5

Menampilkan Data Dari Beberapa Tabel

Tujuan

Setelah menyelesaikan pelajaran ini, Anda akan bisa melakukan sebagai berikut :

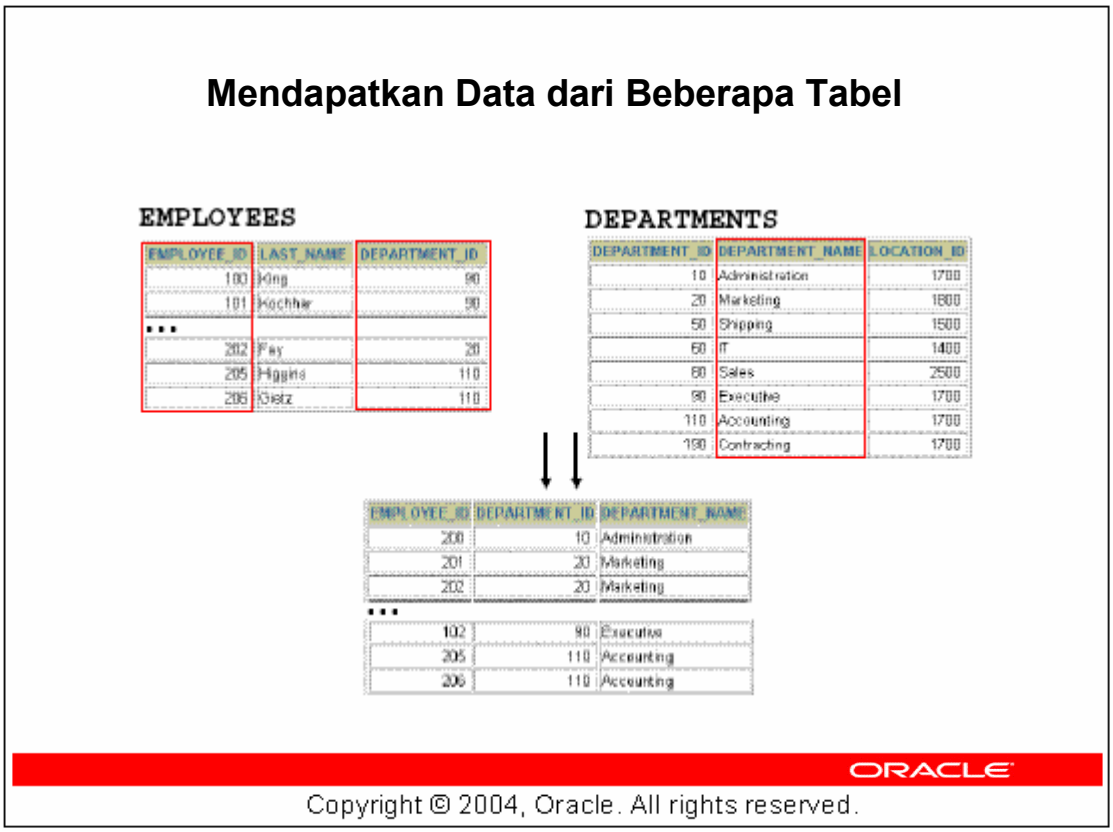
- **Menulis pernyataan-pernyataan `SELECT` untuk mengakses data dari beberapa tabel menggunakan *equijoin* dan *nonequijoin***
- **Menggabungkan suatu tabel itu sendiri dengan menggunakan *self-join***
- **Menampilkan data yang secara umum tidak sesuai kondisi penggabungannya dengan menggunakan *outer joins***
- **Menghasilkan suatu *Cartesian product* dari seluruh baris dari dua tabel atau lebih**

ORACLE

Tujuan

Pelajaran ini menjelaskan bagaimana cara untuk mendapatkan data dari beberapa tabel. Suatu *joins* (penggabungan) digunakan untuk menampilkan informasi dari beberapa tabel. Karena itu, Anda dapat *menggabungkan* tabel-tabel secara bersamaan untuk menampilkan informasi dari beberapa tabel.

Catatan : informasi tentang penggabungan (*joins*) ditemukan di ”SQL Queries and Subqueries: Joins” pada *Oracle SQL Reference*.



Mendapatkan Data dari Beberapa Tabel

Kadang-kadang Anda perlu untuk menggunakan data dari beberapa tabel. Contoh pada slide, suatu laporan menampilkan data dari dua tabel yang terpisah:

- Nomor-nomor pegawai ada pada tabel `EMPLOYEES`.
- Nomor-nomor departemen ada pada tabel `EMPLOYEES` dan tabel `DEPARTEMENTS`.
- Nama-nama departemen ada pada tabel `DEPARTEMENTS`.

Untuk menghasilkan laporan, Anda perlu untuk menghubungkan tabel `EMPLOYEES` dan tabel `DEPARTEMENTS` dan mengakses data dari kedua tabel tersebut.

Tipe-Tipe pada *Join*

***Join* yang mengacu pada standar SQL:1999 termasuk sebagai berikut :**

- ***Cross joins***
- ***Natural joins***
- **Klausa `USING`**
- ***Full* (atau *two-sided*) *outer joins***
- **Kondisi-kondisi penggabungan tidak tetap untuk *outer join***

ORACLE

Tipe-tipe join

Untuk mengabungkan (*join*) tabel-tabel, Anda dapat menggunakan sintak *join* yang mengacu pada standar SQL:1999.

Catatan: Sebelumnya pada Oracle9*i*, sintak join berbeda dengan Standard ANSI. Acuan join sintak SQL:1999 tidak menawarkan beberapa keuntungan performa daripada sintak *join* khusus milik Oracle yang ada pada keluaran awal. Untuk informasi lebih rinci tentang sintak *join* khusus milik Oracle, lihat Appendix C.

Menggabungkan Tabel-tabel Menggunakan Sintak SQL:1999

Gunakan suatu *join* untuk meng-query data dari beberapa tabel :

```
SELECT   table1.column, table2.column
FROM     table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)]|
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)]|
[CROSS JOIN table2];
```


Mendefinisikan Join

Dalam sintak:

`table1.column` menunjukkan tabel dan kolom dari mana data diperoleh

`NATURAL JOIN` menggabungkan dua tabel berdasarkan nama kolom yang sama

`JOIN table USING column_name` melakukan suatu *equijoin* berdasarkan nama kolom.

`JOIN table ON table1.column_name` melakukan suatu *equijoin* berdasarkan suatu kondisi pada klausa `ON`, `= table2.column_name`

`LEFT/RIGHT/FULL OUTER` digunakan untuk melakukan *outer join*s.

`CROSS JOIN` mengembalikan suatu *Cartesian product* dari dua tabel.

Untuk informasi lebih lanjut, lihat "`SELECT`" pada *Oracle SQL reference*.

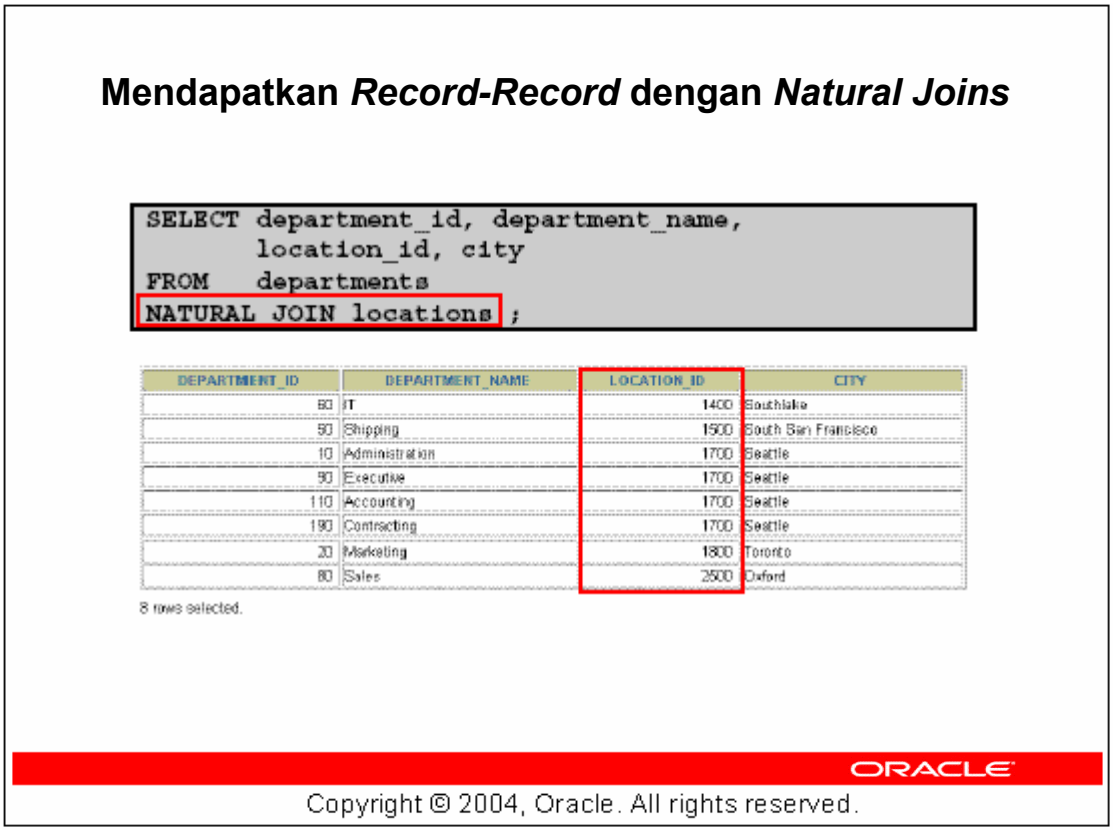
Membuat *Natural Joins*

- **Klausa `NATURAL JOIN` adalah didasarkan pada semua kolom pada dua tabel yang memiliki nama yang sama.**
- **`NATURAL JOIN` memilih baris-baris dari dua tabel yang memiliki nilai-nilai yang sama dalam semua kolom yang sesuai.**
- **Jika kolom-kolom memiliki nama yang sama memiliki tipe data berbeda, menghasilkan suatu *error.***

Membuat *Natural Joins*

Anda dapat menggabungkan tabel-tabel secara otomatis berdasarkan kolom-kolom pada dua tabel yang memiliki tipe data dan nama yang sama. Anda melakukan hal ini dengan menggunakan kata kunci `NATURAL JOIN`.

Catatan: Join dapat terjadi hanya pada kolom-kolom yang memiliki nama dan tipe data yang sama pada kedua tabel. Jika kolom-kolom memiliki nama yang sama tetapi memiliki tipe data yang berbeda, maka sintak `NATURAL JOIN` menyebabkan suatu error.



Mendapatkan *Record-Record* dengan *Natural Joins*

Contoh pada slide, tabel `LOCATIONS` digabungkan ke tabel `DEPARTMENT` oleh kolom `LOCATION_ID`, dimana satu-satunya kolom dengan nama yang sama pada kedua tabel. Jika terdapat kolom lain yang sama, join akan menggunakan semua kolom-kolom tersebut.

***Natural Join* dengan suatu klausa `WHERE`**

Pembatasan tambahan pada suatu *natural join* diterapkan dengan menggunakan klausa `WHERE`. Contoh berikut membatasi baris-baris sebagai outputnya pada suatu department ID sama dengan 20 atau 50 :

```
SELECT          department_id, department_name,
                location_id, city
FROM            departments
NATURAL JOIN    locations
WHERE           department_id IN(20,50);
```

Membuat *Join-Join* dengan Klausa USING

- **Jika beberapa kolom memiliki nama-nama yang sama tapi tipe datanya tidak sesuai, klausa NATURAL JOIN dapat dimodifikasi dengan klausa USING untuk menentukan kolom-kolom yang akan digunakan sebagai suatu *equijoin*.**
- **Gunakan klausa USING untuk penyesuaian hanya satu kolom saat beberapa kolom sama.**
- **Jangan gunakan nama tabel atau alias pada kolom-kolom referensi.**
- **Klausa-klausa NATURAL JOIN dan USING adalah *mutually exclusive.***

ORACLE

Klausa USING

Natural joins menggunakan semua kolom dengan menyesuaikan nama-nama dan tipe data-tipe data untuk menggabungkan tabel-tabel. Klausa USING dapat digunakan untuk menentukan hanya kolom-kolom tertentu yang akan digunakan untuk suatu *equijoin*. Kolom-kolom yang direferensikan pada klausa USING tidak akan memiliki suatu perubah/*qualifier* (nama tabel atau alias) di manapun pada pernyataan SQL.

Sebagai contoh, pernyataan berikut adalah tepat :

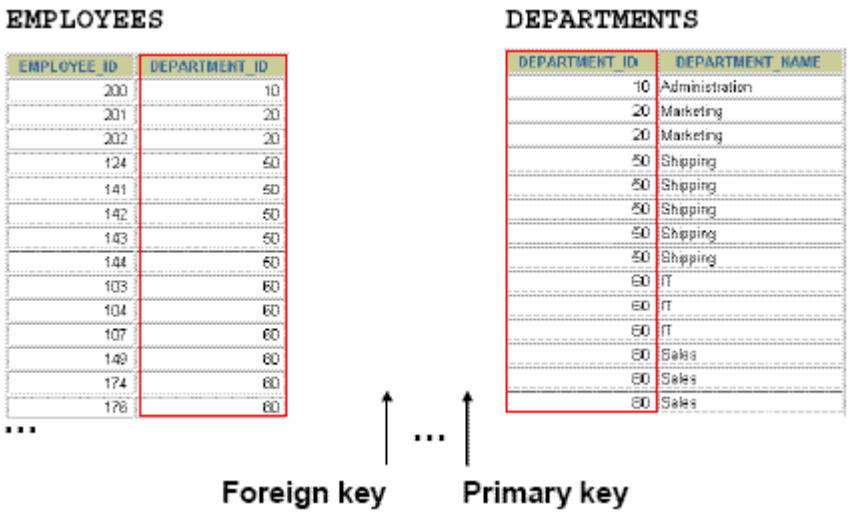
```
SELECT    l.city, d.department_name
FROM      locations l JOIN departments d USING (location_id)
WHERE     location_id = 1400;
```

Pernyataan berikut tidak tepat karena LOCATION_ID dirubah (*qualified*) pada klausa WHERE :

```
SELECT    l.city, d.department_name
FROM      locations l JOIN department d USING (location_id)
WHERE     d.location_id = 1400;
ORA-25154: column part of USING clause cannot have qualifier
```

Pembatasan yang sama juga berlaku untuk NATURAL *join*. Oleh karena itu, kolom-kolom yang memiliki nama yang sama pada kedua tabel harus digunakan tanpa ada qualifier/perubah.

Menggabungkan Nama-Nama Kolom



Klausa `USING` untuk *Equijoins*

Untuk menentukan nama departement dari para pekerja, Anda membandingkan nilai pada kolom `DEPARTMENT_ID` di dalam tabel `EMPLOYEES` dengan nilai-nilai `DEPARTMENT_ID` di dalam tabel `DEPARTMENTS`. Hubungan antara tabel-tabel `EMPLOYEES` dan `DEPARTMENTS` adalah suatu *equijoin*; karena itu, nilai-nilai pada kolom `DEPARTMENT_ID` di kedua tabel harus sama. Seringkali, tipe dari *join* ini melibatkan kelengkapan-kelengkapan *primary* dan *foreign key*.

Catatan: *Equijoin*s juga disebut *simple joins* atau *inner joins*.

Mendapatkan *Record-Record* dengan Klausa `USING`

```
SELECT employees.employee_id, employees.last_name,
       departments.location_id, department_id
FROM   employees JOIN departments
USING (department_id) ;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
124	Mourgos	1500	50
141	Rajs	1500	50
142	Davies	1500	50
144	Vargas	1500	50
143	Matos	1500	50

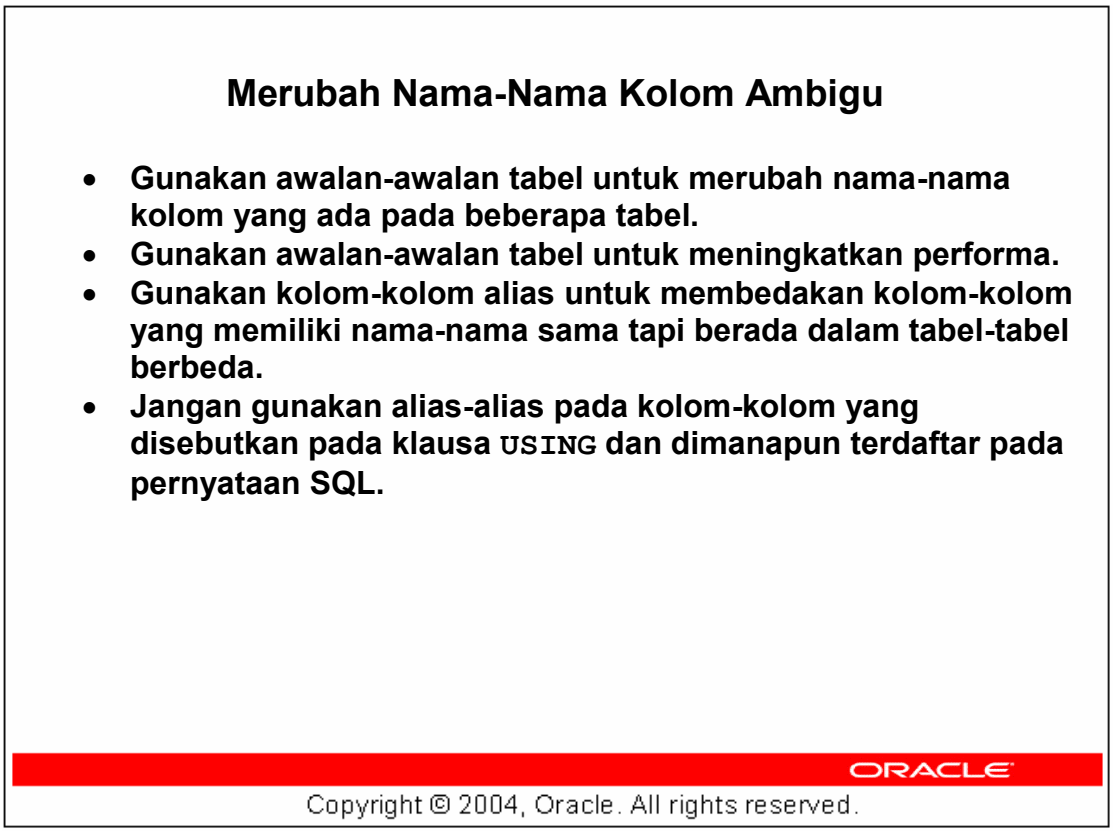
...

19 rows selected.

ORACLE

Mendapatkan *Record-Record* dengan Klausa `USING`

Contoh pada slide join kolom `DEPARTMENT_ID` di dalam tabel-tabel `EMPLOYEES` dan tabel `DEPARTMENTS`, dan juga ditampilkan lokasi dimana seorang pekerja bekerja.



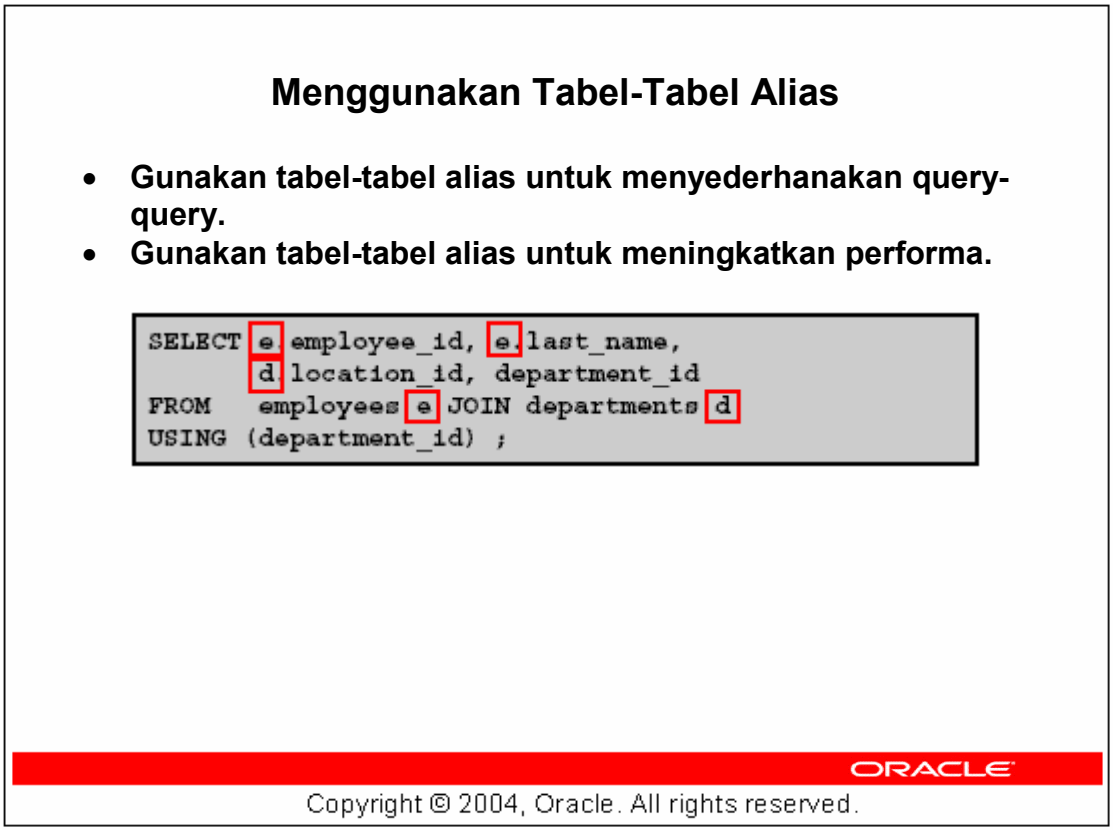
Merubah Nama-Nama Kolom Ambigu

Anda perlu untuk merubah nama-nama kolom dengan nama tabel untuk menghindari ambiguitas (kerancuan). Tanpa awalan-awalan tabel, kolom `DEPARTMENT_ID` pada daftar `SELECT` bisa jadi baik dari tabel `DEPARTMENTS` atau tabel `EMPLOYEES`. Hal tersebut penting untuk menambah awalan tabel untuk mengeksekusi *query* Anda :

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

Jika ada nama kolom yang tidak sama diantara dua tabel, tidak perlu untuk merubah kolom-kolom. Bagaimanapun, menggunakan awalan tabel akan meningkatkan performa, karena Anda memberitahukan server Oracle dimana harus mencari kolom-kolom dengan tepat.

Catatan: Saat menggabungkan dengan klausa `USING`, Anda tidak bisa merubah suatu kolom yang digunakan pada klausa `USING` itu sendiri. Lagipula, jika kolom tersebut digunakan dimana-mana pada pernyataan SQL, Anda tidak dapat meng-*alias*kannya.



Menggunakan Tabel-Tabel Alias

Merubah nama-nama kolom dengan nama-nama tabel bisa sangat menghabiskan banyak waktu, khususnya jika nama-nama tabel yang digunakan panjang. Anda dapat menggunakan tabel-tabel alias daripada nama-nama tabel. Seperti suatu kolom alias memberikan nama lain suatu kolom, suatu tabel alias memberikan nama lain suatu tabel. Tabel-tabel alias membantu menjaga kode SQL menjadi lebih sederhana, sehingga menghemat penggunaan memory.

Perhatikan bagaimana tabel-tabel alias diidentifikasi pada klausa `FROM` dalam contoh. Suatu nama tabel disebutkan secara utuh, diikuti oleh spasi dan kemudian suatu tabel alias. Tabel `EMPLOYEES` diberikan suatu alias `e`, dan tabel `DEPARTMENTS` memiliki suatu alias `d`.

Pedoman-pedoman

- Tabel-tabel alias dapat mencapai 30 karakter panjangnya, tapi alias-alias pendek lebih baik daripada alias-alias panjang.
- Jika suatu tabel alias digunakan untuk nama tabel khusus pada suatu klausa `FROM`, maka tabel alias itu harus diganti dengan nama tabel sepanjang pernyataan `SELECT`.
- Tabel-tabel alias sebaiknya mempunyai arti.
- Tabel alias berlaku hanya pada pernyataan `SELECT` saat ini.

Membuat *Join-Join* dengan Klausa ON

- **Kondisi join untuk *natural join* pada dasarnya adalah suatu e*quijoin* pada seluruh kolom-kolom dengan nama yang sama.**
- **Gunakan klausa ON untuk menentukkan kondisi-kondisi perubahan atau menentukan kolom-kolom untuk digabungkan.**
- **Kondisi penggabungan adalah dipisahkan dari kondisi-kondisi pencarian lain.**
- **Klausa ON membuat kode mudah untuk dipahami.**

Klausa ON

Gunakan klausa ON untuk menentukan suatu kondisi join. ON membiarkan Anda menentukan kondisi-kondisi pengabunggan terpisah dari setiap kondisi-kondisi pencarian atau penyaringan pada klausa WHERE.

Mengambil *Record-Record* dengan Klausa ON

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.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

...

19 rows selected.

Membuat *Join-Join* dengan Klausa ON

Pada contoh ini, kolom DEPARTMENT_ID dalam tabel EMPLOYEES dan DEPARTMENTS digabungkan dengan menggunakan klausa ON. Dimanapun suatu nomor departemen dalam tabel EMPLOYEES sama dengan suatu nomor departemen dalam tabel DEPARTEMENS, baris dikembalikan.

Anda juga dapat menggunakan klausa ON untuk menggabungkan kolom-kolom yang memiliki nama-nama yang berbeda.

Self-Joins Menggunakan Klausa ON

EMPLOYEES (WORKER)

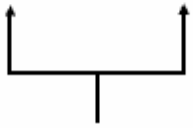
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

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

...



MANAGER_ID di dalam tabel WORKER adalah sama dengan EMPLOYEE_ID di dalam tabel MANAGER.

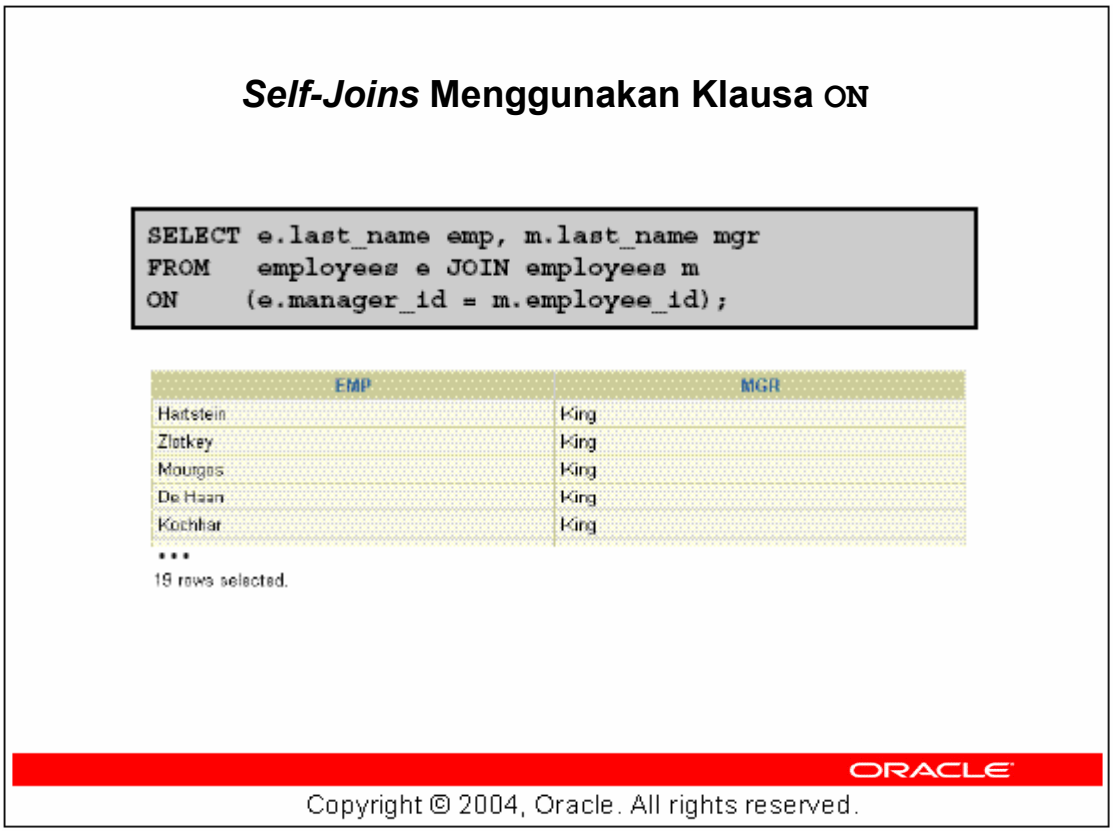
ORACLE

Menggabungkan Suatu Tabel dengan Tabel itu Sendiri

Kadangkala Anda perlu untuk menggabungkan suatu tabel dengan tabel itu sendiri. Untuk mencari nama manager dari masing-masing pegawai, Anda perlu untuk menggabungkan tabel EMPLOYEES dengan tabel EMPLOYEES sendiri, atau melakukan suatu *self- join*. Sebagai contoh, untuk mencari nama manager dari Lorentz, Anda perlu untuk :

- Mencari nama Lorentz dalam tabel EMPLOYEES dengan mencarinya di kolom LAST_NAME.
- Mencari nomor manager dari Lorentz dengan mencarinya di kolom MANAGER_ID. Nomor manager Lorentz adalah 103.
- Mencari nama manager dengan EMPLOYEE_ID 103 dengan mencarinya di kolom LAST_NAME. Hunold adalah pegawai dengan nomor pegawai 103, jadi Hunold adalah manager dari Lorentz.

Pada proses ini, Anda mencarinya dalam tabel dua kali. Yang pertama mencari nama Lorentz di kolom LAST_NAME dan nilai dari 103 di MANAGER _ID. Kedua saat Anda mencari nomor 103 di kolom EMPLOYEE_ID dan mencari nama Hunold di kolom LAST_NAME.



Menggabungkan Suatu Tabel dengan Tabel itu Sendiri (lanjutan)

Klausa `ON` juga dapat digunakan untuk menggabungkan kolom-kolom yang memiliki nama-nama yang berbeda, di dalam tabel yang sama atau di tabel yang berbeda.

Pada contoh ditunjukkan suatu *self- join* dari tabel `EMPLOYEES`, berdasarkan pada kolom-kolom `EMPLOYEE_ID` dan `MANAGER_ID`.

Menerapkan Kondisi-Kondisi Tambahan untuk Suatu *Join*

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

ORACLE

Menerapkan Kondisi-Kondisi Tambahan pada Suatu *Join*

Anda dapat menerapkan kondisi-kondisi tambahan untuk *join*.

Pada contoh ditunjukkan suatu *join* pada tabel-tabel `EMPLOYEES` dan `DEPARTMENTS` dan, sebagai tambahan, ditampilkan hanya pegawai-pegawai yang memiliki nomor manager 149. Untuk menambah penambahan kondisi-kondisi pada klausa `ON`, Anda dapat menambahkan klausa-klausa `AND`. Sebagai alternatif, Anda dapat menggunakan klausa `WHERE` untuk menerapkan tambahan kondisi-kondisi :

```
SELECT    e.employee_id, e.last_name, e.department_id,
          d.department_id, d.location_id
FROM      employees e JOIN departments d
ON        (e.department_id = d.department_id)
WHERE     e.manager_id = 149;
```

Membuat *Three-Way Joins* dengan Klausa ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

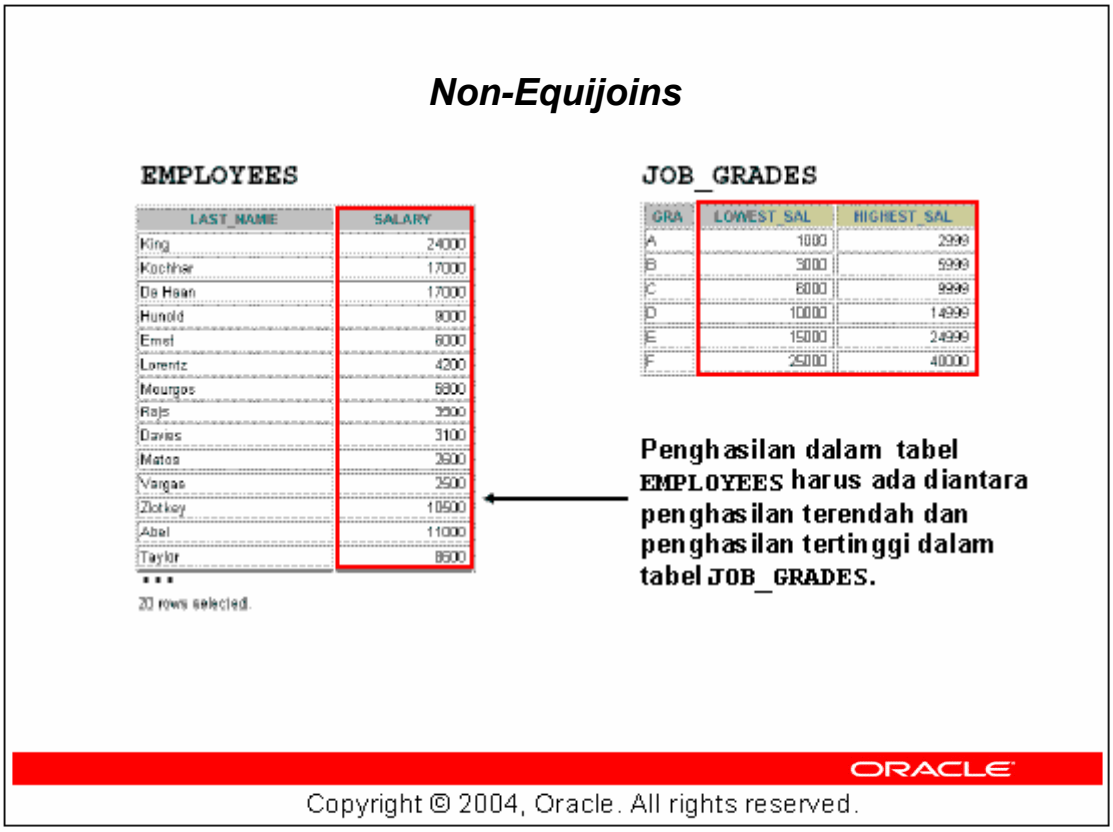
...

19 rows selected.

ORACLE

Three-Way Joins

Suatu *Three-Way join* adalah suatu *join* pada tiga buah tabel. Dalam acuan sintak SQL:1999, *join-join* dilakukan dari kiri ke kanan. Jadi *join* pertama dilakukan `EMPLOYEES JOIN DEPARTMENTS`. Kondisi *join* pertama dapat merujuk kolom-kolom pada tabel `EMPLOYEES` dan `DEPARTMENTS` tapi tidak bisa merujuk ke kolom-kolom pada `LOCATIONS`. Kondisi *join* kedua dapat merujuk kolom-kolom pada ketiga tabel.



Non-Equijoins

Suatu *non-equijoins* adalah suatu kondisi *join* yang berisi suatu operator ke-sama-an (*equality*).

Hubungan antara tabel `EMPLOYEES` dengan tabel `JOB_GRADES` adalah suatu contoh dari n*on-equijoins.* Hubungan antara kedua kolom adalah bahwa kolom `SALARY` dalam tabel `EMPLOYEES` harus berada diantara nilai-nilai di kolom-kolom `LOWEST_SALARY` dan `HIGEST_SALARY` dalam tabel `JOB_GRADES`. Hubungan diperoleh menggunakan suatu operator lain daripada ke-sama-an (=).

Mendapatkan *Record-Record* dengan *Non-Equijoins*

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     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
Raja	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...
20 rows selected.

ORACLE

Non-Equijoins (lanjutan)

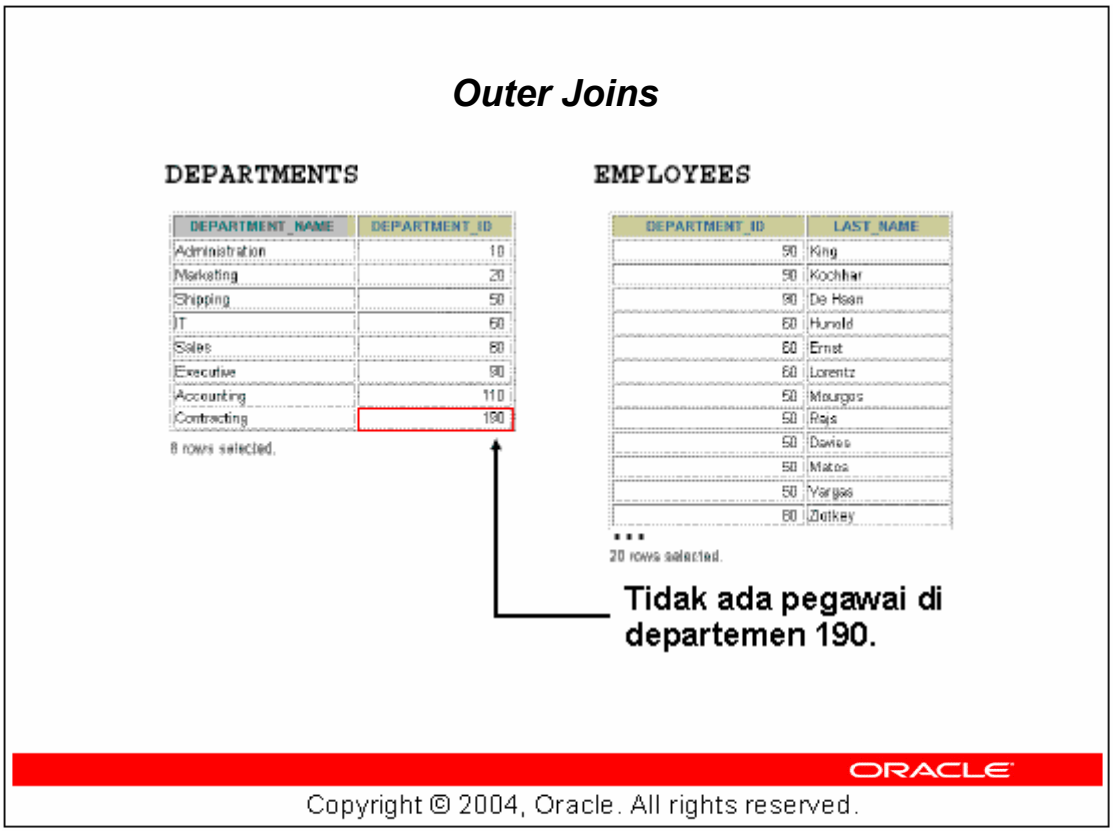
Contoh pada slide membuat suatu *non-equijoins* untuk mengevaluasi suatu tingkat penghasilan para pegawai. Penghasilan harus ada di *antara* pada rentang penghasilan terendah dan penghasilan tertinggi.

Penting untuk diperhatikan bahwa semua pegawai hanya muncul sekali saat query ini dijalankan. Tidak ada nama pegawai yang diulang dalam daftar. Ada dua alasan untuk itu :

- Tidak satupun baris dalam tabel job grade yang berisi tingkatan-tingkatan yang *overlap*. Maka, nilai penghasilan untuk seorang pegawai bisa berada hanya di antara salah satu baris-baris penghasilan terendah dan penghasilan tertinggi dalam tabel tingkat gaji.
- Semua penghasilan para pegawai berada dalam batas-batas yang disediakan oleh tabel job grade. Maka, tidak akan ada pegawai mendapat penghasilan kurang dari nilai terendah yang termasuk pada kolom `LOWEST_SAL` atau lebih dari nilai tertinggi yang termasuk pada kolom `HIGHEST_SAL`.

Catatan: Kondisi-kondisi lain (seperti <= dan >=) dapat digunakan, tapi lebih sederhana menggunakan `BETWEEN`. Ingat untuk menentukan lebih dulu nilai terendah dan kemudian nilai tertinggi saat menggunakan `BETWEEN`.

Tabel-tabel alias digunakan pada contoh slide untuk alasan performa, bukan karena kemungkinan ambiguitas.



Mengembalikan *Record-Record* yang Tidak Secara Langsung Sesuai dengan *Outer Joins*

Jika suatu baris tidak memenuhi kondisi *join*, baris tidak muncul pada hasil query. Sebagai contoh, dalam kondisi *equijoin* pada tabel-tabel `EMPLOYEES` dan `DEPARTMENTS`, nomor department 190 tidak muncul karena tidak ada pegawai dengan nomor department tersebut dicatat dalam tabel `EMPLOYEES`. Malahan tampak 20 orang pegawai pada sekelompok hasil, Anda lihat 19 *record*.

Untuk mengembalikan catatan departemen yang tidak mempunyai beberapa pegawai, Anda bisa gunakan *outer join*.

INNER *JOIN* VERSUS OUTER *JOIN*

- **Dalam SQL:1999, penggabungan dua tabel mengembalikan hanya baris-baris yang sesuai disebut *inner join*.**
- **Suatu penggabungan antara dua tabel yang mengembalikan hasil-hasil dari *inner join* seperti baris-baris yang tidak sesuai dari tabel-tabel sebelah kiri (atau kanan) disebut *left* (atau *right*) *outer join*.**
- **Suatu penggabungan dua tabel yang mengembalikan hasil-hasil dari suatu *inner join* seperti hasil-hasil dari sebelah kiri dan sebelah kanan adalah *full outer join*.**

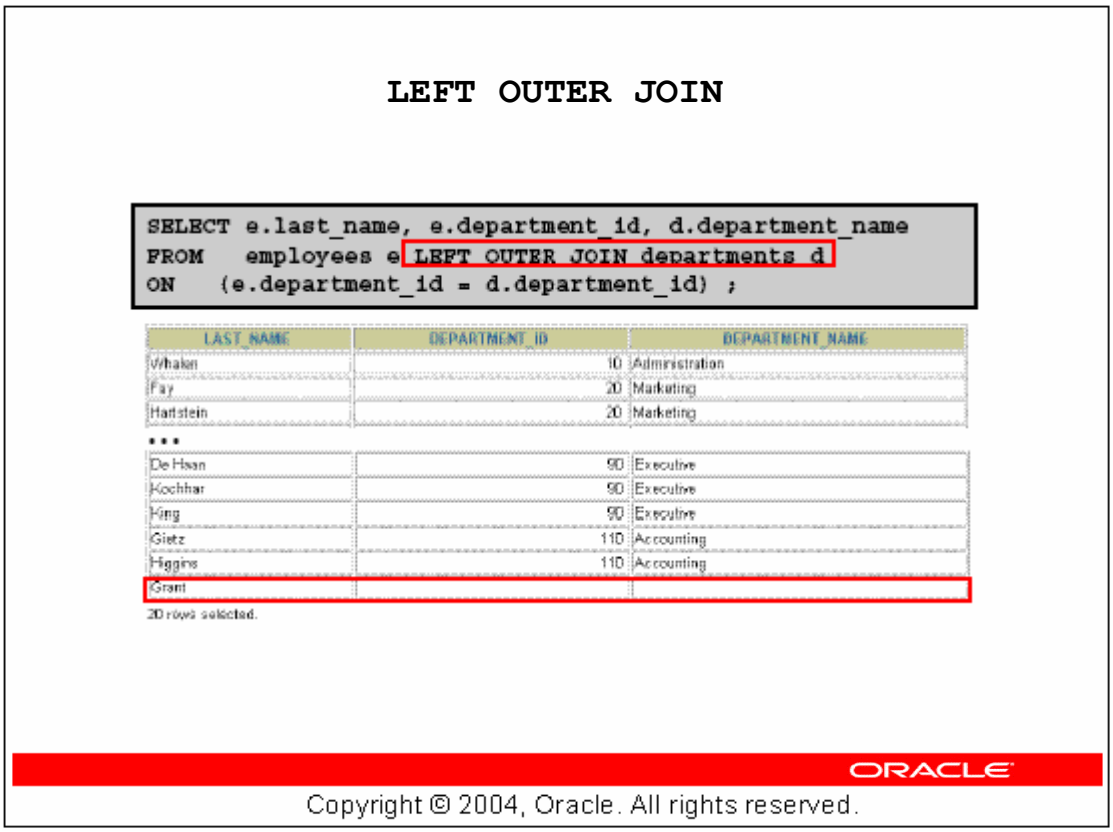
ORACLE

INNER *Join* Versus OUTER *Join*

Menggabungkan tabel-tabel dengan klausa-klausa `NATURAL JOIN`, `USING` atau `ON` menghasilkan suatu *inner join*. Beberapa baris yang tidak sesuai tidak ditampilkan pada output. Untuk mengembalikan baris-baris yang tidak sesuai, Anda bisa menggunakan suatu *outer join*. Suatu *outer join* mengembalikan semua baris-baris yang memenuhi kondisi *join* dan juga mengembalikan beberapa atau semua baris-baris tersebut dari satu tabel yang mana tidak ada baris-baris dari tabel lain yang memenuhi kondisi *join*.

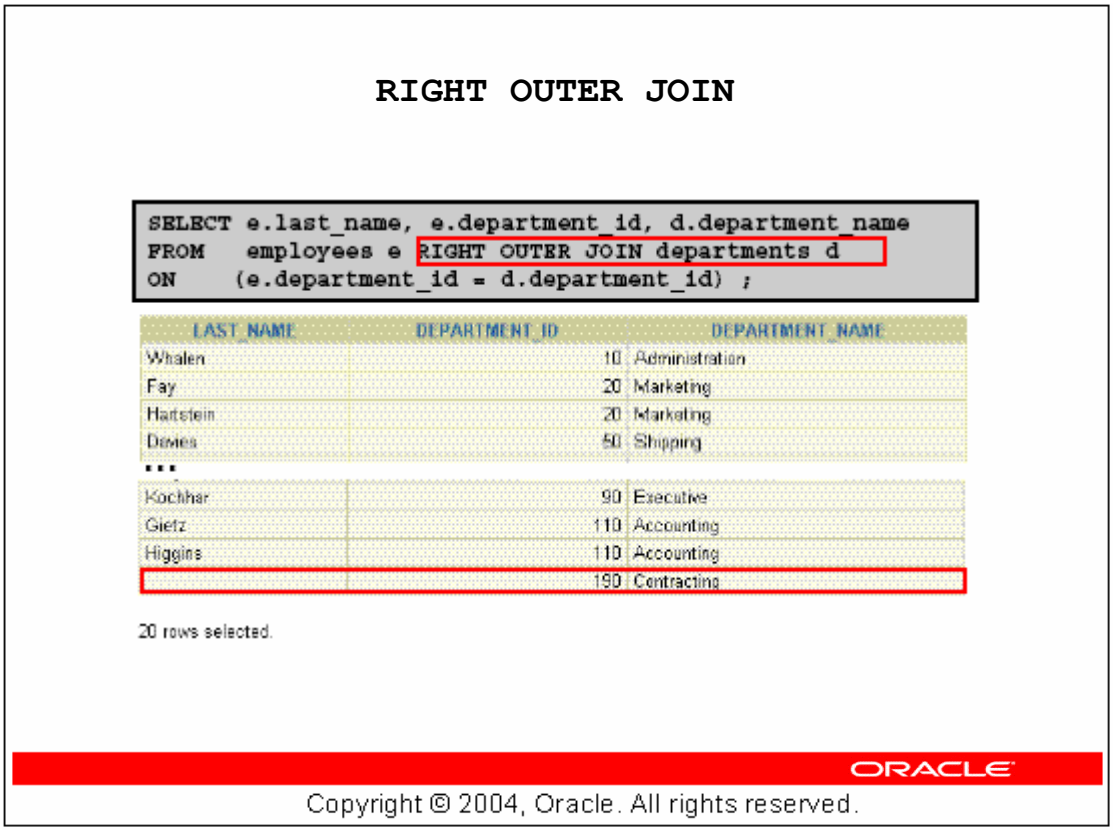
Ada tiga tipe dari *outer join*:

- `LEFT OUTER`
- `RIGHT OUTER`
- `FULL OUTER`



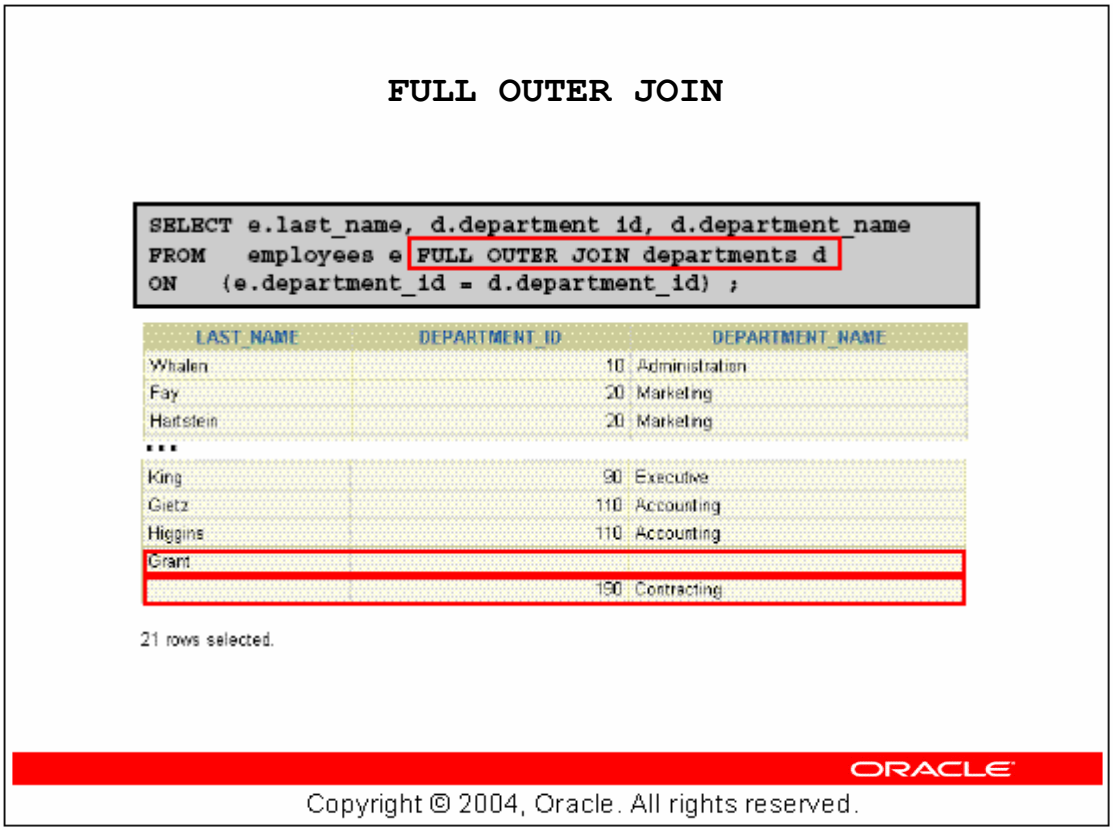
Contoh dari `LEFT OUTER JOIN`

Query ini mendapatkan semua baris dalam tabel `EMPLOYEES`, yang ada disebelah kiri tabel meskipun tabel `EMPLOYEES` tidak ada yang sesuai dalam tabel `DEPARTEMENTS`.



Contoh dari `RIGHT OUTER JOIN`

Query ini mendapatkan semua baris dalam tabel `DEPARTMENTS`, yang ada disebelah kanan tabel meskipun dalam tabel `DEPARTMENTS` tidak ada yang sesuai dalam tabel `EMPLOYEES`.



Contoh dari `FULL OUTER JOIN`

Query ini mendapatkan semua baris dalam tabel `EMPLOYEES`, meskipun tabel EMPLOYEES tidak ada yang sesuai dalam tabel `DEPARTMENTS`. Query ini juga mendapatkan semua baris dalam tabel `DEPARTMENTS`, meskipun tabel `DEPARTMENTS` tidak ada yang sesuai dalam tabel `EMPLOYEES`.

Cartesian Products

- **Suatu *Cartesian product* terbentuk ketika :**
  - **Suatu kondisi *join* dihilangkan**
  - **Suatu kondisi *join* tidak tepat**
  - **Seluruh baris dalam tabel pertama digabungkan ke seluruh baris dalam tabel kedua**
- **Untuk menghindari suatu *Cartesian product*, selalu sertakan sebuah kondisi *join* yang tepat.**

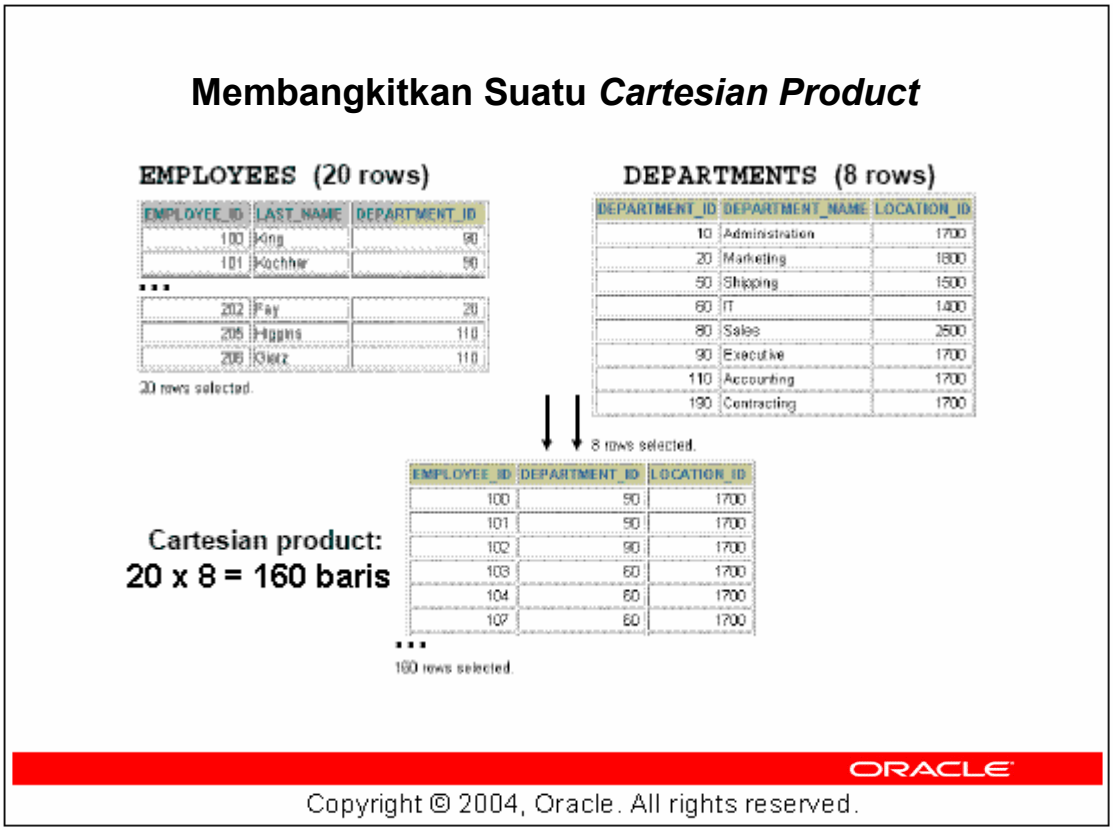
ORACLE

Cartesian Products

Ketika suatu kondisi *join* tidak tepat atau dihilangkan semua, hasilnya adalah *Cartesian product,* dimana semua kombinasi dari baris-baris ditampilkan. Semua baris dalam tabel pertama digabungkan dengan semua baris dalam tabel kedua.

Sebuah *Cartesian product* cenderung untuk menghasilkan sejumlah besar baris, dan hasilnya kurang bermanfaat. Anda sebaiknya selalu menyertakan kondisi *join* yang tepat kecuali jika Anda mempunyai kebutuhan tertentu untuk menggabungkan semua baris dari seluruh tabel.

Cartesian product berguna untuk beberapa uji coba ketika Anda perlu untuk menghasilkan sejumlah besar baris untuk memperagakan sejumlah data yang beralasan.



Cartesian Products (lanjutan)

Sebuah *Cartesian product* dihasilkan jika suatu kondisi *join* dihilangkan. Contoh pada slide menampilkan nama belakang pegawai dan nama departemen dari tabel-tabel `EMPLOYEES` dan `DEPARTMENTS`. Karena tidak ada kondisi *join* yang ditentukan, semua baris (20 baris) dari tabel `EMPLOYEES` digabungkan dengan semua baris (8 baris) dalam tabel `DEPARTMENTS`, sehingga menghasilkan 160 baris dalam output.

Membuat *Cross Join*

- **Klausa `CROSS JOIN` menghasilkan *cross-product* dari dua tabel.**
- **`CROSS JOIN` disebut juga *Cartesian product* antara dua tabel.**

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

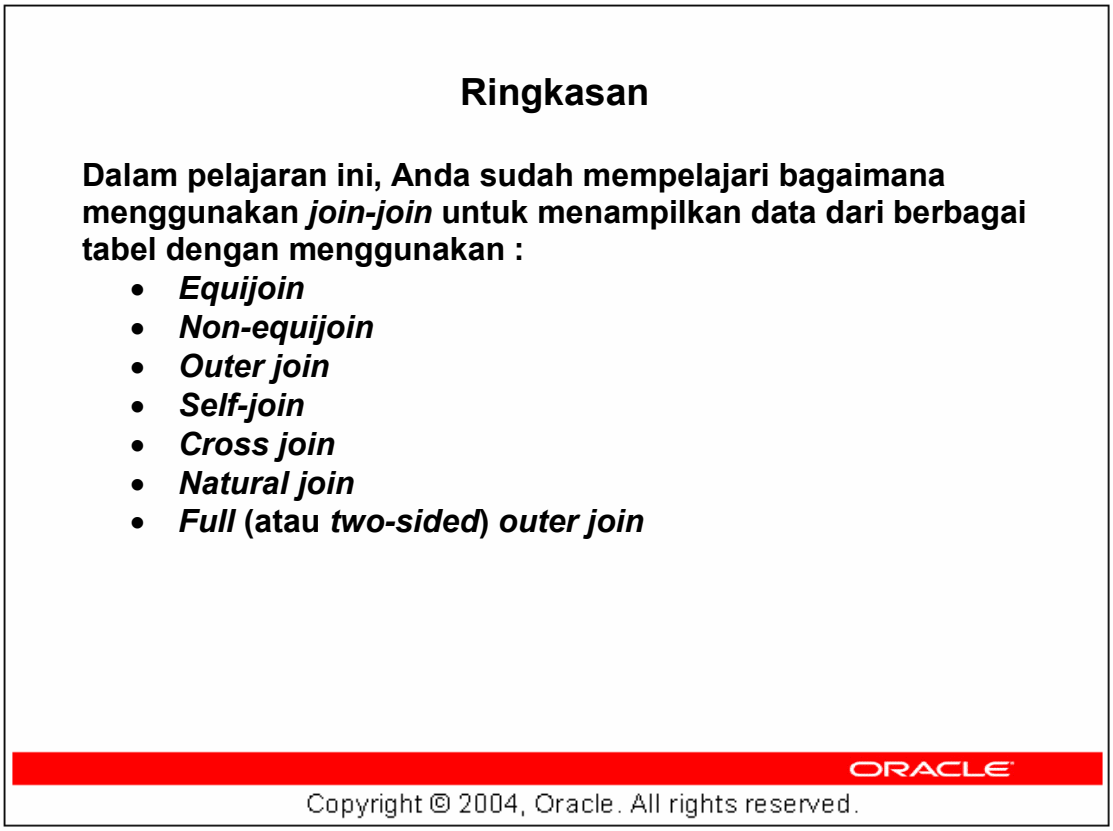
LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

...

160 rows selected.

Membuat Cross Joins

Contoh pada slide menghasilkan suatu *Cartesian product* dari tabel-tabel `EMPLOYEES` dan `DEPARTMENTS`.



Ringkasan

Ada berbagai macam cara untuk menggabungkan tabel-tabel.

Tipe-tipe dari *join-join* :

- *Equijoins*
- *Non-equijoins*
- *Outer joins*
- *Self-joins*
- *Cross joins*
- *Natural joins*
- *Full* (atau *two-sided*) *outer joins*

Cartesian Products

Suatu *Cartesian product* menghasilkan suatu tampilan dari semua kombinasi baris-baris. Ini dilakukan baik dengan menghilangkan klausa `WHERE` atau menentukan klausa `CROSS JOIN`.

Tabel-tabel Alias

- Tabel-tabel alias mempercepat akses database.
- Tabel-tabel alias dapat membantu menjaga kode SQL lebih kecil dengan menghemat memory.