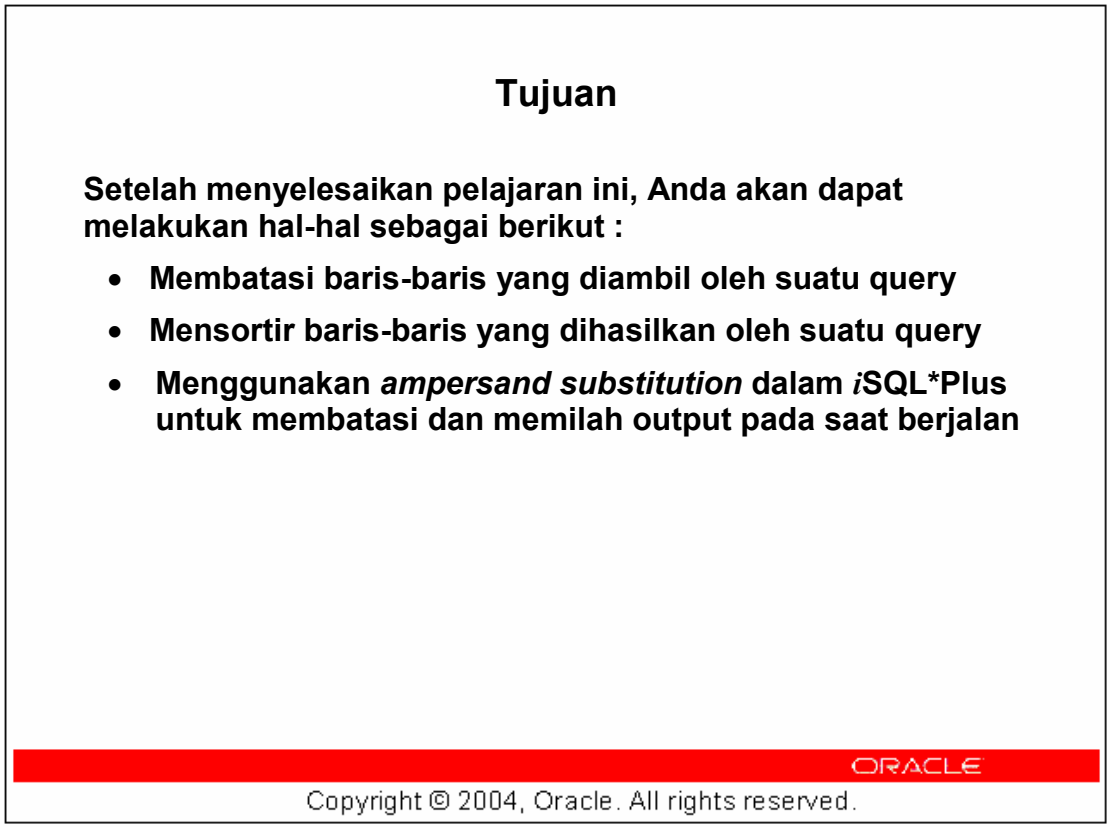
2

Membatasi dan Mensortir Data

ORACLE



Tujuan

Saat data diambil dari database, Anda mungkin perlu untuk melakukan hal berikut :

- Membatasi baris-baris pada data yang ditampilkan
- Menentukan urutan baris-baris mana yang akan ditampilkan

Pelajaran ini menjelaskan pernyataan SQL yang Anda gunakan untuk menjalankan aksi-aksi tersebut.

Membatasi Baris-Baris Menggunakan Suatu *Selection*

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

...

20 rows selected.

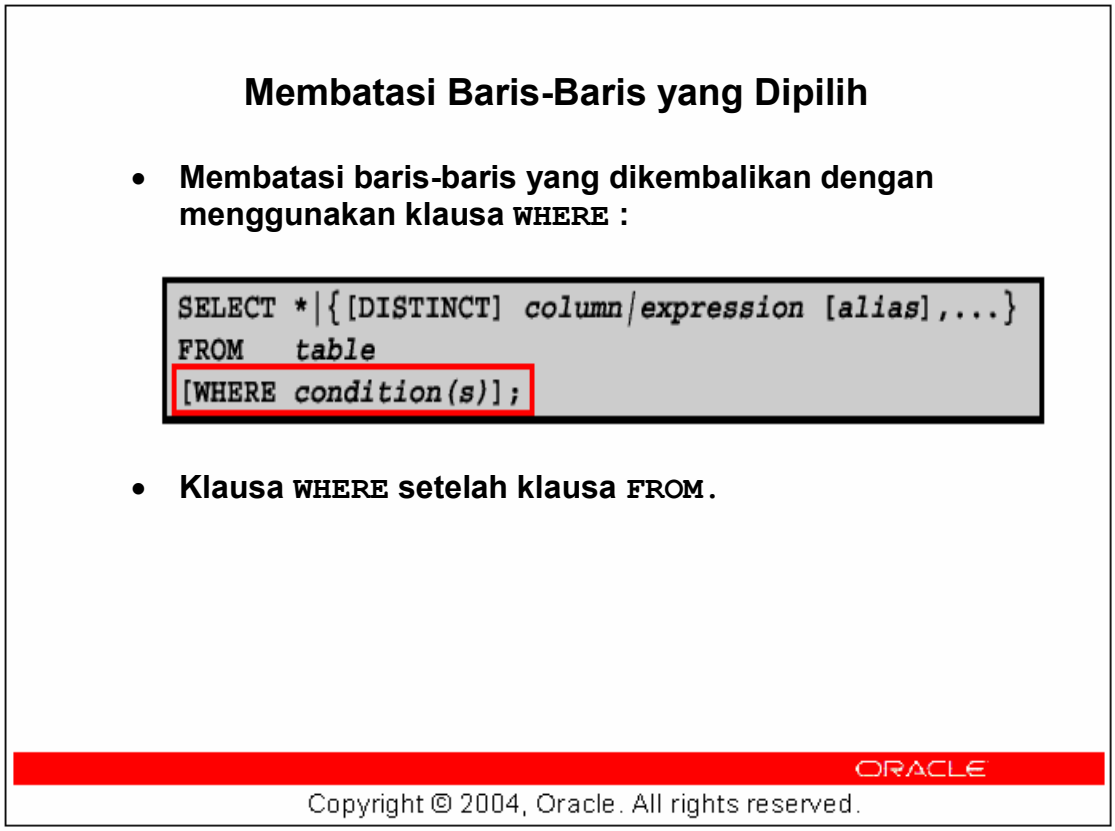
"Hasil seluruh pegawai di departemen 90"

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

ORACLE

Membatasi Baris-Baris dengan Menggunakan Suatu *Selection*

Contoh pada slide, misalnya Anda ingin menampilkan semua nama pegawai pada departemen 90. Baris-baris dengan suatu nilai 90 pada kolom `DEPARTEMENT_ID` adalah satu-satunya yang dikembalikan. Metode pembatasan ini adalah dasar dari klausa `WHERE` dalam SQL.



Membatasi Baris-Baris yang Dipilih

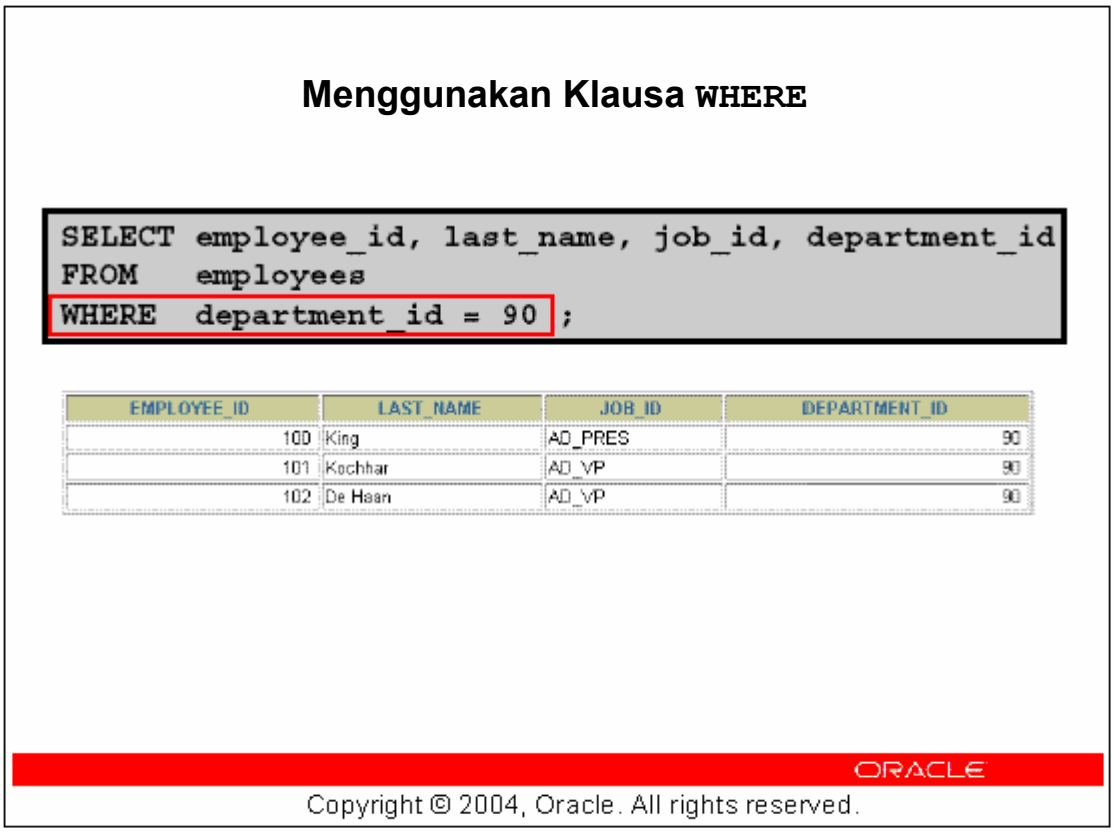
Anda bisa membatasi baris-baris yang dihasilkan dari query dengan menggunakan klausa `WHERE`. Suatu klausa `WHERE` berisi suatu kondisi yang harus terpenuhi, dan tepat setelah klausa `FROM`. Jika kondisinya benar, baris yang memenuhi kondisi dikembalikan.

Dalam sintak:

`WHERE`	membatasi query ke baris-baris yang memenuhi kondisi
`condition`	susunan nama-nama kolom, ekspresi-ekspresi, konstanta-konstanta dan operator pembanding

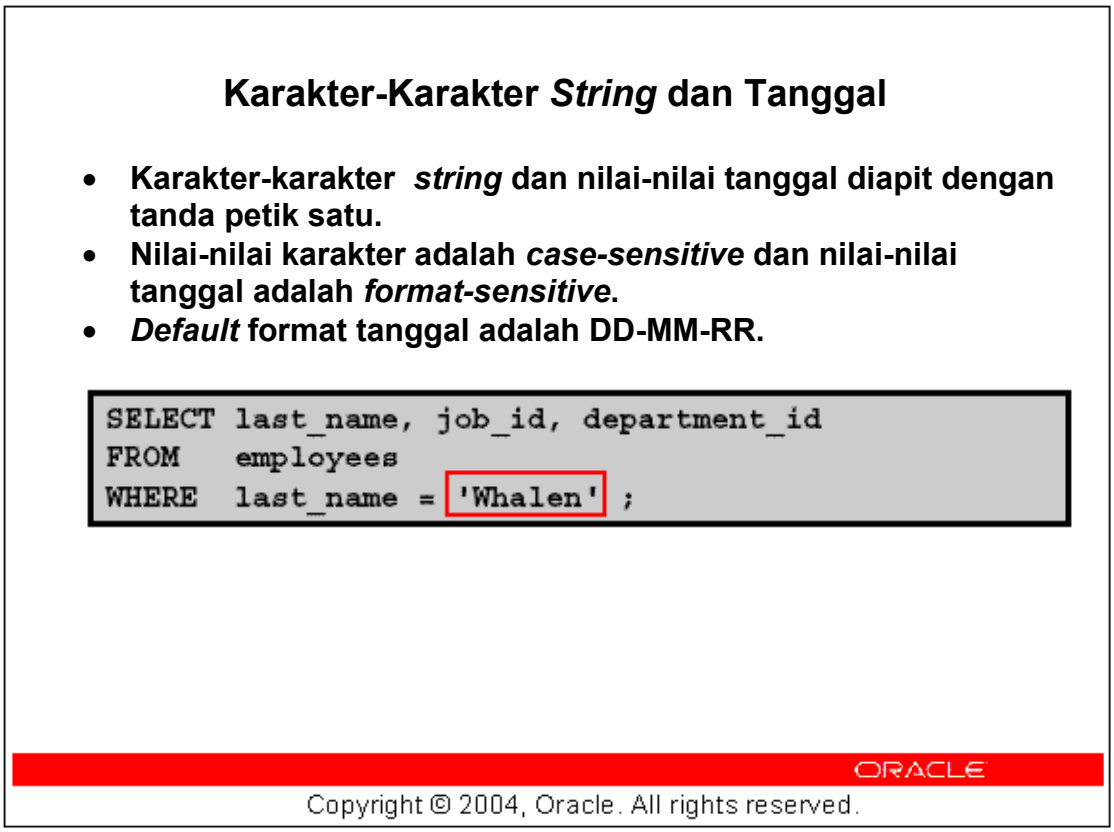
Klausa `WHERE` dapat membandingkan nilai-nilai dalam kolom-kolom, nilai-nilai literal, ekspresi-ekspresi aritmatika atau fungsi-fungsi (*functions*). Klausa `WHERE` terdiri dari 3 bagian :

- Nama kolom
- Kondisi pembanding
- Nama kolom, konstanta atau daftar nilai-nilai



Menggunakan Klausa `WHERE`

Pada contoh, pernyataan `SELECT` menghasilkan kode pegawai, nama, job ID dan nomor departemen dari semua pegawai yang ada di departement 90.



Karakter-Karakter *String* dan Tanggal

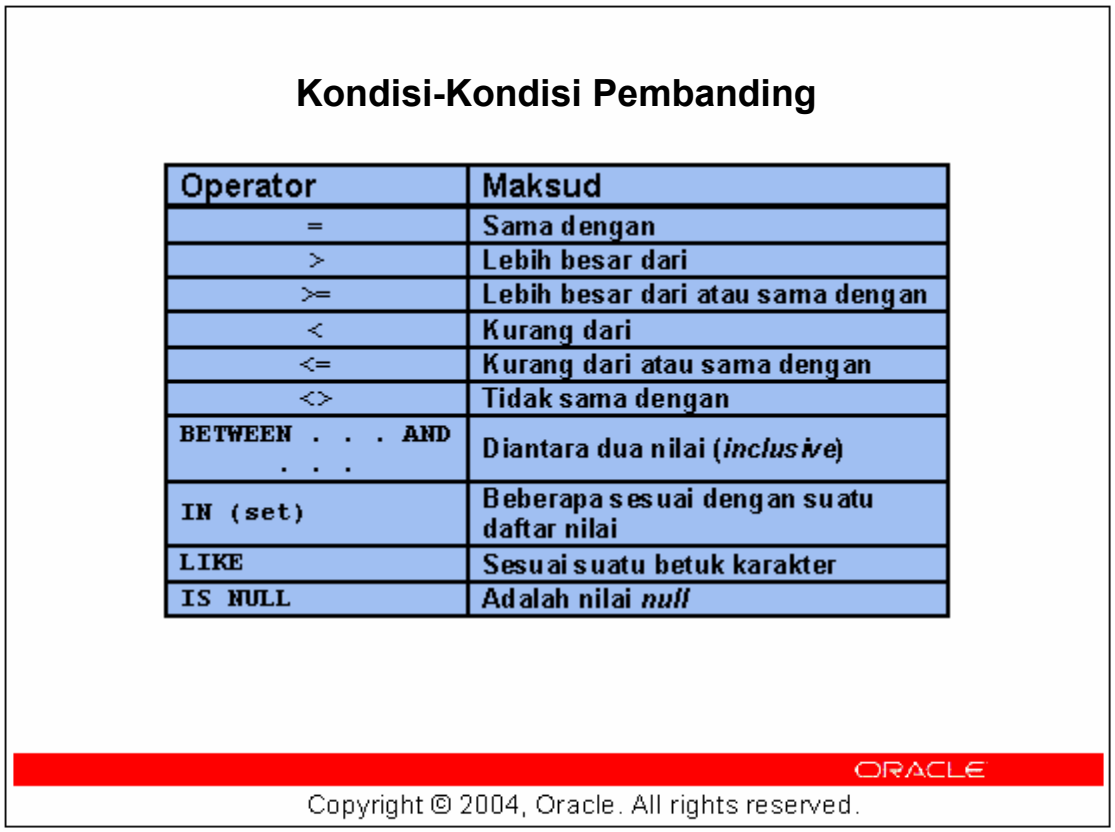
Karakter-karakter *string* (rangkaian) dan tanggal dalam klausa `WHERE` harus diapit dengan tanda petik satu (`' '`). Meskipun demikian, konstanta angka tidak harus diapit oleh tanda petik satu.

Semua pencarian karakter adalah *case-sensitive*. Dalam contoh berikut, tidak ada baris-baris yang dihasilkan karena tabel `EMPLOYEES` menyimpan semua nama belakang dalam bentuk campuran (*mixed case*):

```
SELECT   last_name, job_id, departement_id
FROM     employees
WHERE    last_name = 'WHALEN';
```

Database Oracle menyimpan tanggal dengan format angka sendiri, yang menunjukkan abad, tahun, bulan, hari, jam, menit dan detik. *Default* tanggal yang ditampilkan adalah DD-MON-RR.

Catatan : :Lebih rinci tentang format RR dan tentang merubah *default* format tanggal, lihat pelajaran selanjutnya.



Kondisi-Kondisi Pembanding

Kondisi-kondisi pembanding adalah digunakan dalam kondisi-kondisi yang membandingkan satu ekspresi dengan nilai atau ekspresi lain. Kondisi-kondisi tersebut digunakan dalam klausa `WHERE` dengan format sebagai berikut :

Sintak :

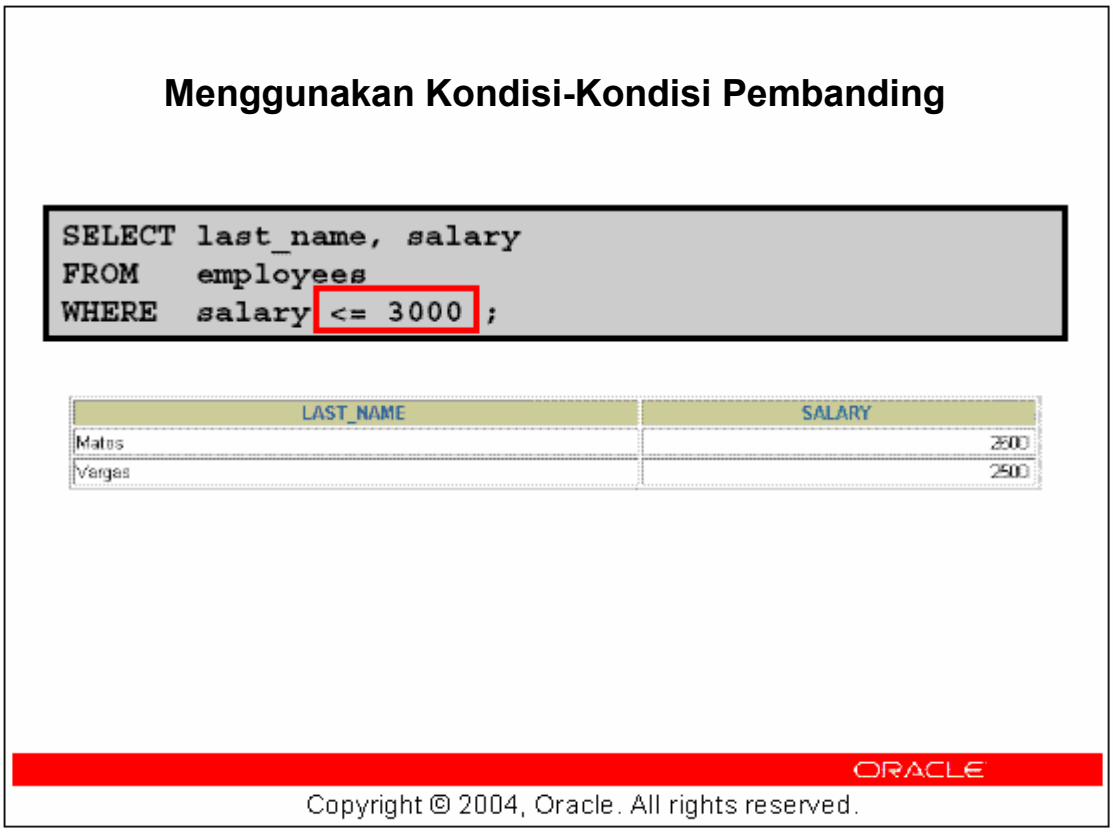
```
. . . WHERE expr operator value
```

Contoh

```
. . . WHERE hire_date = '01-JAN-95'
. . . WHERE salary >= 6000
. . . WHERE last_name = 'Smith'
```

Suatu alias tidak bisa digunakan dalam klausa `WHERE`.

Catatan : Simbol-simbol `!=` dan **^=** juga dapat menunjukkan kondisi tidak sama dengan.



Menggunakan Kondisi-Kondisi Pembanding

Dalam contoh, pernyataan `SELECT` mengambil nama belakang dan penghasilan dari tabel `EMPLOYEES` untuk beberapa pegawai yang mempunyai penghasilan kurang atau sama dengan $3,000. Catatan bahwa disediakan nilai eksplisit pada klausa `WHERE`. Nilai eksplisit `3000` dibandingkan dengan nilai penghasilan pada kolom `SALARY` di tabel `EMPLOYEES`.

Menggunakan Kondisi BETWEEN

Gunakan kondisi BETWEEN untuk menampilkan baris-baris berdasarkan rentang nilai:

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

Batas bawah Batas atas

LAST_NAME	SALARY
Raj	3500
Davies	3100
Matos	2600
Vargas	2500

ORACLE

Menggunakan Kondisi BETWEEN

Anda dapat menampilkan baris-baris berdasarkan rentang nilai menggunakan kondisi rentang (*range*) `BETWEEN`. Rentang yang Anda tentukan terdiri satu batas bawah dan satu batas atas.

Pernyataan `SELECT` pada slide menghasilkan baris-baris dari tabel `EMPLOYEES` untuk beberapa pegawai yang mempunyai penghasilan antara $2,500 dan $3,500.

Nilai-nilai yang telah ditentukan pada kondisi `BETWEEN` adalah *inclusive* (yang termasuk). Anda harus menentukan batas bawah dulu.

Anda juga bisa menggunakan kondisi `BETWEEN` untuk nilai-nilai huruf (karakter) :

```
SELECT last_name
FROM   employees
WHERE  last_name BETWEEN 'King' AND 'Smith';
```

Menggunakan Kondisi IN

Gunakan IN *membership condition* untuk menguji nilai-nilai dalam suatu daftar :

```
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.

ORACLE

Menggunakan Kondisi IN

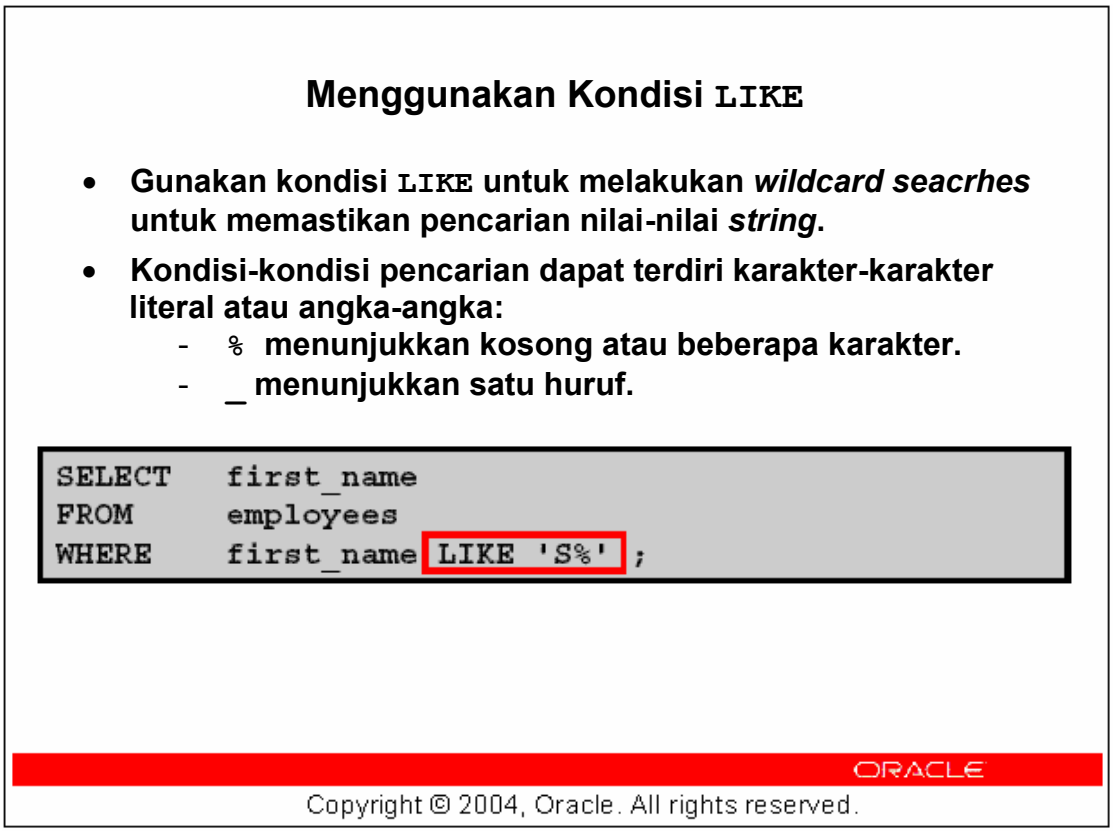
Untuk menguji nilai dalam suatu kelompok nilai-nilai tertentu, gunakan kondisi IN. Kondisi IN disebut juga sebagai *membership condition* .

Contoh pada slide menampilkan nomor pegawai, nama belakang, penghasilan dan nomor manajer para pegawai untuk semua pegawai dimana nomor manajer para pegawai adalah 100, 101 atau 201.

Kondisi IN dapat digunakan pada setiap tipe data. Contoh berikut ini menghasilkan suatu baris dari tabel EMPLOYEES untuk beberapa pegawai yang mempunyai nama belakang termasuk dalam daftar nama-nama pada klausa WHERE :

```
SELECT employee_id, manager_id, departement_id
FROM   employees
WHERE  last_name IN ('Hartstein','Vargas');
```

Jika karakter-karakter atau tanggal digunakan dalam daftar, harus diapit oleh tanda petik satu (' ').



Menggunakan Kondisi `LIKE`

Anda mungkin tidak selalu tahu nilai pasti yang dicari . Anda dapat memilih baris-baris yang sesuai dengan pola karakter dengan menggunakan kondisi `LIKE`. Cara kerja (operation) pola karakter-yang bersesuaian adalah seperti mengacu ke pencarian acak (*wildcard search*). Dua simbol yang dapat digunakan untuk membuat pencarian *string* (rangkaian).

Simbol	Keterangan
%	Mewakili setiap urutan kosong atau beberapa karakter
_	Mewakili setiap karakter tunggal

Pernyataan `SELECT` pada slide menghasilkan nama depan pegawai dari tabel `EMPLOYEES` untuk beberapa pegawai yang memiliki nama depan yang diawali dengan huruf *S*. Catatan huruf besar *S*. Nama-nama yang diawali dengan suatu huruf *s* tidak ditampilkan.

Kondisi `LIKE` dapat digunakan sebagai suatu *shortcut* (jalan pintas) untuk beberapa perbandingan `BETWEEN`. Contoh berikut menampilkan nama belakang dan tanggal mulai bekerja dari semua pegawai yang bergabung antara bulan Januari 1995 dan Desember 1995 :

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date LIKE '%95';
```

Menggunakan Kondisi `LIKE`

- **Anda bisa mengkombinasikan karakter-karakter *pattern-matching* (pola bersesuaian):**

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

LAST_NAME
Kochhar
Lorentz
Mourgos

- **Anda dapat menggunakan *identifier* `ESCAPE` untuk mencari kenyataan simbol-simbol % dan _ .**

ORACLE

Memgkombinasikan Pencarian Acak (*Wildcard*) Karakter-Karakter

Simbol-simbol `%` dan _ dapat digunakan pada setiap kombinasi dengan karakter-karakter literal. Contoh pada slide menampilkan nama-nama dari semua pegawai yang memiliki nama belakang dengan huruf *o* sebagai karakter kedua.

Opsi `ESCAPE`

Saat Anda perlu untuk mendapatkan kepastian dari karakter-karakter % dan _, gunakan pilihan `ESCAPE`. Pilihan ini menentukan keluaran karakter (*escape character*) apa. Jika Anda ingin mencari rangkaian-rangkaian (*strings*) yang berisi 'SA_' , Anda dapat menggunakan pernyataan SQL berikut :

```
SELECT employee_id, last_name, job_id
FROM employees WHERE job_id LIKE '%SA\_%' ESCAPE '\';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

Opsi `ESCAPE` ditandai *backslash* (\) sebagai keluaran karakter. Dalam pola, keluaran karakter diawali *underscore* (_). Hal ini menyebabkan server Oracle menafsirkan *underscore* secara harfiah (*literally*).

Menggunakan Kondisi-Kondisi NULL

Menguji *null* dengan operator `IS NULL`.

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

LAST_NAME	MANAGER_ID
King	

ORACLE

Menggunakan Kondisi-Kondisi NULL

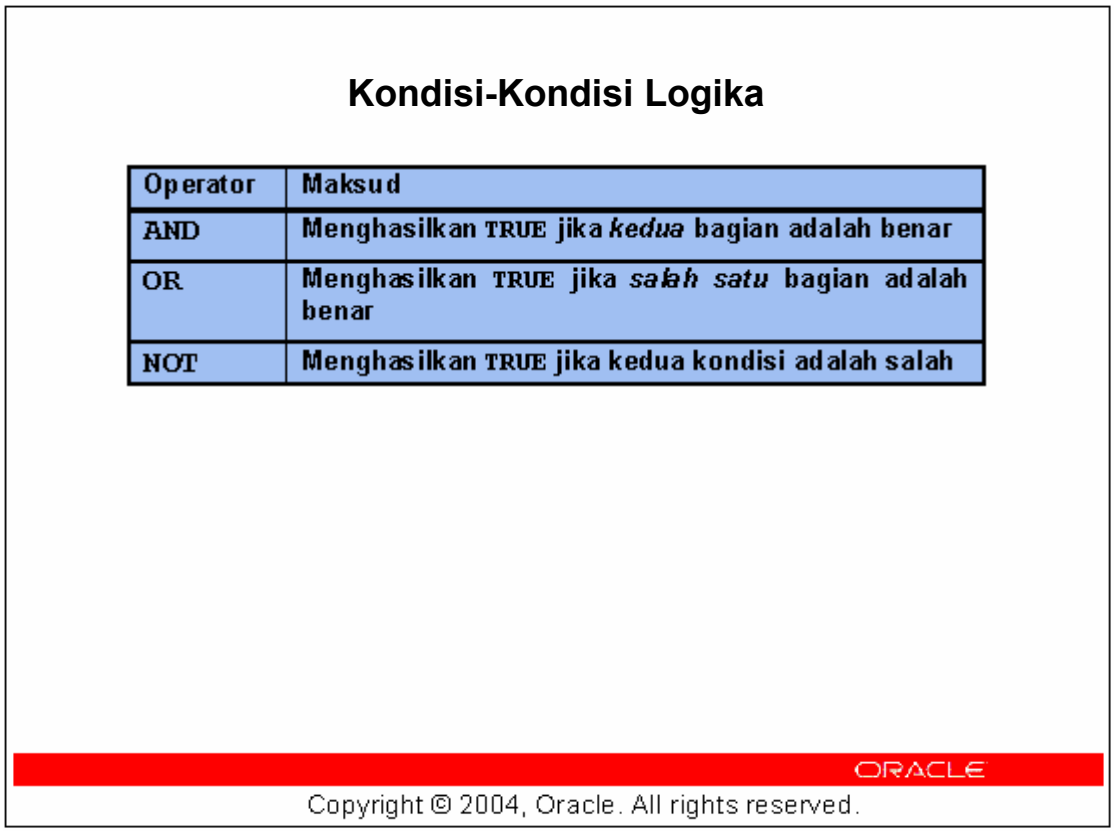
Kondisi-kondisi `NULL` terdiri dari kondisi `IS NULL` dan kondisi `IS NOT NULL`. Kondisi `IS NULL` untuk menguji *null-null*. Nilai *null* berarti nilainya tidak ada (*unavailable*), tidak diberikan (*unassigned*), tidak diketahui (*unknown*) atau tidak dipakai (*inapplicable*). Oleh sebab itu, Anda tidak bisa menguji dengan = karena suatu *null* bukan sama atau tidak sama dengan dalam sembarang nilai. Contoh pada slide menghasilkan nama belakang dan manager-manager dari para pegawai yang tidak mempunyai manager.

Berikut adalah contoh lainnya : Untuk menampilkan nama belakang, job ID dan komisi dari semua pegawai yang ditandai *tidak* untuk menerima komisi, gunakan pernyataan SQL berikut:

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

LAST_NAME	JOB_ID	COMMISION_PCT
King	AD_PRES	
Kochhar	AD_VP	
...		
Higgins	AC_MGR	
Gietz	AC_ACCOUNT	

16 rows selected.



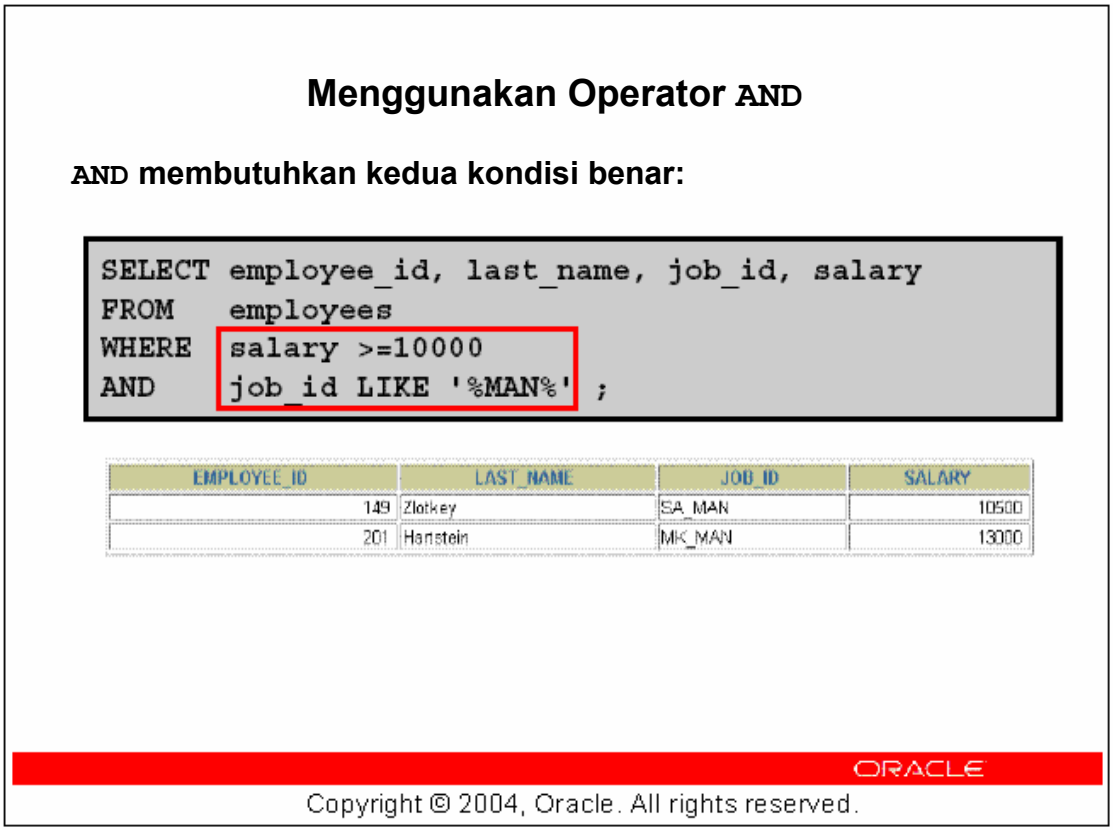
Kondisi-Kondisi Logika

Suatu kondisi logika menggabungkan hasil dari dua bagian kondisi-kondisi untuk menghasilkan hasil tunggal berdasarkan kondisi-kondisi tersebut, atau kondisi-kondisi logika membalik hasil dari suatu kondisi tunggal. Satu baris dikembalikan, hanya jika hasil secara keseluruhan dalam suatu kondisi adalah benar (*true*).

Tiga operator logika yang ada pada SQL:

- `AND`
- `OR`
- `NOT`

Semua contoh-contoh sejauh ini ditentukan hanya satu kondisi dalam klausa `WHERE`. Anda dapat menggunakan beberapa kondisi dalam satu klausa `WHERE` menggunakan operator `AND` dan `OR`.



Menggunakan operator AND

Dalam contoh, kedua kondisi harus benar untuk setiap *record* yang dipilih. Oleh sebab itu, hanya pegawai-pegawai yang mempunyai suatu nama job yang berisi rangkaian kata 'MAN' *dan* berpenghasilan $10,000 atau lebih yang dipilih.

Semua karakter yang dicari adalah *case-sensitive*. Tidak ada baris-baris yang dihasilkan jika ' MAN' bukan huruf besar. Rangkaian-rangkaian karakter harus diapit oleh tanda petik.

Tabel Kebenaran AND

Tabel berikut ini menampilkan hasil-hasil yang mengkombinasikan dua ekspresi dengan AND:

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Menggunakan Operator OR

OR membutuhkan salah satu kondisi benar :

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

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

8 rows selected.

ORACLE

Menggunakan Operator OR

Dalam contoh, salah satu kondisi bisa bernilai benar untuk sembarang *record* yang dipilih. Oleh karena itu beberapa pegawai yang mempunyai job ID yang berisi rangkaian kata 'MAN' *atau* berpenghasilan $10,000 atau lebih yang dipilih.

Tabel Kebenaran OR

Tabel berikut ini menampilkan hasil-hasil dari kombinasi dua ekspresi dengan `OR` :

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	NULL
NULL	TRUE	NULL	NULL

Menggunakan operator NOT

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

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

10 rows selected.

ORACLE

Menggunakan Operator NOT

Contoh di slide menampilakan nama belakang dan job ID untuk semua pegawai dimana job ID *bukan* `IT_PROG`, `ST_CLERK`, atau `SA_REP`.

Tabel Kebenaran NOT

Tabel berikut ini menampilkan hasil penggunaan operator NOT pada suatu kondisi:

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Catatan : Operator `NOT` bisa juga digunakan bersama operator-operator SQL yang lain, seperti `BETWEEN`, `LIKE` dan `NULL`.

```
. . . WHERE job_id      NOT IN ('AC_ACCOUNT','AD_VP')
. . . WHERE salary      NOT BETWEEN 10000 AND 15000
. . . WHERE last_name   NOT LIKE '%A%'
. . . WHERE commission_pct IS NOT NULL
```

Aturan-Aturan *Precedence*

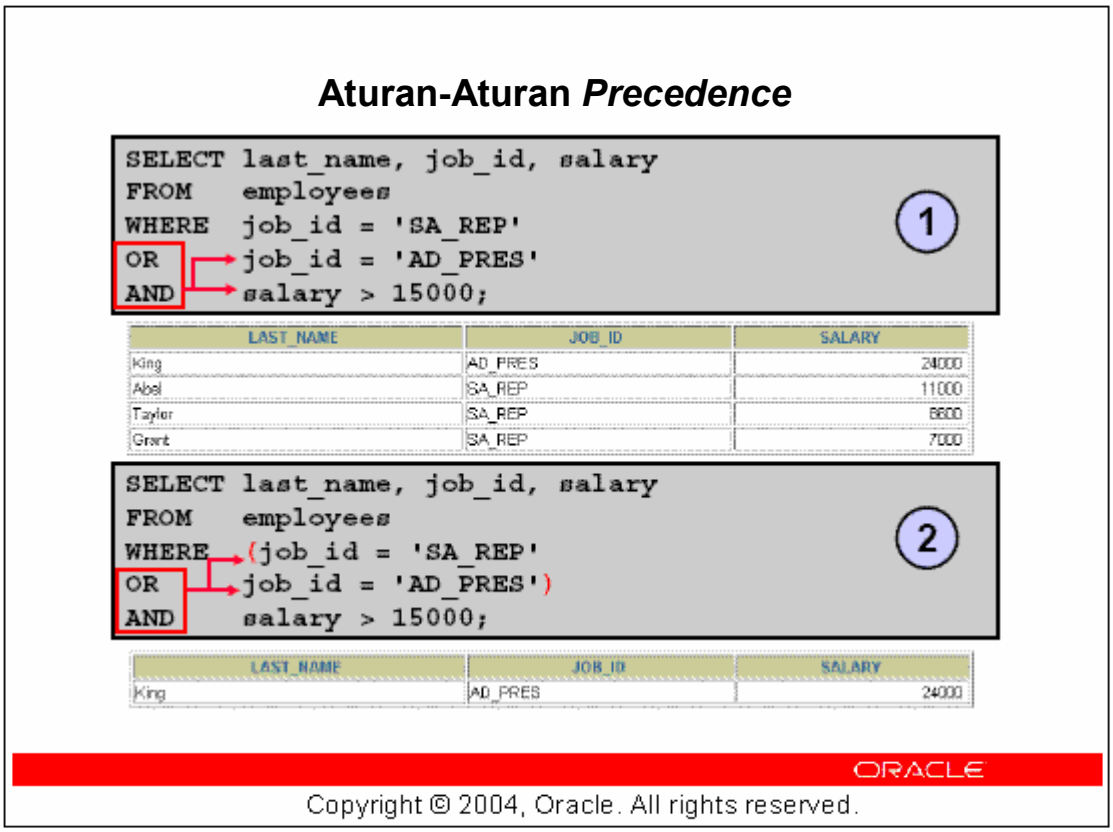
Operator	Maksud
1	Operator-operator aritmatika
2	Operator-operator penggabungan (*concatenation*)
3	Operator-operator pembanding
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Tidak sama dengan
7	Kondisi logika NOT
8	Kondisi logika AND
9	Kondisi logika OR

Anda bisa menggunakan tanda kurung untuk mengabaikan aturan-aturan *precedence*.

ORACLE

Aturan-aturan *Precedence*

Aturan-aturan *precedence* (aturan yang harus didahulukan) menentukan urutan ekspresi mana yang harus dievaluasi dan dihitung. Daftar tabel diatas adalah *default* urutan *precedence*. Anda bisa mengabaikan *default* urutan dengan menggunakan tanda kurung disekitar ekspresi yang ingin Anda hitung pertama kali.



1. Contoh dari *Precedence* pada Operator AND

Dalam contoh ini, ada dua kondisi :

- Kondisi pertama adalah job ID yaitu `AD_PRESS` *dan* penghasilan lebih besar dari $15,000.
- Kondisi kedua adalah job ID yaitu `SA_REP`.

Oleh karena itu, pernyataan `SELECT` dibaca sebagai berikut :

"Pilih suatu baris jika seorang pegawai adalah seorang presiden *dan* berpenghasilan lebih dari $15,000, *atau* jika pegawai adalah *sales representative*."

2. Contoh Menggunakan Tanda Kurung

Dalam contoh ini, ada dua kondisi :

- Kondisi pertama adalah job ID yaitu `AD_PRES` *atau* `SA_REP`.
- Kondisi kedua adalah penghasilan lebih besar dari $15,000.

Oleh karena itu, pernyataan `SELECT` dibaca sebagai berikut :

"Pilih suatu baris jika seorang pegawai adalah seorang presiden *atau* *sales representative*, *dan* jika pegawai berpenghasilan lebih dari $15,000."

Menggunakan Klausa ORDER BY

- **Mengambil penyortiran baris-baris dengan klausa ORDER BY :**
  - **ASC : Urutan *Ascending*, *default***
  - **DESC : Urutan *Descending***
- **Klausa ORDER BY terdapat diakhir pernyataan SELECT :**

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

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

...

20 rows selected.

ORACLE

Menggunakan klausa ORDER BY

Urutan dari baris-baris yang dihasilkan dari suatu hasil query adalah tidak tetap. Klausa ORDER BY bisa digunakan untuk menyortir baris-baris. Jika Anda menggunakan klausa ORDER BY, ORDER BY harus berada diakhir pernyataan SQL. Anda bisa menentukan suatu ekspresi , suatu alias atau posisi kolom sebagai kondisi pensortiran.

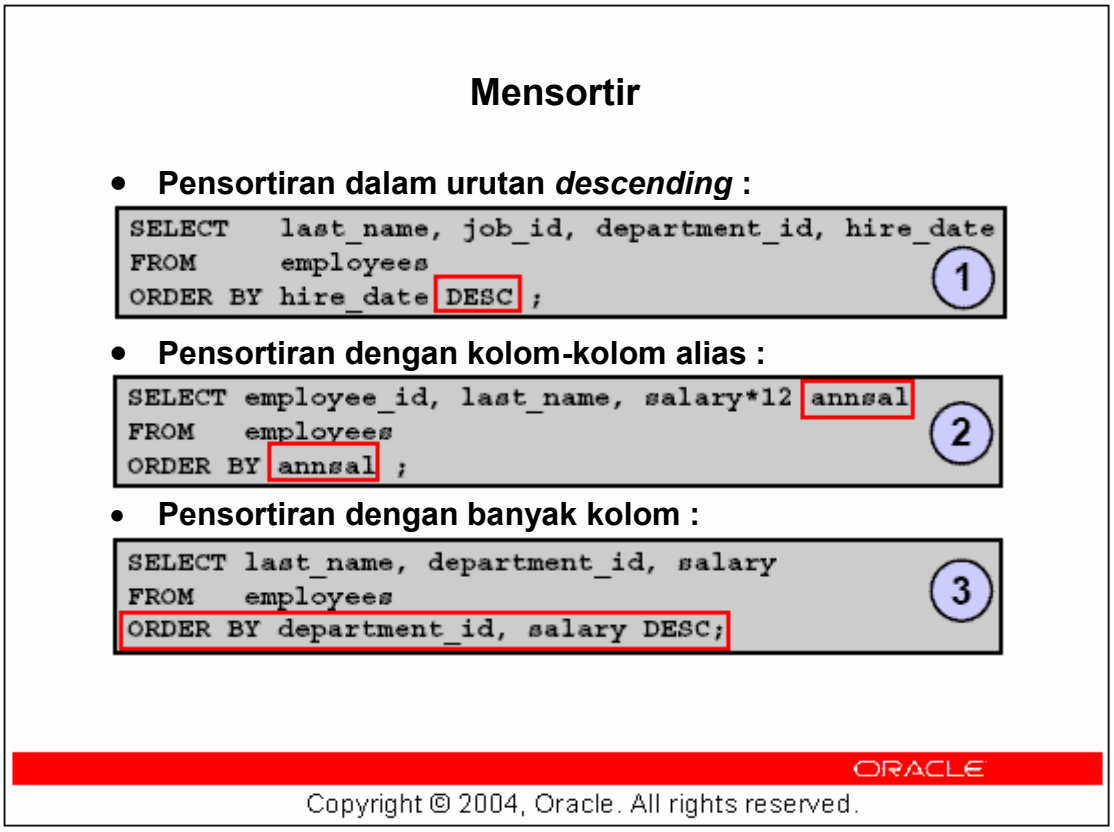
Sintak

```
SELECT  expr
FROM    table
[WHERE  condition(s)]
[ORDER BY {column,expr,numeric_position}[ASC|DESC]];
```

Dalam sintak :

ORDER BY	menentukan suatu urutan dimana baris-baris yang dihasilkan ditampilkan
ASC	mengurutkan baris-baris dalam urutan *ascending*, ASC adalah *default* pengurutan
DESC	mengurutkan baris-baris dalam urutan *descending*

Jika klausa ORDER BY tidak digunakan, urutan pensortiran tidak tetap, maka server Oracle mungkin tidak mengambil baris-baris dalam urutan yang sama pada query yang sama untuk kedua kalinya. Gunakan klausa ORDER BY untuk menampilkan baris-baris dalam urutan tertentu.



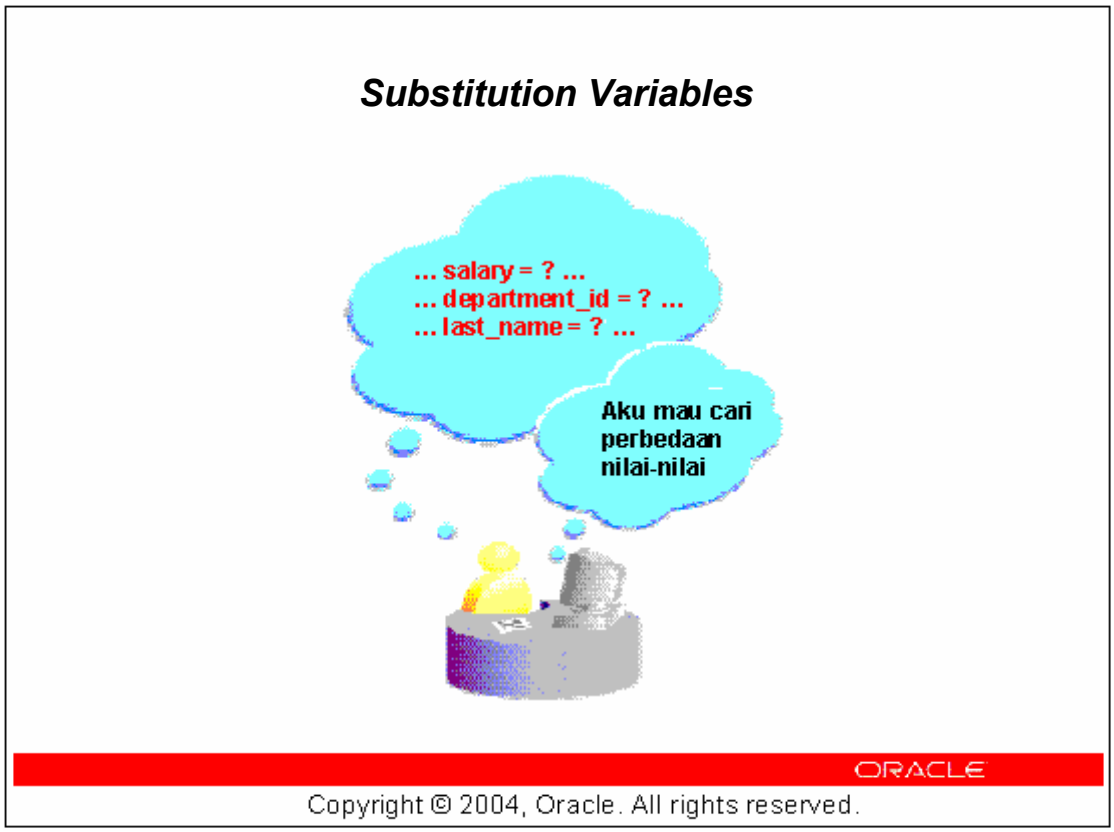
Default Pengurutan pada data

Default urutan pensortiran adalah *ascending* :

- Nilai-nilai *numeric* yang pertama ditampilkan adalah nilai terendah (contoh, 1 sampai 999).
- Nilai-nilai tanggal yang pertama ditampilkan adalah nilai pertama (contoh, 01-JAN-92 sebelum 01-JAN-95).
- Nilai-nilai karakter ditampilkan dalam urutan alfabetikal (contoh, A pertama dan Z terakhir).
- Nilai-nilai *null* ditampilkan terakhir pada urutan *ascending* dan diawal pada urutan *descending*.
- Anda bisa mensortir tiap kolom yang tidak ada dalam daftar `SELECT`.

Contoh

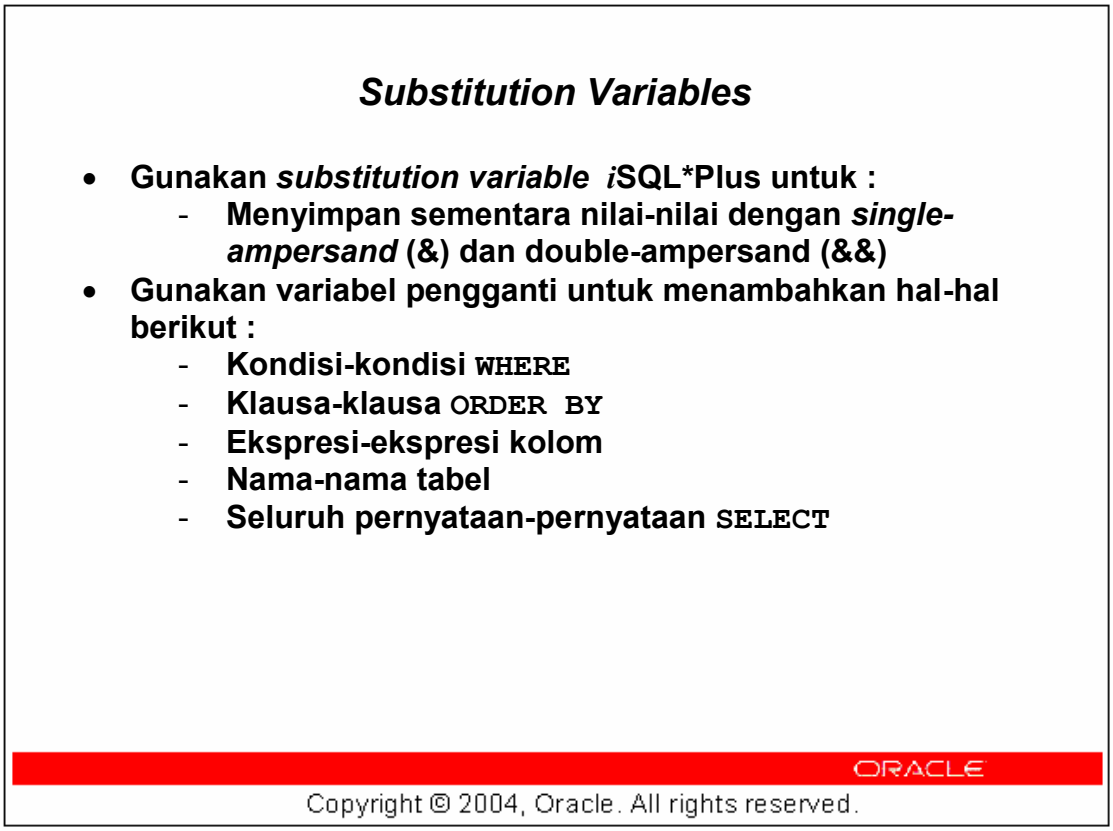
1. Untuk membalik urutan baris-baris mana yang ditampilkan , tentukan *keyword* (kata kunci) `DESC` setelah nama kolom pada klausa `ORDER BY`. Dalam slide dicontohkan hasil pensortiran per pegawai yang paling baru diterima.
2. Anda dapat menggunakan kolom alias pada klausa `ORDER BY`. Dalam slide dicontohkan pensortiran data per penghasilan tahunan.
3. Anda dapat mensortir hasil-hasil query dengan lebih dari satu kolom. Batas pensortiran adalah sejumlah kolom-kolom yang ada dalam tabel. Dalam klausa `ORDER BY`, tentukan kolom-kolom dan pisahkan nama-nama kolom menggunakan koma. Jika Anda menginginkan untuk membalik urutan suatu kolom, tentukan `DESC` setelah nama-nama kolom tersebut.



Substitution Variables

Contoh-contoh sejauh ini telah memberikan kode tetap (*hard-coded*). Dalam suatu aplikasi akhir, seorang user ingin laporan cepat dan laporan berjalan tanpa peringatan lebih jauh. Jarak antar data ingin ditentukan sebelumnya melalui klausa `WHERE` tetap dalam file *script* *i*SQL*Plus.

Dengan menggunakan *i*SQL*Plus, Anda bisa membuat laporan yang mengingatkan user untuk menyediakan nilai-nilai mereka sendiri untuk membatasi jarak antar data yang dihasilkan dengan menggunakan *substitution variables* (variabel-variabel pengganti). Anda bisa menyimpan *substitution variables* dalam *file command* atau dalam pernyataan SQL tunggal. Suatu variabel bisa disebut sebagai suatu penampung dimana nilai-nilai disimpan untuk sementara. Saat perintah dijalankan, nilai akan terganti.



Substitution Variable (lanjutan)

Dalam *i*SQL*Plus, Anda bisa menggunakan *substitution variable single-ampersand* (&) untuk menyimpan nilai-nilai sementara.

Sebelumnya Anda bisa menentukan variable-variabel dalam *i*SQL*Plus dengan menggunakan pernyataan `DEFINE`. `DEFINE` membuat dan menetapkan suatu nilai pada suatu variable.

Contoh Pembatasan Jarak Suatu Data

- Melaporkan bilangan hanya untuk bagian saat ini atau menentukan jarak data.
- Melaporkan suatu data yang relevan hanya untuk laporan yang diinginkan user.
- Menampilkan personel hanya dalam departemen

Efek-efek interaktif lainnya

Efek-efek interaktif tidak membatasi secara langsung interaksi user dengan klausa `WHERE`. Prinsip yang sama bisa digunakan untuk mencapai tujuan lain, seperti :

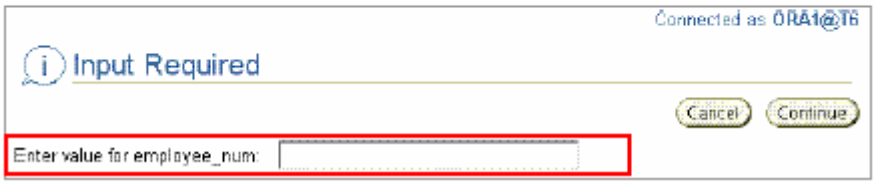
- Memperoleh nilai-nilai input dari suatu file daripada dari seseorang.
- Melewatkan nilai-nilai dari satu pernyataan SQL ke yang lainnya.

*i*SQL*Plus tidak mendukung pemeriksaan-pemeriksaan validasi (kecuali untuk tipe data) pada input user.

Menggunakan *Substitution Variable* &

Gunakan awalan variable dengan sebuah *ampersand* (&) untuk mengingatkan user atas suatu nilai :

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```



ORACLE

Single-Ampersand Substitution Variable

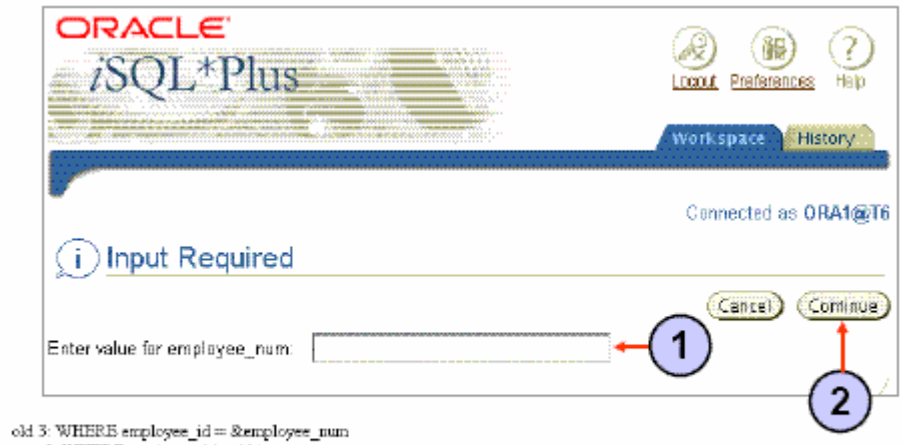
Saat menjalankan sebuah laporan, user sering menginginkan untuk membatasi suatu data yang dihasilkan secara dimanis. *i*SQL*Plus menyediakan fleksibilitas ini dengan variable-variable user. Gunakan sebuah *ampersand* (&) untuk mengidentifikasi setiap variable dalam pernyataan SQL Anda. Anda tidak perlu mendefinisikan nilai dari setiap variable.

Notasi	Penjelasan
&user_variabel	Menandakan suatu variable dalam pernyataan SQL; jika variable tidak ada, *i*SQL*Plus mengingatkan user akan suatu nilai (*i*SQL*Plus membuang suatu variable baru sekali setiap penggunaan.)

Contoh pada slide membuat suatu variable pengganti *i*SQL*Plus untuk nomor pegawai. Saat perintah dieksekusi, *i*SQL*Plus mengingatkan user akan satu nomor pegawai dan menampilkan nomor pegawai, nama belakang, penghasilan, dan nomor departemen untuk pegawai tersebut.

Dengan *single-ampersand,* user diingatkan setiap kali perintah dieksekusi, jika variable tidak ada.

Menggunakan *Substitution Variable* &




```
old 3: WHERE employee_id = &employee_num
new 3: WHERE employee_id = 101
```

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

S*ingle-Ampersand Substituion Variable* (lanjutan)

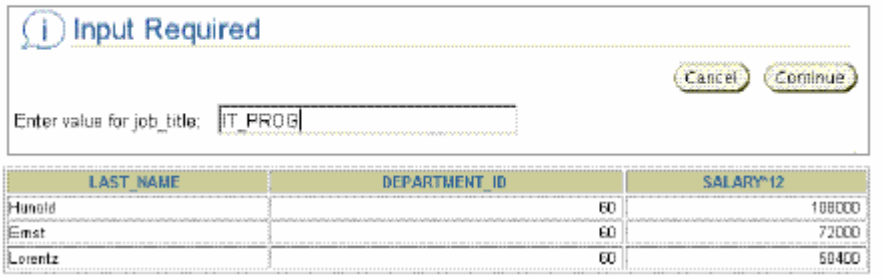
Saat *i*SQL*Plus mendeteksi bahwa pernyataan SQL mengandung sebuah *ampersand,* Anda diingatkan untuk memasukan nilai sebagai variable pengganti yang disebutkan pada pernyataan SQL.

Setelah Anda memasukkan nilai dan menekan tombol c*ontinue*, hasil ditampilkan dalam area output pada sesi *i*SQl*Plus Anda.

Nilai-Nilai Karakter dan Tanggal dengan *Substitution Variable*

Gunakan tanda petik tunggal untuk nilai-nilai tanggal dan karakter :

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

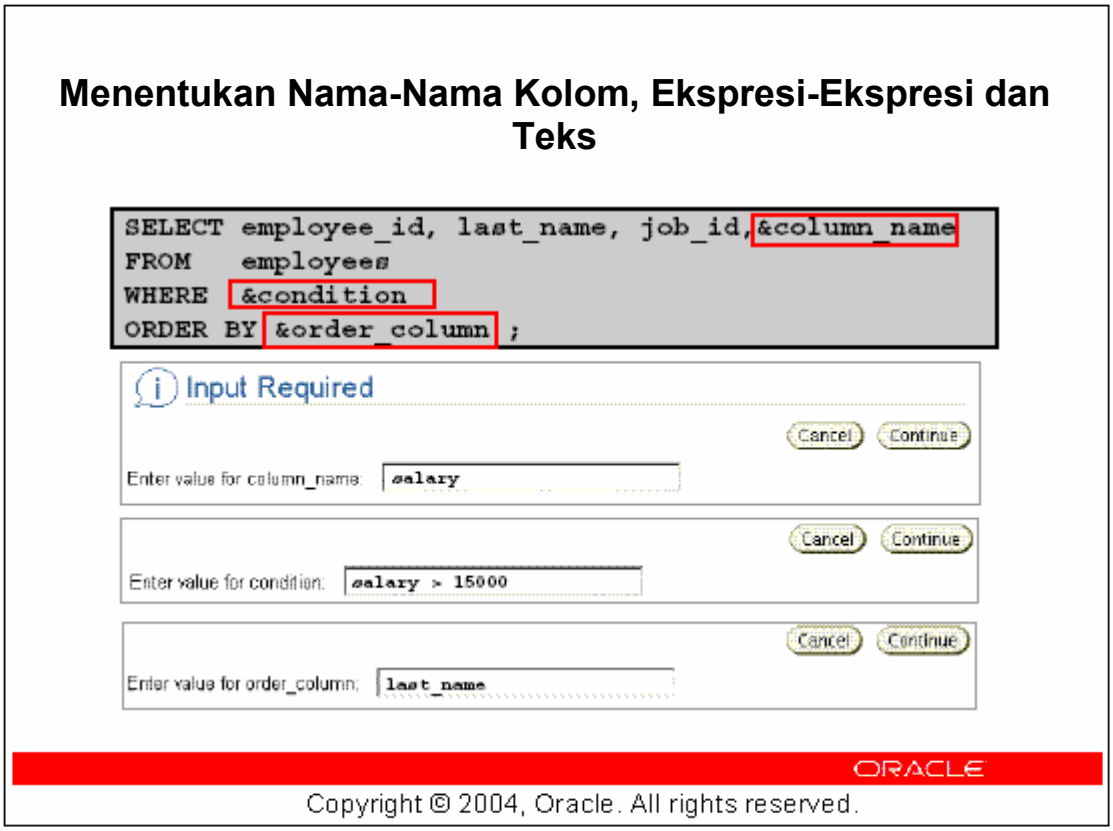
Menentukan Nilai-Nilai Karakter dan Tanggal dengan *Substitution Variable*

Dalam klausa `WHERE`, nilai-nilai tanggal dan karakter harus diapit dengan tanda petik tunggal. Aturan yang sama berlaku untuk *substitution variable*.

Apit variable dengan tanda petik tunggal dalam pernyataan SQL itu sendiri.

Slide menunjukkan query untuk menghasilkan nama-nama pegawai, nomor-nomor departemen, dan penghasilan per tahun untuk semua pegawai berdasarkan nilai nama job pada *substitution variable* *i*SQL*Plus.

Menentukan Nama-Nama Kolom, Ekspresi-Ekspresi dan Teks



Menentukan Nama-Nama Kolom, Ekspresi-Ekspresi dan Teks

Anda tidak hanya dapat menggunakan *substitution variables* pada klausa `WHERE` pada suatu pernyataan SQL, tapi variable-variable tersebut bisa juga digunakan untuk mengganti nama-nama kolom, ekspresi-ekspresi atau teks.

Contoh

Contoh pada slide menampilkan nomor pegawai , nama, nama job, dan setiap kolom lainnya yang ditentukan oleh user saat berjalan, dari table `EMPLOYEES`. Untuk setiap *substitution variable* dalam pernyataan `SELECT`, Anda diingatkan untuk memasukkan suatu nilai, lalu Anda menekan tombol *Continue* untuk memproses.

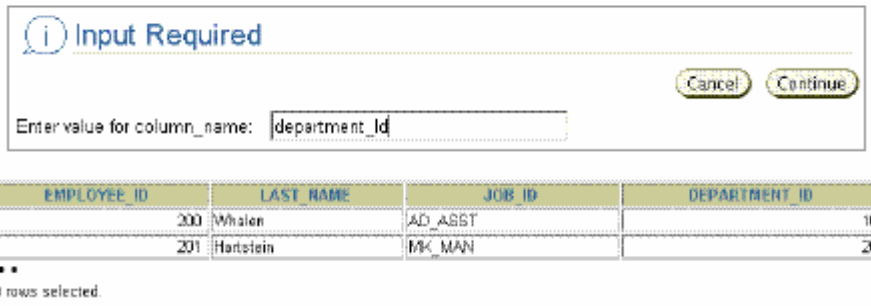
Jika Anda tidak memasukkan nilai pada *substitution variable*, Anda akan mendapati *error* saat Anda mengeksekusi perintah itu.

Catatan : Suatu *substitution variable* bisa digunakan dimanapun dalam pernyataan `SELECT`, kecuali sebagai masukkan kata pertama di *command prompt*.

Menggunakan *Substitution Variable* &&

Gunakan *double ampersand* (&&) jika Anda ingin untuk menggunakan kembali nilai variabel tanpa mengingatkan user setiap saat :

```
SELECT   employee_id, last_name, job_id, &&column_name
FROM     employees
ORDER BY &column_name ;
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

...

20 rows selected.

Double-Ampersand Subtitution Variable

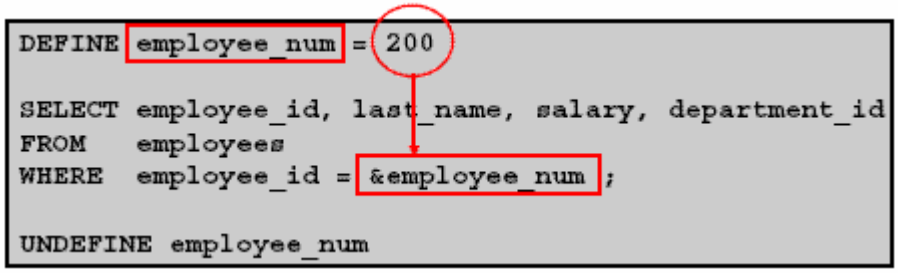
Anda bisa menggunakan *substitution variable double ampersand* (`&&`) jika Anda ingin menggunakan kembali nilai variable tanpa mengingatkan user setiap saat. User melihat peringatan untuk suatu nilai hanya sekali. Contoh pada slide, user diminta memberi nilai untuk variable `column-name` hanya sekali. Nilai yang disediakan oleh user (`department_id`) digunakan baik untuk menampilkan dan meminta data.

*i*SQL*Plus menyimpan nilai yang disediakan dengan menggunakan pernyataan `DEFINE`; `DEFINE` menggunakan `DEFINE` lagi manakala Anda merujuk suatu nama variabel. Setelah suatu user variabel ada pada tempatnya , Anda perlu menggunakan pernyataan `UNDEFINE` untuk menghapusnya seperti berikut :

```
UNDEFINE column_name
```

Menggunakan Perintah *i*SQL*Plus DEFINE

- **Gunakan perintah *i*SQL*Plus DEFINE untuk membuat dan memberikan suatu nilai ke suatu variabel.**
- **Gunakan perintah *i*SQL*Plus UNDIFINE untuk menghapus suatu variable.**



ORACLE

Menggunakan perintah *i*SQL*Plus DEFINE

Dalam contoh ditunjukkan pembuatan suatu variable pengganti *i*SQL *Plus untuk satu nomor pegawai dengan menggunakan pernyataan DEFINE. Saat dijalankan, ditampilkan nomor pegawai, nama, penghasilan, dan nomor departemen untuk karyawan itu.

Karena variabel dibuat menggunakan perintah *i*SQL*Plus DEFINE, user tidak diingatkan untuk memasukkan suatu nilai untuk nomor pegawai. Akan tetapi, nilai variable yang didefiniskan secara otomatis digantikan dalam pernyataan SELECT.

Substitution variable EMPLOYEE_NUM ada dalam suatu sesi sampai user tidak mendefinisikannya lagi atau keluar dari sesi *i*SQL*Plus.

Menggunakan Pernyataan VERIFY

Gunakan pernyataan VERIFY untuk memeriksa suatu tampilan pada *substitution variable*, baik sebelum dan sesudah *i*SQL*Plus menggantikan *substitution variables* dengan suatu nilai :

```
SET VERIFY ON
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num;
```

```
'employee_num' 200
```

```
old   3: WHERE  employee_id = &employee_num
new   3: WHERE  employee_id = 200
```

ORACLE

Menggunakan Perintah VERIFY

Untuk menegaskan perubahan-perubahan dalam pernyataan SQL, gunakan perintah *i*SQL*Plus VERIFY. Atur SET VERIFY ON pastikan *i*SQL*Plus untuk menampilkan teks dari suatu perintah sebelum dan sesudah dia mengganti *substitution variables* dengan nilai-nilai.

Contoh pada slide menampilkan nilai yang lama sebagai nilai baru pada kolom EMPLOYEE_ID.

Variabel-variabel Sistem *i*SQL*Plus

*i*SQL*Plus menggunakan beragam variabel-variabel yang mengontrol suatu lingkungan kerja. Salah satu dari variabel-variable itu adalah VERIFY. Untuk mendapatkan daftar lengkap semua variable-variabel sistem, Anda bisa mengeluarkan pernyataan SHOW ALL.

Ringkasan

Dalam pelajaran ini, Anda sudah mempelajari bagaimana untuk :

- **Menggunakan klausa `WHERE` untuk membatasi baris-baris pada output :**
  - **Menggunakan kondisi-kondisi pembanding**
  - **Menggunakan kondisi-kondisi operator `BETWEEN`, `IN`, `LIKE` dan `NULL`**
  - **Menerapkan operator-operator logika `AND`, `OR` dan `NOT`**
- **Menggunakan klausa `ORDER BY` untuk mensortir output baris-baris :**

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

- **Menggunakan *ampersand substitution* dalam *i*SQL*Plus untuk membatasi dan mensortir keluaran saat berjalan.**

ORACLE

Ringkasan

Dalam pelajaran ini, Anda sudah mempelajari tentang pembatasan dan pensortiran baris-baris yang dihasilkan oleh pernyataan `SELECT`. Anda juga sudah mempelajari bagaimana menerapkan beragam operator-operator dan kondisi-kondisi.

Dengan menggunakan *substitution variables* *i*SQL*Plus, Anda dapat menambah keleluasaan pada pernyataan SQL Anda. Anda dapat meng-query user-user saat berjalan dan memungkin mereka untuk menentukan kriteria.