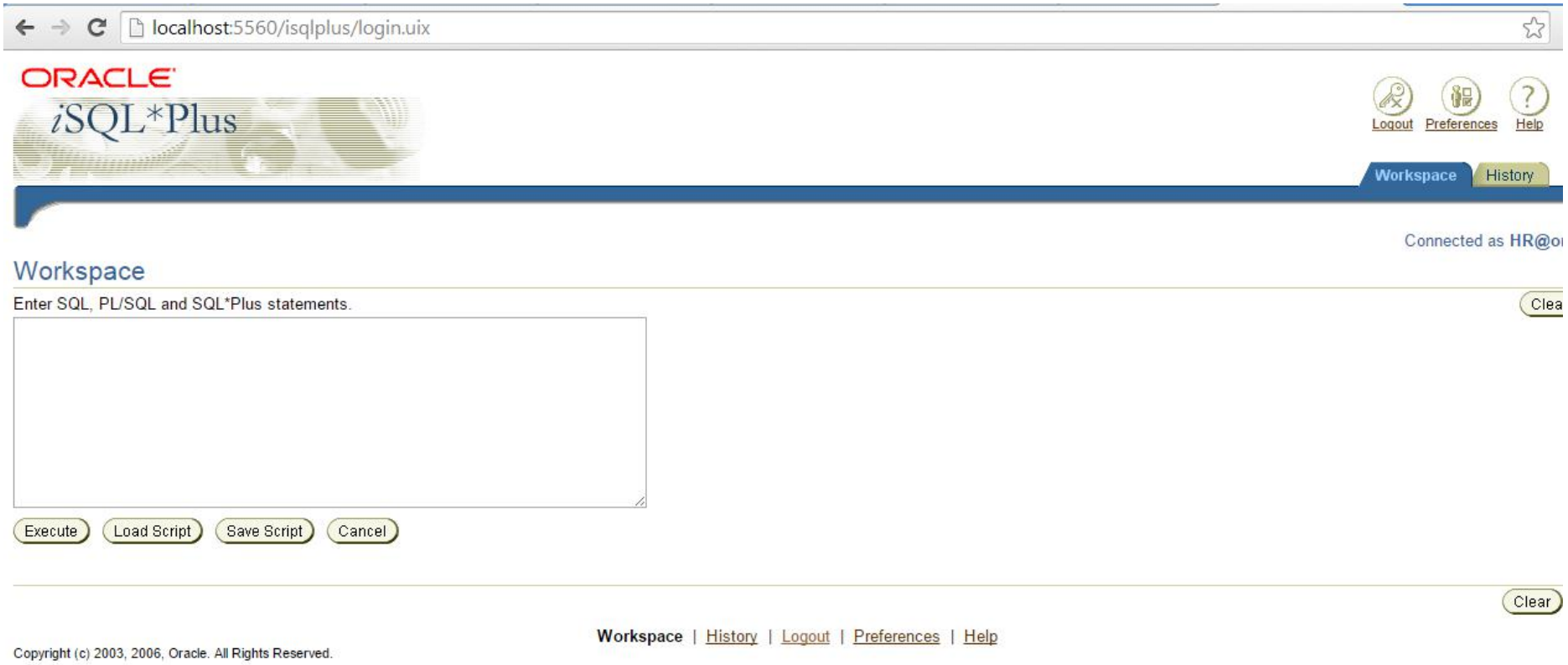
  http://nama_mesin/isqlplus

  - nama mesin adalah nomer IP tempat dimana database ditempatkan pada server atau pada local machine.

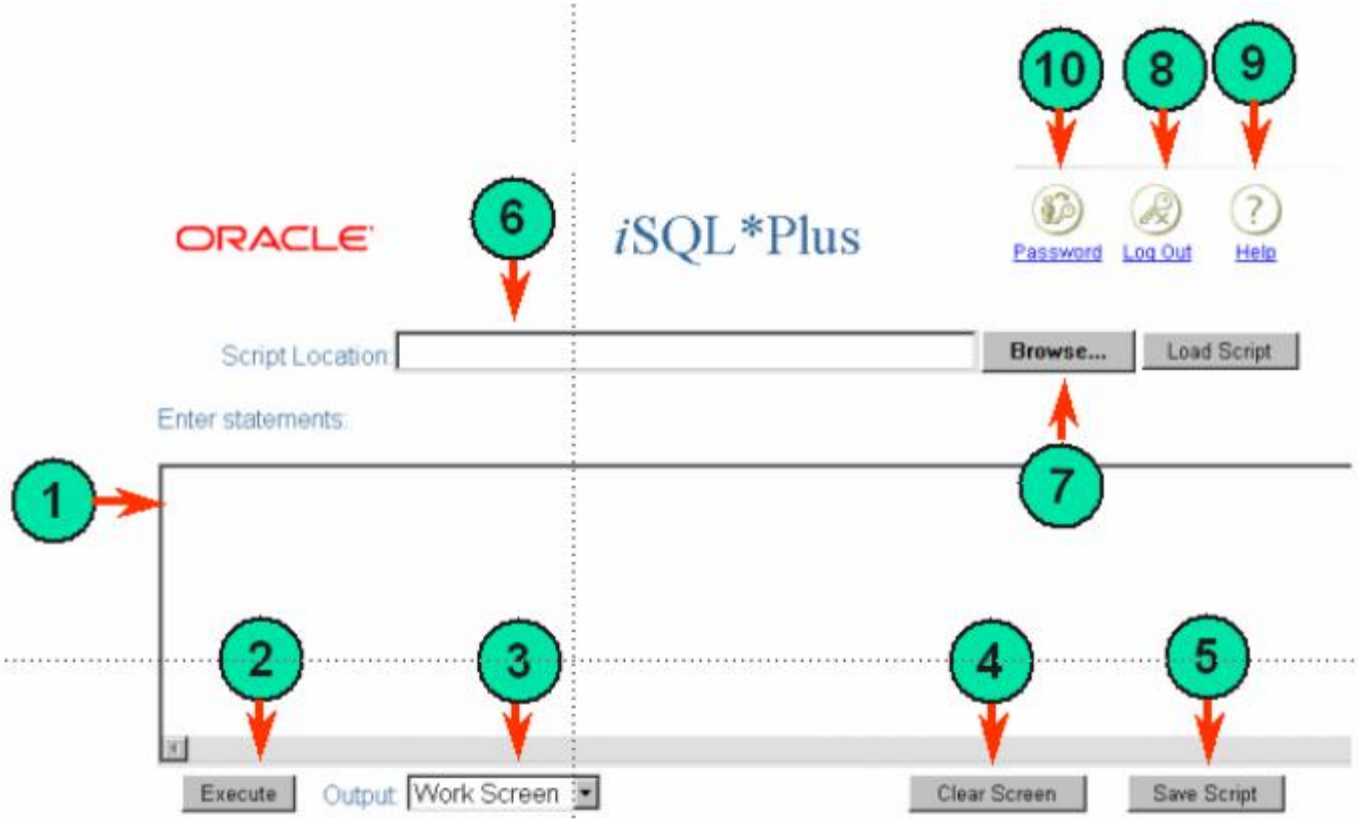
# iSQL*PLUS


localhost:5560/isqlplus/
ORACLE
iSQL*Plus
Login
Unauthorized use of this site is prohibited and may be subject to civil and criminal prosecution.
* Indicates required field
* Username
HR
* Password
Connect Identifier
ORCL
Login
Help
Copyright (c) 2003, 2006, Oracle. All Rights Reserved.

# iSQL*PLUS


localhost:5560/isqlplus/login.uix
ORACLE
iSQL*Plus
Logout
Preferences
Help
Workspace
History
Connected as HR@o
Workspace
Enter SQL, PL/SQL and SQL*Plus statements.
Clea
Execute
Load Script
Save Script
Cancel
Clear
Workspace | History | Logout | Preferences | Help
Copyright (c) 2003, 2006, Oracle. All Rights Reserved.

# iSQL*PLUS Environment



# Menampilkan Struktur Tabel

- Untuk menampilkan struktur table digunakan perintah DESCRIBE.
- Formatnya :

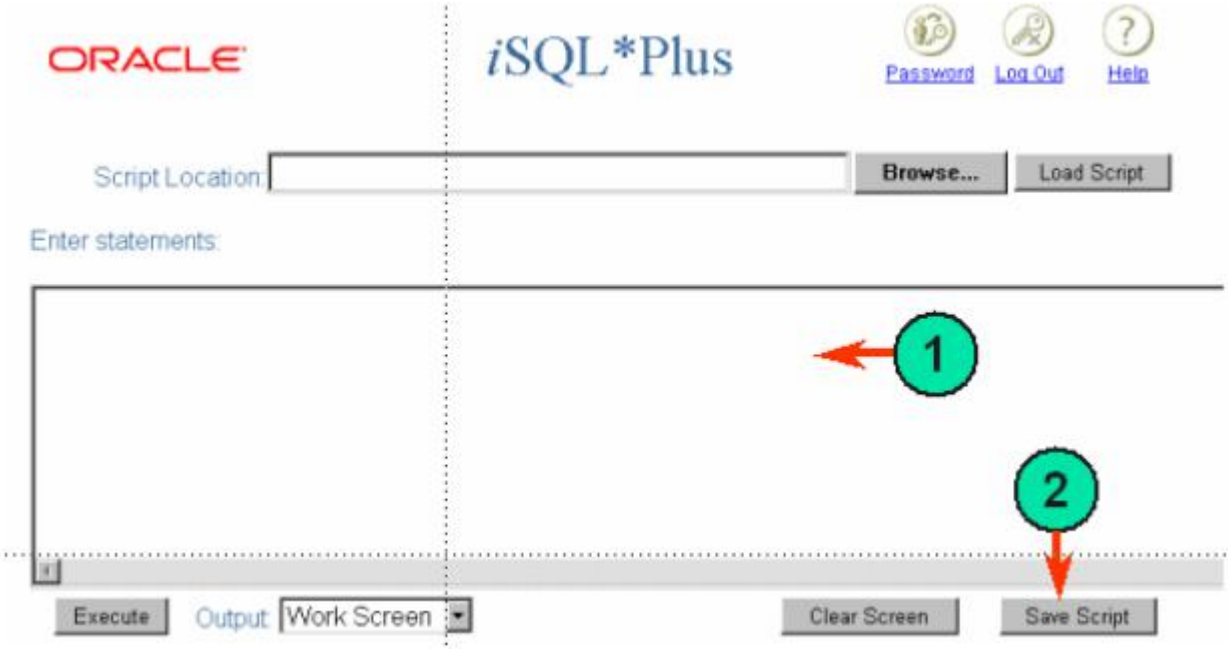
  DESC[RIBE] namatabel

```
DESCRIBE employees
```

| Name | Null? | Type |
|---|---|---|
| EMPLOYEE_ID | NOT NULL | NUMBER(6) |
| FIRST_NAME | | VARCHAR2(20) |
| LAST_NAME | NOT NULL | VARCHAR2(25) |
| EMAIL | NOT NULL | VARCHAR2(25) |
| PHONE_NUMBER | | VARCHAR2(20) |
| HIRE_DATE | NOT NULL | DATE |
| JOB_ID | NOT NULL | VARCHAR2(10) |
| SALARY | | NUMBER(8,2) |
| COMMISSION_PCT | | NUMBER(2,2) |
| MANAGER_ID | | NUMBER(6) |
| DEPARTMENT_ID | | NUMBER(4) |

# File Script

- Apa yang ditulis dalam editing window pada iSQL*PLUS dapat disimpan sebagai file script.
- File script dapat dipanggil, disimpan dan dijalankan.



# Pembatasan dan Pengurutan Data

- Membatasi baris yang didapatkan dari suatu query
- Mengurutkan baris yang didapat dari suatu query

# Membatasi Baris Mengggunakan Klausa WHERE

- Baris-baris data yang dihasilkan dari suatu query dapat dibatasi dengan memberikan klausa WHERE.
- Bentuk umumnya sebagai berikut :

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

# Contoh Pembatasan Data

- Mendapatkan pegawai yang bekerja di Department 90

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

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

# Pembatasan Data dengan Operator Pembandingan

- Operator perbandingan dapat digunakan pada klausaWHERE, dan mempunyai sintak sebagai berikut :

  WHERE expr operator value

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

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

# Operator Pembandingan

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

# Operator BETWEEN

- Operator BETWEEN digunakan untuk menampilkan baris berdasarkan suatu jangkauan (range) nilai.

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

Lower limit Upper limit

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

# Operator IN

- Operator IN digunakan untuk pembandingan dengan nilai-nilai yang ada dalam tanda kurung).

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

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

8 rows selected.

# Operator LIKE

- Operator LIKE digunakan untuk membentuk pencarian string yang sesuai dengan nilai yang
- dicantumkan pada kondisi. Kondisi pencarian dapat berisi karakter atau bilangan, dan 2 (dua) symbol khusus berikut :

  % menotasikan zero atau banyak karakter

  _ menotasikan satu karakter

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

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

| LAST_NAME |
| --- |
| Kochhar |
| Lorentz |
| Mourgos |

# Operator NULL

- Contoh: perintah SQL berikut digunakan untuk menampilkan nama belakang pegawai yang tidak memiliki manager.

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

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

# Penggunaan Operator Logika

- Operator logika AN

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

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

- Operator logika OR

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

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

8 rows selected.

- Operator logika NO

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

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

# Aturan Presedensi dan Penggunaan tanda kurung

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

| LAST_NAME | JOB_ID | SALARY |
|---|---|---|
| King | AD_PRES | 24000 |
| Abel | SA_REP | 11000 |
| Taylor | SA_REP | 8600 |
| Grant | SA_REP | 7000 |

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

| LAST_NAME | JOB_ID | SALARY |
|---|---|---|
| King | AD_PRES | 24000 |

# Mengurutkan Data

- Klausa ORDER BY digunakan untuk mengurutkan data.
- Terdapat 2 (dua) jenis : ASC (Ascending – urut naik), dan DESC (Descending – urut turun).
- Secara default dianggap urut naik.

# Penggunaan klausa ORDER BY

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

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

...

20 rows selected.

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

| LAST_NAME | JOB_ID | DEPARTMENT_ID | HIRE_DATE |
|---|---|---|---|
| Zlotkey | SA_MAN | 80 | 29-JAN-00 |
| Mourgos | ST_MAN | 50 | 16-NOV-99 |
| Grant | SA_REP | | 24-MAY-99 |
| Lorentz | IT_PROG | 60 | 07-FEB-99 |
| Vargas | ST_CLERK | 50 | 09-JUL-98 |
| Taylor | SA_REP | 80 | 24-MAR-98 |
| Matos | ST_CLERK | 50 | 15-MAR-98 |
| Fay | MK_REP | 20 | 17-AUG-97 |
| Davies | ST_CLERK | 50 | 29-JAN-97 |

...

20 rows selected.

# Pengurutan data dengan kolom alias

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal;
```

| EMPLOYEE_ID | LAST_NAME | ANNSAL |
|---|---|---|
| 144 | Vargas | 30000 |
| 143 | Matos | 31200 |
| 142 | Davies | 37200 |
| 141 | Rajs | 42000 |
| 107 | Lorentz | 50400 |
| 200 | Whalen | 52800 |
| 124 | Mourgos | 69600 |
| 104 | Ernst | 72000 |
| 202 | Fay | 72000 |
| 176 | Grant | 84000 |

...

20 rows selected.

# Pengurutan data dari beberapa kolom

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

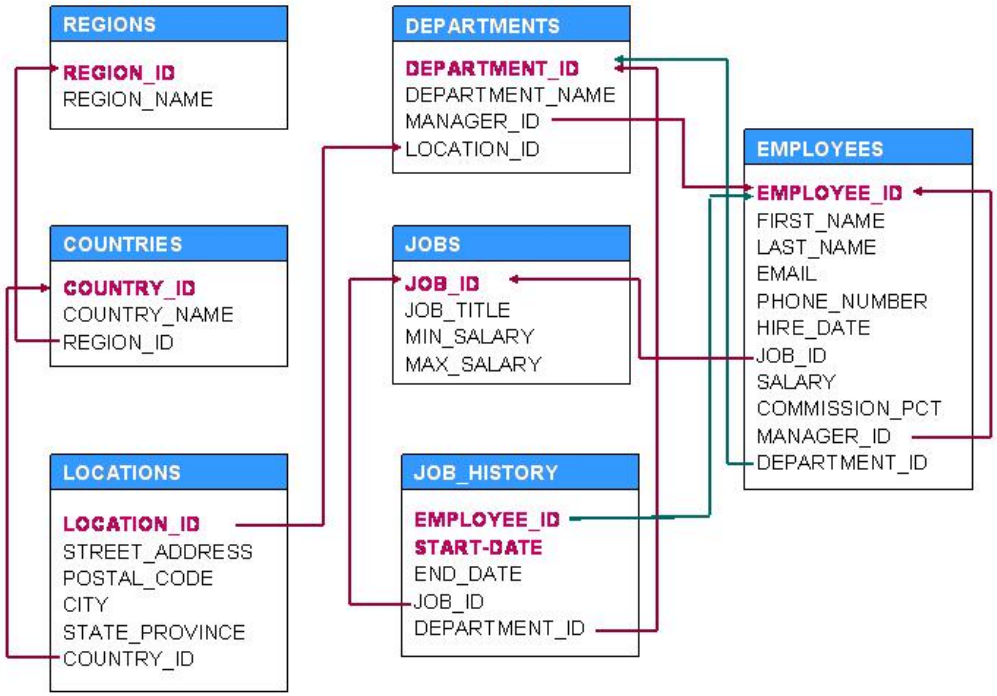
| LAST_NAME | DEPARTMENT_ID | SALARY |
|---|---|---|
| Whalen | 10 | 4400 |
| Hartstein | 20 | 13000 |
| Fay | 20 | 6000 |
| Mourgos | 50 | 5800 |
| Rajs | 50 | 3500 |
| Davies | 50 | 3100 |
| Matos | 50 | 2600 |
| Vargas | 50 | 2500 |

...

20 rows selected.

# Latihan Soal

- Gunakan model relasional berikut, untuk menjawab pertanyaan pada latihan soal!



# Latihan soal

1. Tampilkan struktur dari table DEPARTMENTS, kemudian tampilkan semua datanya !
2. Tampilkan struktur dari table EMPLOYESS. Buat query untuk menampilkan nomer pegawai, nama, pekerjaan, dan tanggal mulai bekerja untuk tiap pegawai.
3. Buat query untuk menampilkan jenis-jenis pekerjaan dari pegawai

# Latihan soal

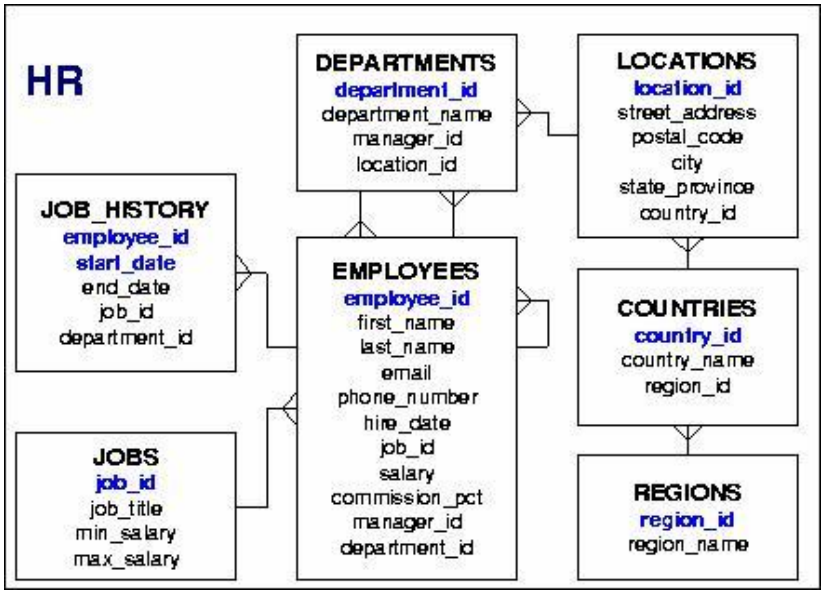
4. Tampilkan nama depan pegawai digabung dengan pekerjaan dengan dipisah tanda koma, kemudian beri judul “Pegawai dan Pekerjaan”
5. Buat query untuk menampilkan nama dan gaji dari pegawai yang memiliki gaji lebih dari $ 15000.

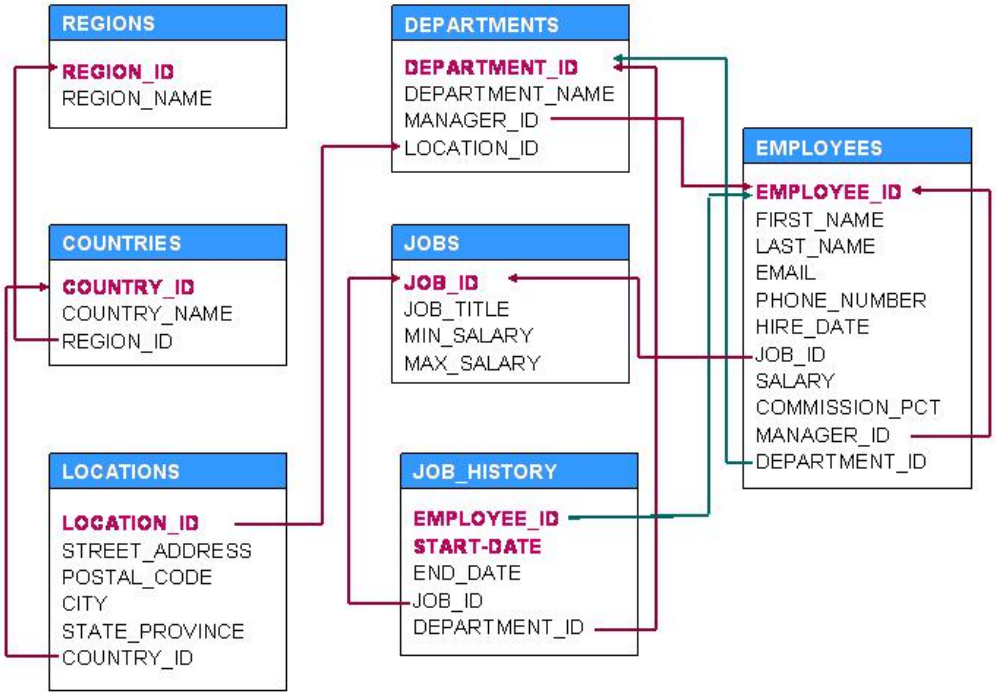
# Materi 2
# Operasi Join

# Tujuan Pembelajaran

- Dapat menampilkan data ke lebih dari satu tabel dengan menggunakan menggunakan operator join.
- Dapat menampilkan data yang tidak memenuhi kondisi join dengan menggunakan operator outer join
- Dapat melakukan join terhadap table itu sendiri (self join)

# ER-D & Model Relasional skema HR



# EquiJoin

- Definisi: Misal table EMPLOYEES memiliki primary key employee_id, dan memiliki foreign key department_id dimana departement_id ini merupakan primary key dari table yang lain yaitu table DEPARTMENTS.
- Relasi antara EMPLOYEES dengan DEPARTEMENTS disebut equi-join.
- Relasi antara dua tabel ditulis dalam klausa WHERE.

# Contoh Equi Join

```
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 |
|---|---|---|---|---|
| 200 | Whalen | 10 | 10 | 1700 |
| 201 | Hartstein | 20 | 20 | 1800 |
| 202 | Fay | 20 | 20 | 1800 |
| 124 | Mourgos | 50 | 50 | 1500 |
| 141 | Rajs | 50 | 50 | 1500 |
| 142 | Davies | 50 | 50 | 1500 |
| 143 | Matos | 50 | 50 | 1500 |
| 144 | Vargas | 50 | 50 | 1500 |

...

19 rows selected.

# Menambahkan Kondisi Pencarian dengan Operator AND

- Jika pada statement SQL sebelumnya (5.8) ditambahkan kondisi pencarian untuk nama pegawai = ‘Matos’ saja yang akan ditampilkan maka perintahnya SQL

```
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 AND
employees.last_name='Matos';
```

Execute | Load Script | Save Script | Cancel

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_ID | LOCATION_ID |
|---|---|---|---|---|
| 143 | Matos | 50 | 50 | 1500 |

# Penggunaan Tabel Alias

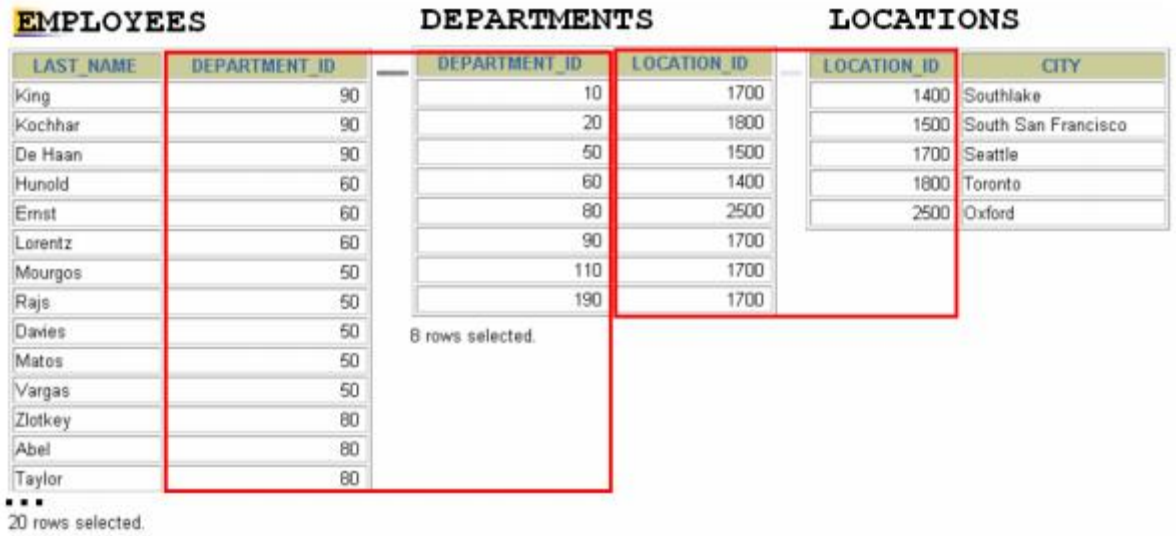
- Query dapat disederhanakan dengan penggunaan table alias.
- Contoh query berikut :

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

- Dengan menggunakan table alias akan diubah seperti berikut :

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

# Join lebih dari Dua Tabel



EMPLOYEES

| LAST_NAME | DEPARTMENT_ID |
|---|---|
| King | 90 |
| Kochhar | 90 |
| De Haan | 90 |
| Hunold | 60 |
| Ernst | 60 |
| Lorentz | 60 |
| Mourgos | 50 |
| Rajs | 50 |
| Davies | 50 |
| Matos | 50 |
| Vargas | 50 |
| Zlotkey | 80 |
| Abel | 80 |
| Taylor | 80 |

...

20 rows selected.

DEPARTMENTS

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

8 rows selected.

LOCATIONS

| LOCATION_ID | CITY |
|---|---|
| 1400 | Southlake |
| 1500 | South San Francisco |
| 1700 | Seattle |
| 1800 | Toronto |
| 2500 | Oxford |

- Relasi dari ketiga tabel dinyatakan dalam klausa WHERE sebagai berikut :

  WHERE employees.department_id=departments.department_id AND departments.location_id = locations.location_id;

# Non-EquiJoin

- Relasi antara dua table disebut non-equijoin jika kolom pada table pertama berkorespondensi langsung dengan kolom pada table kedua.

# Contoh Non-Equi Join

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

| LAST_NAME | SALARY | GRA |
|---|---|---|
| Matos | 2600 | A |
| Vargas | 2500 | A |
| Lorentz | 4200 | B |
| Mourgos | 5800 | B |
| Rajs | 3500 | B |
| Davies | 3100 | B |
| Whalen | 4400 | B |
| Hunold | 9000 | C |
| Ernst | 6000 | C |

...

20 rows selected.

# Outer Join

- Jika terdapat baris yang tidak memenuhi kondisi join, dan akan ditampilkan pada hasil query, maka digunakan outer join.
- Misal pada hasil query berikut, nama departemen ‘CONTRACTING’ tidak ditampilkan karena tidak memenuhi kondisi join, artinya pada tabel employee tidak ada pegawai yang bekerja pada departemen CONTRACTING.

# Ilustrasi Outer Join

**DEPARTMENTS**

| DEPARTMENT_NAME | DEPARTMENT_ID |
|---|---|
| Administration | 10 |
| Marketing | 20 |
| Shipping | 50 |
| IT | 60 |
| Sales | 80 |
| Executive | 90 |
| Accounting | 110 |
| Contracting | 190 |

8 rows selected.

**EMPLOYEES**

| DEPARTMENT_ID | LAST_NAME |
|---|---|
| 90 | King |
| 90 | Kochhar |
| 90 | De Haan |
| 60 | Hunold |
| 60 | Ernst |
| 60 | Lorentz |
| 50 | Mourgos |
| 50 | Rajs |
| 50 | Davies |
| 50 | Matos |
| 50 | Vargas |
| 80 | Zlotkey |

...

20 rows selected.

**Tidak ada pegawai yang bekerja di department 190.**

# Sintak Outer Join

```
SELECT  table1.column, table2.column
FROM    table1, table2
WHERE   table1.column(+) = table2.column;
```

```
SELECT  table1.column, table2.column
FROM    table1, table2
WHERE   table1.column = table2.column(+);
```

# Penggunaan Outer Join

- Jika akan ditampilkan kolom pada tabel departemen (DEPARTMENTS) yang tidak bersesuaian dengan semua kolom yang ada pada table pegawai (EMPLOYEES), (dalam kondisi : tidak ada pegawai yang berkerja di departemen ‘CONTRACTING’ dengan nomer 190, sehingga nomer 190 tidak muncul di tabel employees), digunakan query

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id(+) = d.department_id ;
```

| LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|
| Whalen | 10 | Administration |
| Hartstein | 20 | Marketing |
| Fay | 20 | Marketing |
| Mourgos | 50 | Shipping |
| Rajs | 50 | Shipping |
| Davies | 50 | Shipping |
| Matos | 50 | Shipping |
| ... | | |
| Gietz | 110 | Accounting |
| | | Contracting |

# Contoh Outer Join (1)

- Mencari pegawai yang tidak bekerja di department tertentu

```
SELECT *
FROM EMPLOYEES E, DEPARTMENTS D
WHERE
    E.DEPARTMENT_ID=D.DEPARTMENT_ID(+)
        AND  E.DEPARTMENT_ID IS NULL;
```

# Contoh Outer Join (2)

- Mencari departemen yang tidak memiliki pegawai yang bekerja di dalamnya

```
SELECT *
FROM EMPLOYEES E, DEPARTMENTS D
WHERE
    E.DEPARTMENT_ID(+)=D.DEPARTMENT_ID
        AND E.DEPARTMENT_ID IS NULL;
```

# Self Join

**EMPLOYEES (WORKER)**

| EMPLOYEE_ID | LAST_NAME | MANAGER_ID |
|---|---|---|
| 100 | King | |
| 101 | Kochhar | 100 |
| 102 | De Haan | 100 |
| 103 | Hunold | 102 |
| 104 | Ernst | 103 |
| 107 | Lorentz | 103 |
| 124 | Mourgos | 100 |

...

**EMPLOYEES (MANAGER)**

| EMPLOYEE_ID | LAST_NAME |
|---|---|
| 100 | King |
| 101 | Kochhar |
| 102 | De Haan |
| 103 | Hunold |
| 104 | Ernst |
| 107 | Lorentz |
| 124 | Mourgos |

...

**MANAGER_ID pada tabel WORKER sama dengan EMPLOYEE_ID pada tabel MANAGER.**

# Contoh Self-Join

```
SELECT worker.last_name || ' works for '
       || manager.last_name
FROM   employees worker, employees manager
WHERE  worker.manager_id = manager.employee_id ;
```

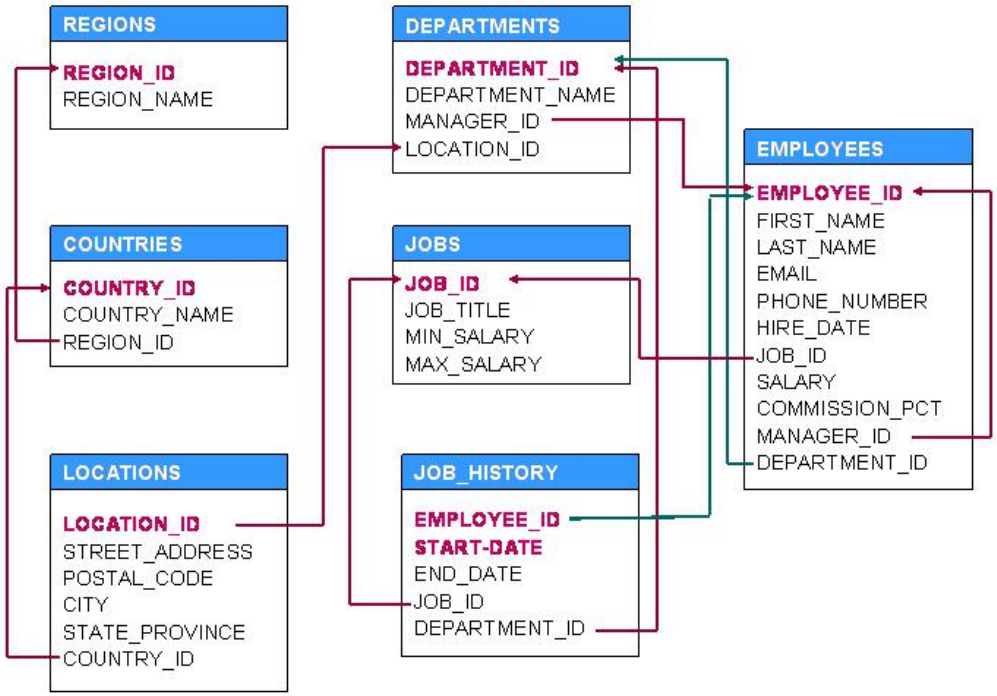
| WORKER.LAST_NAME\|\|'WORKSFOR'\|MANAGER.LAST_NAME |
|---|
| Kochhar works for King |
| De Haan works for King |
| Mourgos works for King |
| Zlotkey works for King |
| Hartstein works for King |
| Whalen works for Kochhar |
| Higgins works for Kochhar |
| Hunold works for De Haan |
| Ernst works for Hunold |

...

19 rows selected.

# Latihan Soal

- Gunakan model relasional berikut, untuk menjawab pertanyaan pada latihan soal!



# Latihan Soal

1. Buat query untuk menampilkan nama pegawai, nomer department dan nama department dari semua pegawai
2. Buat daftar yang unik dari semua pekerjaan pada department 30, tampilkan pula  nama kota (city) dari department 30.
3. Tampilkan nama pegawai, nama department dari semua pegawai yang memiliki komisi (komisi tidak sama dengan NULL)
4. Tampilkan nama pegawai dan nama department untuk semua pegawai yang memiliki huruf ‘A’ pada namanya.

# Latihan Soal

5. Buat query untuk menampilkan nama pegawai, pekerjaan, nomer department, dan nama department untuk semua pegawai yang bekerja di kota ‘DALLAS’
6. Buat query untuk menampilkan nama pegawai dan nomer pegawai, nama manager dan nomer pegawai dari manager.
7. Modifikasi query pada nomer 6, buat outer join untuk menampilkan pula data pegawai yang tidak mempunyai manager.
8. Buat query yang menampilkan nama pegawai, nomer department, dan semua employee yang bekerja pada department yang sama dengan employee.

# Latihan Soal

9. Tampilkan struktur dari table SALGRADE. Buat query yang menampilkan nama pegawai , pekerjaan, nama department, gaji dan grade untuk semua pegawai
10. Buat query untuk menampilkan nama dan tanggal mulai bekerja dari pegawai yang tanggal bekerjanya setelah pegawai bernama ‘BLAKE’
11. Tampilkan semua nama pegawai dan tanggal kerjanya serta nama manager dan tanggal kerjanya dimana tanggal mulai kerja pegawai lebih dulu daripada tanggal mulai kerja managernya.

# Latihan Soal

12. Tampilkan nama pegawai dan pekerjaannya untuk pegawai yang tidak bekerja di department manapun

```
SELECT FIRST_NAME, LAST_NAME, E.DEPARTMENT_ID
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID=D.DEPARTMENT_ID(+) AND
          E.DEPARTMENT_ID IS NULL;
```

Execute | Load Script | Save Script | Cancel

| FIRST_NAME | LAST_NAME | DEPARTMENT_ID |
|---|---|---|
| Kimberely | Grant | |

# Latihan Soal

13. Tampilkan nomer department dan nama departemen dimana tidak ada pegawai yang bekerja di departemen tersebut.

```
SELECT D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID(+)=D.DEPARTMENT_ID AND
          E.DEPARTMENT_ID IS NULL;
```

Execute | Load Script | Save Script | Cancel

| DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|
| 120 | Treasury |
| 130 | Corporate Tax |
| 140 | Control And Credit |
| 150 | Shareholder Services |
| 160 | Benefits |
| 170 | Manufacturing |
| 180 | Construction |
| 190 | Contracting |
| 200 | Operations |
| 210 | IT Support |
| 220 | NOC |
| 230 | IT Helpdesk |
| 240 | Government Sales |
| 250 | Retail Sales |
| 260 | Recruiting |
| 270 | Payroll |

16 rows selected.