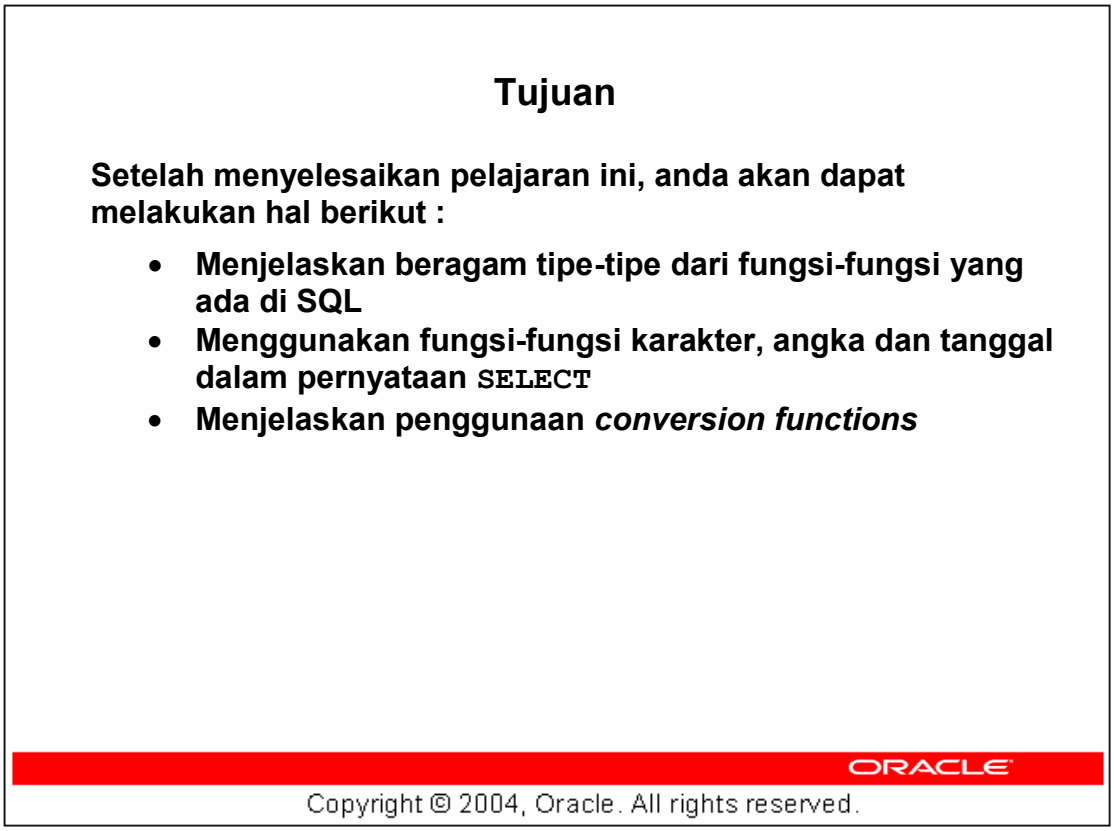
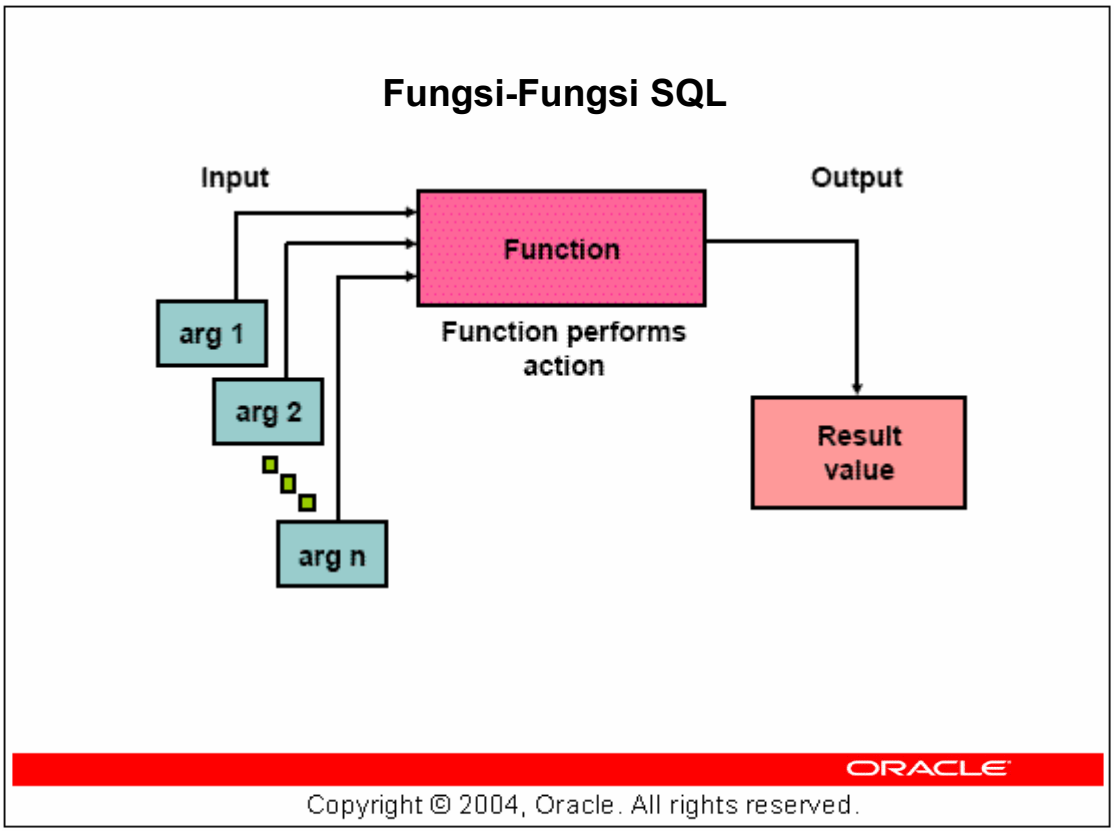
3

Menggunakan *Single-Row Function* untuk Meng-c*ustomize* Output



Tujuan

Fungsi-fungsi membuat blok query dasar menjadi lebih cepat, dan fungsi-fungsi tersebut digunakan untuk memanipulasi nilai-nilai data. Ini adalah bagian pertama dari dua pelajaran yang mengupas fungsi-fungsi. Bagian pertama ini berpusat pada fungsi-fungsi *single-row functions* karakter, angka dan tanggal, seperti adanya fungsi-fungsi tersebut yang digunakan untuk merubah data dari satu tipe ke tipe lainnya (sebagai contoh, konversi dari data karakter ke data angka).



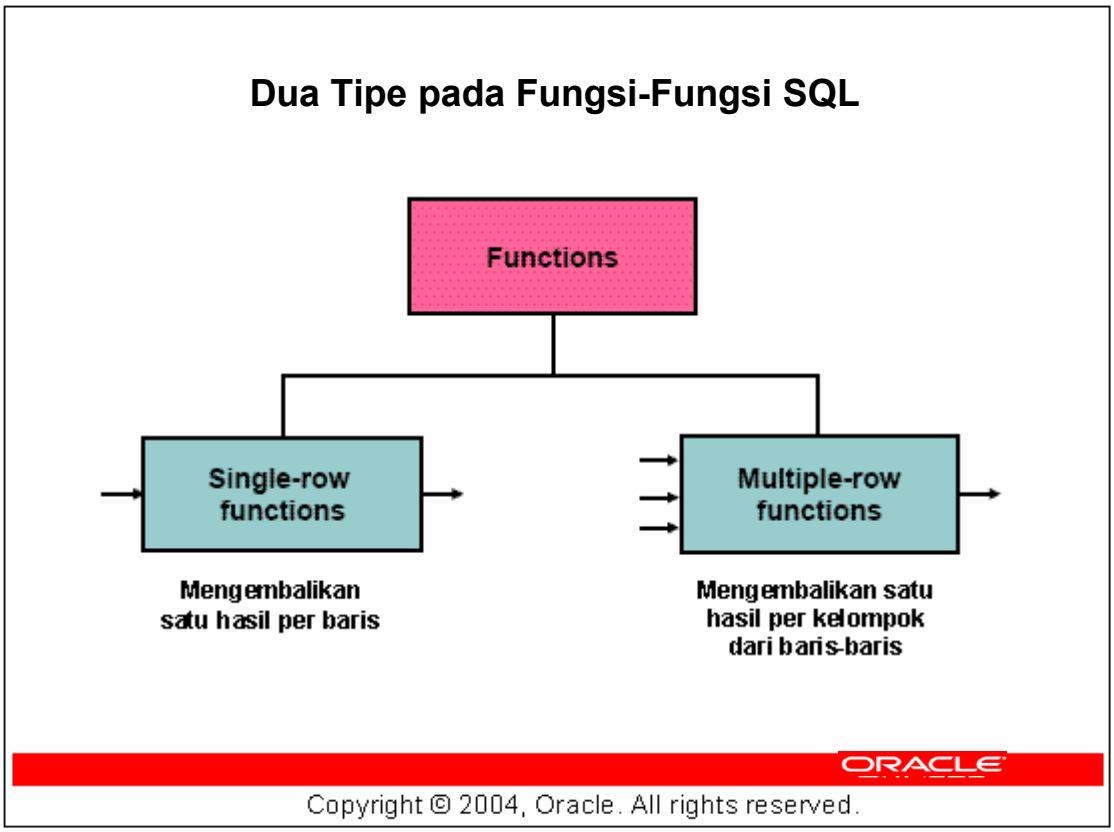
Fungsi-fungsi SQL

Fungsi-fungsi sangat mendukung fitur pada SQL. Fungsi-fungsi tersebut dapat digunakan untuk melakukan hal berikut:

- Melakukan perhitungan-perhitungan pada data
- Memodifikasi item-item data individual
- Memanipulasi keluaran dari kelompok baris-baris
- Format untuk tampilan tanggal dan angka-angka
- Konversi tipe data-tipe data kolom

Fungsi-fungsi SQL kadang-kadang mengambil argument-argumen dan selalu mengembalikan suatu nilai.

Catatan : Kebanyakan fungsi-fungsi yang diuraikan di dalam pelajaran ini dikhususkan untuk SQL versi ORACLE.



Fungsi-Fungsi SQL (lanjutan)

Ada dua tipe dari fungsi-fungsi :

- *Single-rows functions* (fungsi-fungsi baris tunggal)
- *Multiple-row functions* (fungsi-fungsi banyak baris)

Single-Row Functions

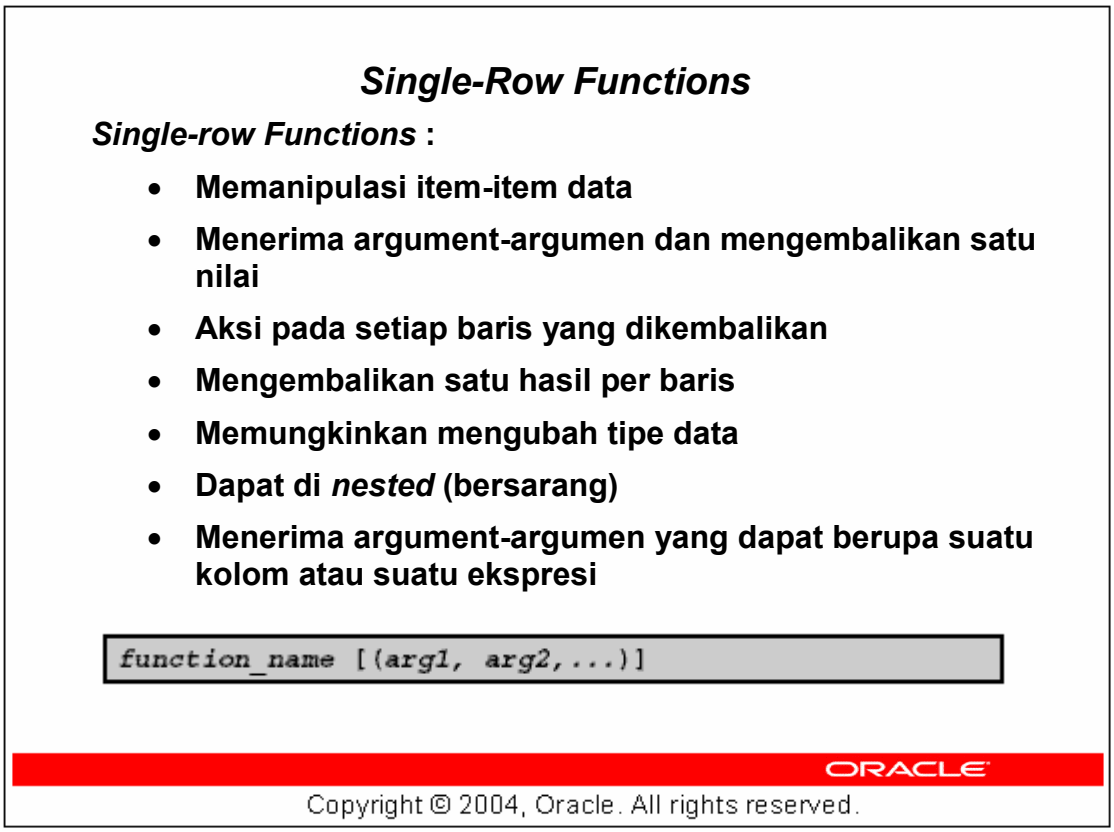
Fungsi-fungsi ini hanya digunakan pada baris-baris tunggal dan mengembalikan satu hasil per baris. Ada tipe-tipe berbeda dari *single-row functions*. Pelajaran ini mencakup hal-hal berikut :

- *Character* (karakter)
- Number (angka)
- *Date* (tanggal)
- *Conversion* (konversi)
- *General* (umum)

Multiple-row Functions

Fungsi-fungsi dapat memanipulasi kelompok dari baris-baris untuk memberi suatu hasil baris-baris per kelompok. Fungsi-fungsi ini dikenal juga sebagai *group functions* (dibahas pada pelajaran selanjutnya).

Catatan : Untuk informasi lebih rinci dan daftar lengkap dari fungsi-fungsi yang ada dan *syntax,* lihat *Oracle SQL Reference*.



Single Rows Functions

Single-row function digunakan untuk memanipulasi item-item data. *Single-row functions* menerima satu atau lebih argumen-argumen dan mengembalikan satu nilai untuk setiap baris yang dihasilkan oleh suatu query. Suatu argumen dapat berupa berikut ini :

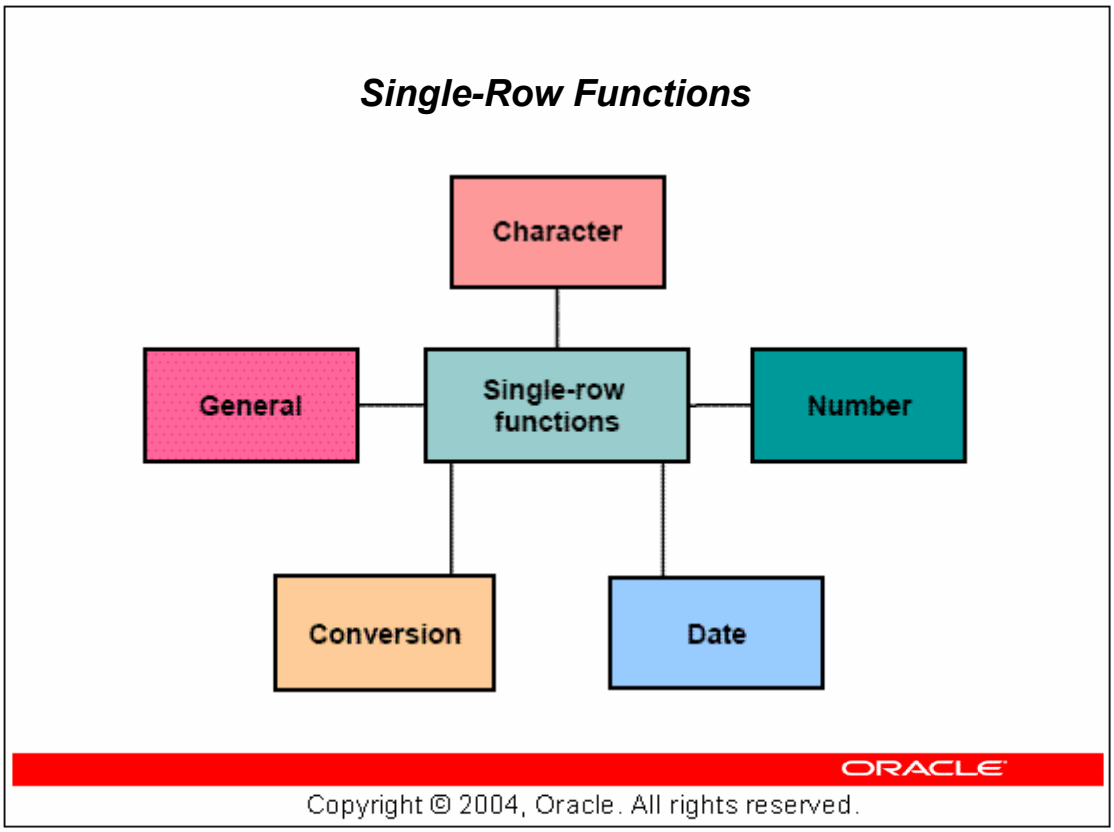
- *User-supplied constant* (Konstanta yang disediakan oleh user)
- Nilai variable
- Nama Kolom
- Ekspresi

Fitur-fitur dari *single-row functions* mencakup :

- Aksi pada setiap baris yang dikembalikan di dalam query
- Mengembalikan satu hasil per baris
- Memungkinkan pengembalian suatu nilai data dari suatu tipe berbeda daripada satu yang direferensikan.
- Mungkin menerima satu atau lebih argument-argumen
- Dapat digunakan didalam klausa-klausa `SELECT`, `WHERE`, dan `ORDER BY`; dapat disarangkan (*nested*)

Dalam sintak :

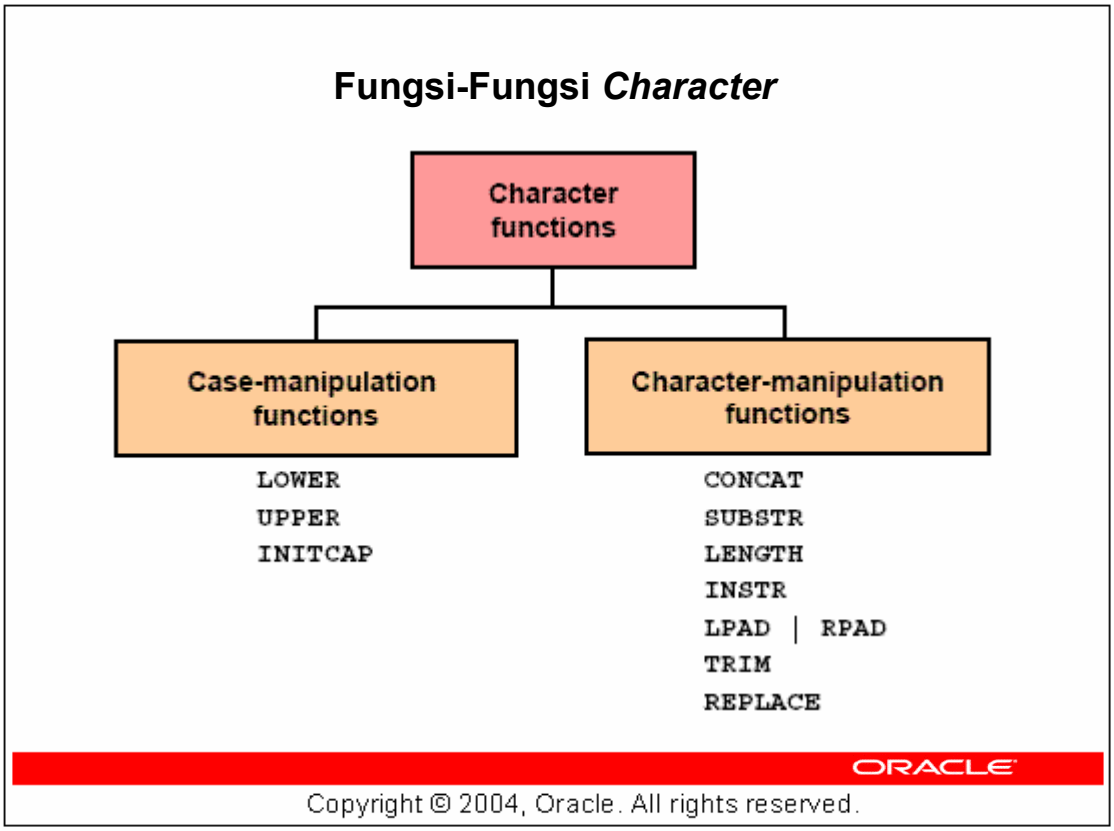
`function_name`	adalah nama dari fungsi
`arg1, arg2`	adalah setiap argumen yang digunakan oleh fungsi. Hal ini bisa jadi diwakili oleh suatu kolom atau ekspresi.



Single-Row Functions (lanjutan)

Pelajaran ini membahas *single-row functions* berikut :

- ***Character Functions*** : Menerima input karakter dan dapat mengembalikan baik nilai-nilai karakter ataupun angka.
- ***Number Functions*** : Menerima masukkan angka dan mengembalikan nilai-nilai angka
- ***Date Functions*** : Bekerja pada nilai-nilai dari tipe data `DATE` (semua *date functions* mengembalikan suatu nilai dari tipe data `DATE` kecuali fungsi `MONTHS_BETWEEN`, yang mengembalikan suatu angka.)
- ***Coversion Functions*** : Mengkonversi suatu nilai dari satu tipe data ke tipe data lainnya.
- ***General Functions*** **(fungsi-fungsi umum)** :
  - `NVL`
  - `NVL 2`
  - `NULLIF`
  - `COALESCE`
  - `CASE`
  - `DECODE`



Character Functions

Single-row character functions menerima data karakter sebagai masukan dan dapat mengembalikan baik nilai-nilai karakter maupun angka. *Character functions* dapat dibagi menjadi :

- *Case-manipulation functions*
- *Character-manipulation functions*

Function	Melakukan
`LOWER(column/expression)`	Mengkonversi nilai-nilai karakter huruf menjadi *lowercase* (huruf kecil)
`UPPER(column/expression)`	Mengkonversi nilai-nilai karakter huruf menjadi *uppercase* (huruf besar)
`INITCAP(column/expression)`	Mengkonversi nilai-nilai alpha karakter menjadi *uppercase* untuk huruf pertama dari tiap kata; semua huruf-huruf lain *lowercase*
`CONCAT(column1/expressio1, colum2/expression2)`	Menggabungkan nilai karakter pertama ke karakter kedua : sama dengan operator penggabungan (`||`)
`SUBSTR (column/expression, m[,n])`	Menghasilkan karakter-karakter tertentu dari nilai karakter dimulai pada posisi karakter ke-*m, ke- n* panjang karakter (jika *m* adalah negatif, dihitung mulai dari akhir nilai karakter. Jika *n* dihilangkan, menghasilkan semua karakter sampai akhir dari rangkaian.)

Catatan : Fungsi-fungsi yang didiskusikan dalam pelajaran ini adalah hanya beberapa dari fungsi-fungsi yang ada.

Fungsi-Fungsi *Character* (lanjutan)

Function	Melakukan
LENGTH (*column/expression*)	Mengembalikan jumlah karakter dalam ekspresi
INSTR (*column/expression, 'string', [,m], [n]*)	Mengembalikan posisi numerik dari suatu rangkaian penamaan. Secara *optional*, Anda dapat menyediakan suatu posisi ke-*m* untuk memulai pencarian, dan yang terjadi di-*n* dari suatu rangkaian. *m* dan *n* *default*-nya 1, artinya pencarian dimulai di awal suatu pencarian dan melaporkan kejadian yang pertama.
LPAD (*column /expression, n,'string'*)	Mengisi nilai karakter perataan kanan (*right-justified*) ke suatu lebar total *n* posisi karakter
RPAD (column /expression, n,'*string*')	Mengisi nilai karakter perataan kiri (*left-justified*) ke suatu lebar total *n* posisi karakter
TRIM (*leading/trailing/both, trim_character* FROM *trim_source*)	Memungkinkan Anda untuk memotong karakter-karakter bagian awal atau bagian akhir atau kedua-duanya dari suatu rangkaian karakter. Jika *trim_character* atau *trim_source* adalah suatu karakter literal, Anda harus mengapitnya dengan tanda petik tunggal. Ini adalah fitur yang ada di Oracle8*i* dan versi selanjutnya.
REPLACE (*text, search_string, replacement_string)*	Mencari suatu ekspresi teks untuk suatu rangkaian karakter dan, jika ditemukan, digantikan dengan rangkaian yang telah ditentukan.

Fungsi-Fungsi *Case-Manipulation*

Fungsi-fungsi ini merubah bentuk pada karakter-karakter *string* :

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

ORACLE

Case-Manipulation Functions

`LOWER`, `UPPER`, dan `INITCAP` adalah tiga *case-conversion functions* (fungsi-fungsi perubahan bentuk).

- **`LOWER`** : Konversi *mixed-case* (bentuk campuran) atau karakter-karakter *string* huruf besar ke huruf kecil
- **`UPPER`** : Konversi *mixed-case* atau karakter-karakter *string* huruf kecil ke huruf besar
- **`INITCAP`** : Konversi huruf pertama setiap kata ke huruf besar dan huruf-huruf selanjunya ke huruf kecil

```
SELECT 'the job id for '||UPPER(last_name)|| ' is '
      ||LOWER(job_id) AS "EMPLOYEE  DETAILS"
FROM employees;
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
. . .
The job id for HIGGINS is ac_mgr
The job id for GIETZ is a ac_account

20 rows selected.

Menggunakan Fungsi-Fungsi *Case-Manipulation*

Menampilkan nomor pegawai, nama, dan nomor departemen untuk pegawai bernama Higgins :

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

ORACLE

Menggunakan Fungsi-Fungsi *Case-Manipulation*

Contoh pada slide menampilkan nomor pegawai, nama, dan nomor departemen untuk pegawai bernama Higgins.

Klausa `WHERE` dari pernyataan SQL yang pertama menentukan karyawan bernama `higgins`. Karena semua data didalam tabel `EMPLOYEES` disimpan dalam bentuk yang sesuai, seseorang bernama `higgins` tidak menemukan kesesuaian didalam tabel, sehingga tidak ada baris-baris yang dipilih.

Klausa `WHERE` dari pernyataan SQL yang kedua menentukan bahwa nama seorang pegawai didalam tabel `EMPLOYEE` dibandingkan dengan `higgins`, dengan mengubah kolom `LAST_NAME` ke huruf kecil untuk tujuan pembandingan. Sejak kedua nama sekarang menjadi huruf kecil, ditemukan suatu kesesuaian sehingga suatu baris dipilih. Klausa `WHERE` dapat ditulis ulang dengan cara berikut untuk menghasilkan hasil yang sama :

```
. . . WHERE last_name= ' Higgins'
```

Nama pada output muncul seperti nama itu disimpan di dalam database. Untuk menampilkan nama dimana hanya huruf pertama dalam bentuk huruf besar, gunakan fungsi `UPPER` pada pernyataan `SELECT`.

```
SELECT employee_id, UPPER (last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

Fungsi-Fungsi *Character-Manipulation*

Fungsi-fungsi manipulasi karakter-karakter *string* :

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*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

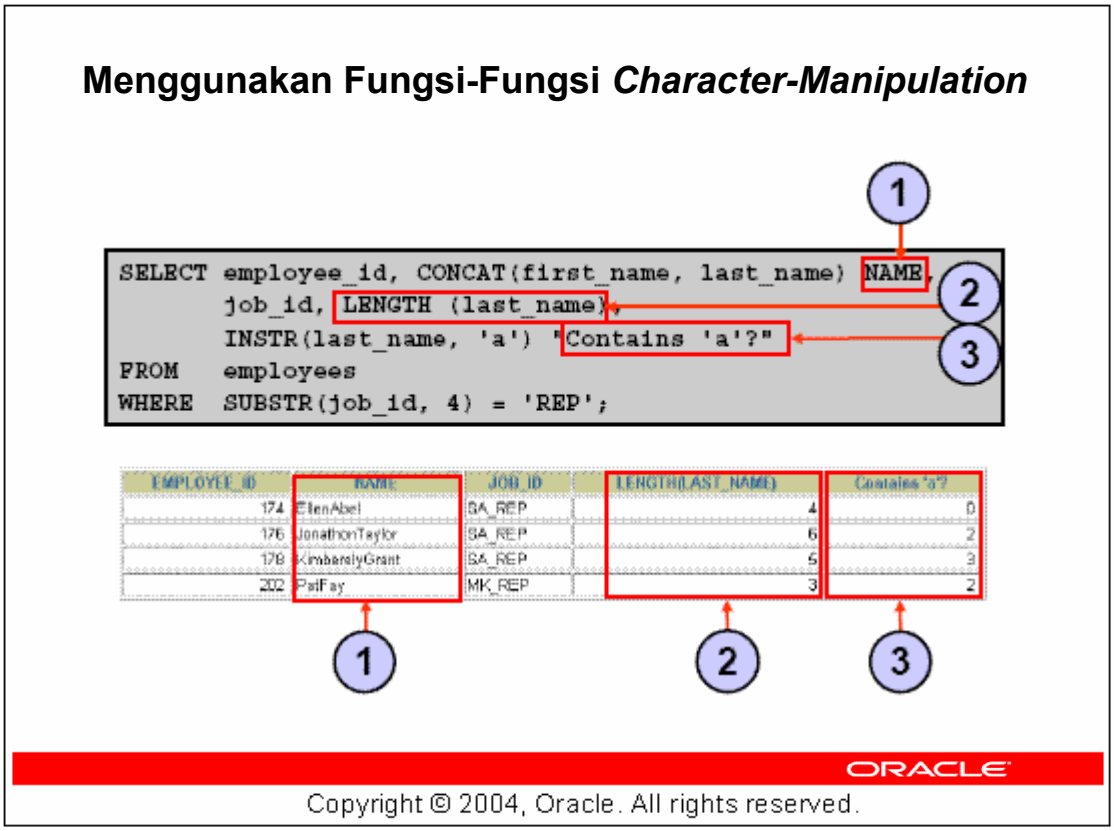
ORACLE

Fungsi-Fungsi *Character-Manipulation*

`CONCAT`, `SUBSTR`, `LENGTH`, `INSTR`, `LPAD`, `RPAD`, dan `TRIM` adalah suatu fungsi-fungsi manipulasi karakter yang dibahas dalam pelajaran ini.

- **`CONCAT` :** Menggabungkan bersama nilai-nilai (Anda dibatasi untuk menggunakan dua parameter pada CONCAT.)
- **`SUBSTR` :** Memotong suatu rangkaian dari panjang tertentu
- **`LENGTH` :** Menampilkan panjang dari suatu rangkaian sebagai suatu nilai numerik
- **`INSTR` :** Menemukan posisi numerik dari suatu karakter nama
- **`LPAD` :** Mengisi nilai karakter *right-justified* (perataan kanan)
- **`RPAD` :** Mengisi nilai karakter *left-justified* (perataan kiri)
- **`TRIM` :** Memotong karakter-karakter bagian awal atau bagian akhir (atau kedua-duanya) dari suatu rangkaian karakter (jika *`trim_character`* atau *`trim_source`* adalah suatu karakter literal, Anda harus mengapitnya didalam tanda petik tunggal.)

Catatan: Anda dapat menggunakan fungsi-fungsi seperti `UPPER` dan `LOWER` dengan *ampersand substitution*. Sebagai contoh, gunakan `UPPER ('& job_title')` sehingga user tidak bisa memasukkan suatu nama pekerjaan dalam suatu bentuk tertentu.



Menggunakan Fungsi-Fungsi *Character-Manipulation*

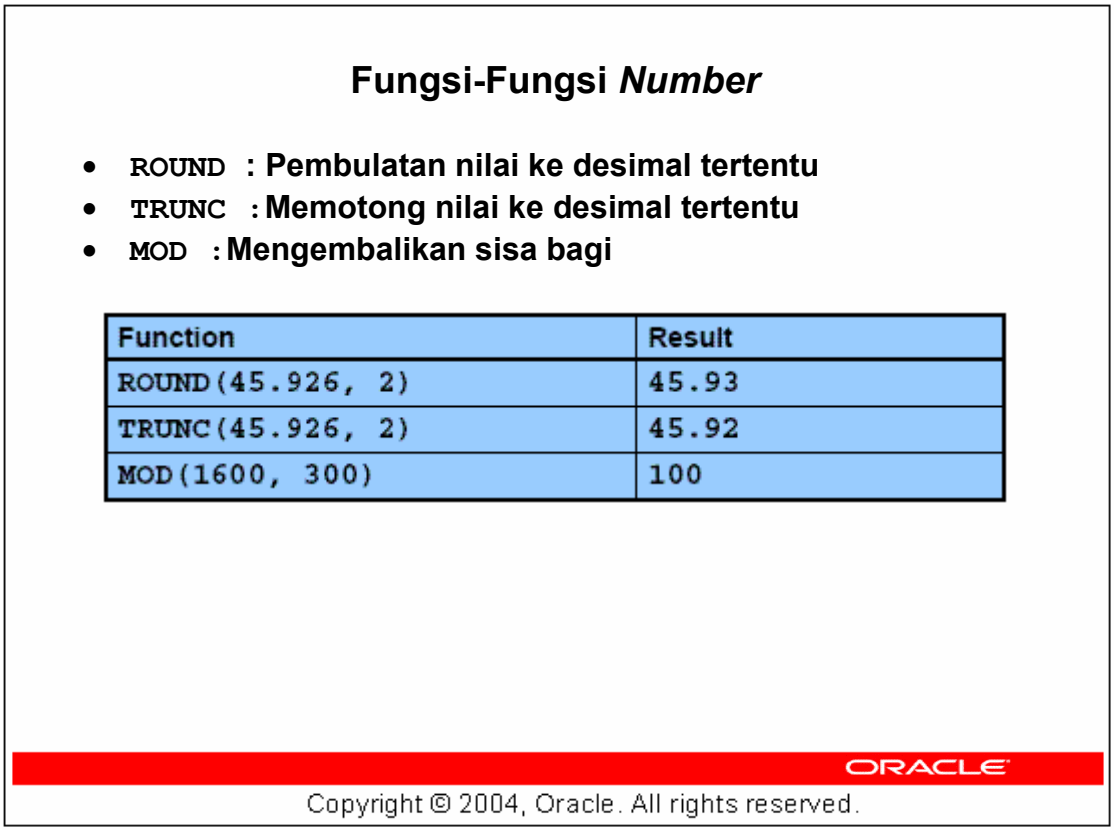
Contoh pada slide menampilkan nama depan pegawai dan nama belakang yang digabung bersama, panjang dari nama belakang pegawai, dan posisi angka dari huruf *a* dalam nama belakang pegawai untuk semua pegawai yang mempunyai rangkaian `REP` terkandung di dalam job ID dimulai dari posisi keempat dari job ID.

Contoh :

Ubah pernyataan SQL pada slide untuk menampilkan suatu data untuk pegawai-pegawai yang nama belakangnya diakhiri dengan huruf *n*.

```
SELECT employee_id, CONCAT (first_name, last_name) NAME,
LENGTH (last_name), INSTR (last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR (last_name, -1, 1) = 'n';
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartsein	9	2



Fungsi-Fungsi *Number*

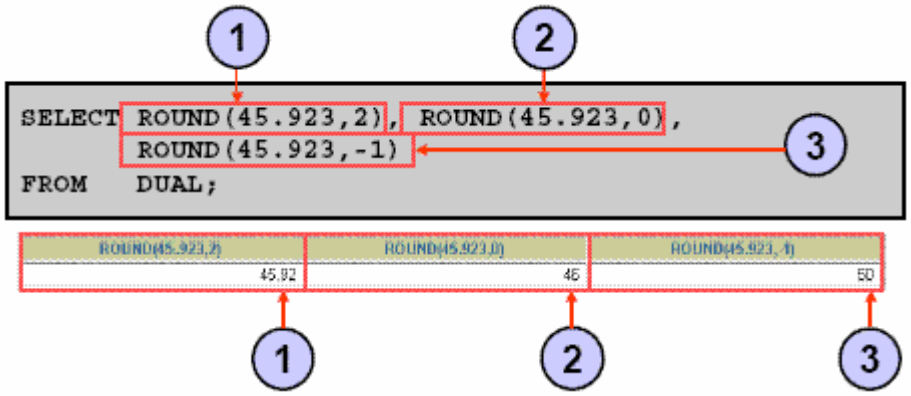
Fungsi-fungsi *Number* menerima input numerik dan mengembalikan nilai-nilai numerik. Pada bagian ini dibahas beberapa fungsi-fungsi *number*.

Function	Purpose
ROUND (*column* | *expression*, *n*)	Membulatkan kolom, ekspresi, atau nilai posisi ke-*n* desimal atau, jika *n* dihilangkan, tidak ada posisi desimal (Jika *n* adalah negatif, angka-angka di kiri dari desimal dibulatkan.)
TRUNC (*column* | *expression*, *n*)	Memotong kolom, ekspresi, atau nilai posisi ke-*n* desimal atau, jika *n* dihilangkan, *n default*-nya nol
MOD (*m*, *n*)	Mengembalikan sisa dari *m* yang dibagi oleh *n*

Catatan : Daftar ini hanya berisi beberapa dari *number function* yang tersedia .
Untuk informasi lebih lanjut, lihat "Number Functions" di *Oracle SQL Reference*.

.

Menggunakan Fungsi ROUND



DUAL adalah *dummy table* (tabel contoh) yang dapat anda gunakan untuk melihat hasil-hasil dari fungsi-fungsi dan perhitungan-perhitungan.

ORACLE

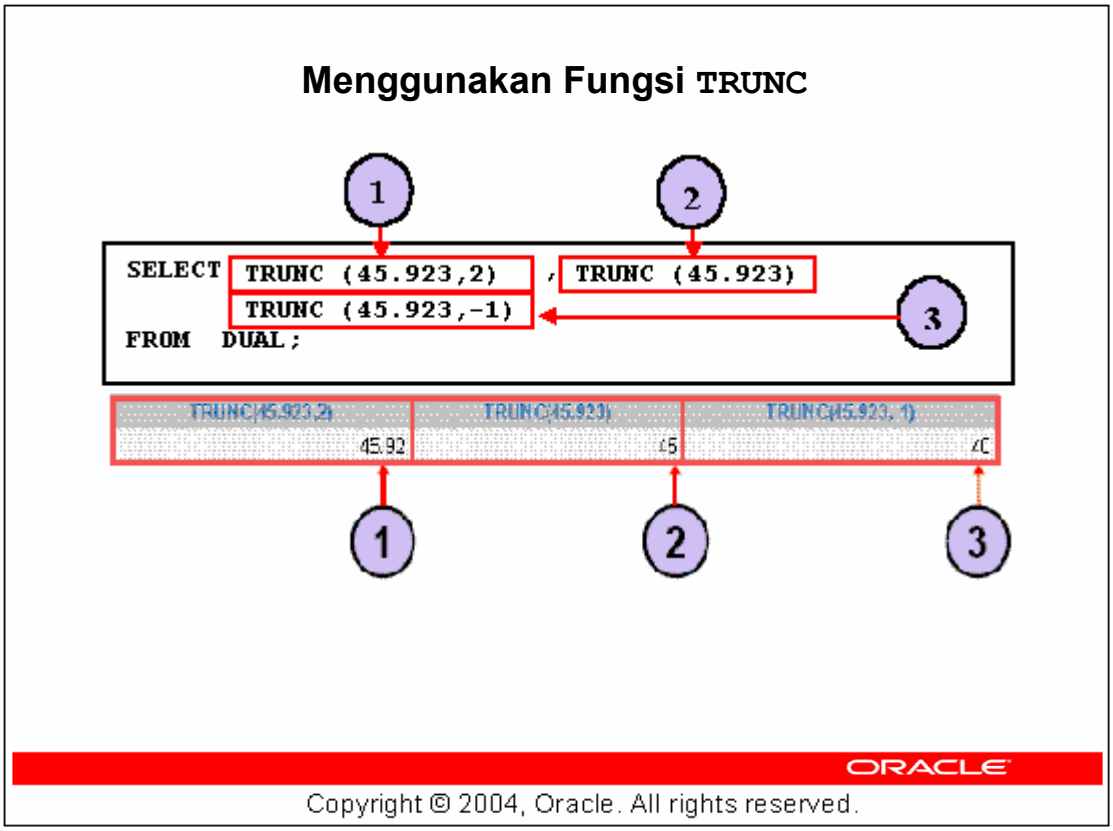
Fungsi ROUND

Fungsi ROUND membulatkan kolom, ekspresi, atau nilai posisi ke-*n* desimal. Jika argumen kedua adalah 0 atau hilang, suatu nilai dibulatkan ke posisi nol desimal. Jika argumen kedua adalah 2, nilai dibulatkan ke posisi dua decimal . Sebaliknya, jika argumen kedua adalah -2, nilai dibulatkan ke posisi dua desimal ke kiri (pembulatan ke unit yang paling dekat dengan 10).

Fungsi ROUND juga dapat digunakan dengan *date function* (fungsi-fungsi tanggal). Selanjutnya Anda akan melihat contoh-contoh dalam pelajaran ini.

DUAL Table

DUAL table dimiliki oleh user SYS dan dapat diakses oleh semua user. DUAL table berisi satu kolom, DUMMY, dan satu baris dengan nilai X. DUAL table bermanfaat ketika Anda ingin mengembalikan suatu nilai sekali saja (sebagai contoh, nilai dari suatu konstanta, *pseudocolumn* (kolom maya), atau ekspresi yang bukan berasal dari suatu tabel dengan data user). DUAL table secara umum digunakan untuk melengkapi sintak klausa SELECT, sebab baik klausa SELECT dan klausa FROM *mandatory* (bersifat perintah) , dan beberapa perhitungan-perhitungan tidak perlu memilih dari tabel-tebel aktual.



Fungsi TRUNC

Fungsi TRUNC memotong kolom, ekspresi, atau nilai posisi ke-*n* desimal.

Fungsi TRUNC bekerja dengan argument-argumen yang sama untuk fungsi ROUND. Jika argumen kedua adalah 0 atau hilang, nilai dipotong ke posisi nol desimal. Jika argumen kedua adalah 2, nilai dipotong ke posisi dua desimal. Sebaliknya, jika argumen kedua adalah -2, nilai dipotong ke posisi dua desimal ke kiri. Jika argumen kedua adalah – 1, nilai dipotong ke posisi satu desimal ke kiri.

Seperti fungsi ROUND, fungsi TRUNC dapat digunakan dengan date function.

Menggunakan Fungsi MOD

Untuk semua pegawai dengan nama pekerjaan Sales Representative, hitung sisa dari penghasilan setelah penghasilan tersebut dibagi dengan 5,000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

ORACLE

Fungsi MOD

`MOD` *function* akan menemukan suatu sisa dari argumen pertama dibagi dengan argumen kedua. Contoh pada slide menghitung sisa dari penghasilan setelah dibagi dengan 5,000 untuk semua pegawai yang job ID-nya adalah `SA_REP`.

Catatan : `MOD` *function* sering digunakan untuk menentukan bahwa suatu nilai adalah ganjil atau genap.

Bekerja pada *Dates*

- **Database Oracle menyimpan tanggal-tanggal dalam suatu format angka sendiri : abad, tahun, hari, jam, menit dan detik.**
- ***Default* tanggal ditampilkan dengan format DD-MM-RR.**
  - **Memungkinkan anda untuk menyimpan tanggal-tanggal abad ke-21 ke dalam abad ke-20 dengan menentukan hanya dua digit terakhir suatu tahun**
  - **Memungkinkan anda untuk menyimpan tanggal-tanggal abad ke-20 ke dalam abad ke-21 dengan cara yang sama**

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-88';
```

LAST_NAME	HIRE_DATE
King	17-JUN-87
Whalen	17-SEP-87

ORACLE

Format Tanggal Oracle

Database Oracle menyimpan tanggal-tanggal dalam suatu format angka sendiri, yang menunjukkan abad, tahun, bulan, hari, jam, menit, dan detik.

Tampilan *default* dan format inputan untuk setiap tanggal adalah DD-MON-RR. Tanggal Oracle berlaku antara *January 1,4712 B.C.* (1 Januari 4712 S.M.) dan *December 31, 9999 A.D.* (31 Desember 9999 M.).

Contoh pada slide, suatu output kolom `HIRE_DATE` ditampilkan format *default* DD-MON-RR. Meskipun begitu, tanggal-tanggal tidak disimpan dalam database dengan format ini. Semua bagian-bagian dari tanggal dan waktu disimpan. Maka, walaupun suatu `HIRE_DATE` seperti 17-JUN-87 ditampilkan sebagai hari, bulan, dan tahun, ada juga informasi *waktu* dan *abad* disatukan dengan date tersebut. Data yang lengkap bisa jadi *June 17, 1987, 5:10:43 p.m.* (17 Juni 1987, 5:10:43 petang).

Data ini disimpan secara internal sebagai berikut:

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	87	06	17	17	10	43

Abad-abad dan Tahun 2000

Saat suatu record pada suatu kolom tanggal disisipkan ke dalam suatu tabel, suatu informasi *abad* diambil dari fungsi `SYSDATE`. Meskipun demikian, ketika suatu kolom tanggal ditampilkan di layar, bagian abad tidak ditampilkan (secara *default*).

Secara internal tipe data `DATE` selalu menyimpan informasi tahun sebagai empat digit angka : dua digit untuk abad dan dua digit untuk tahun. Sebagai contoh, database Oracle menyimpan tahun dengan cara 1987 atau 2004, dan tidak seperti 87 atau 04.

Bekerja pada *Dates*

SYSDATE adalah suatu fungsi yang mengembalikan :

- ***Date***
- ***Time***

Fungsi SYSDATE

SYSDATE adalah fungsi tanggal yang mengembalikan tanggal dan waktu server database saat ini. Anda dapat menggunakan SYSDATE seperti Anda ingin menggunakan setiap nama kolom lain. Sebagai contoh, Anda dapat menampilkan tanggal saat ini dengan memilih SYSDATE dari suatu tabel. Biasanya tabel untuk memilih SYSDATE dari suatu tabel contoh disebut tabel DUAL.

Contoh

Menampilkan tanggal saat ini menggunakan tabel DUAL.

```
SELECT SYSDATE
FROM   DUAL;
```

SYSDATE
28-SEP-01

Aritmatika pada *Dates*

- **Menambah atau mengurangkan suatu angka ke atau dari suatu tanggal yang menghasilkan nilai *date*.**
- **Mengurangkan dua tanggal untuk mencari angka dari hari-hari antara tanggal-tanggal tersebut.**
- **Menambah jam ke suatu tanggal dengan membagi suatu angka dari jam dengan 24.**

Aritmatika *Dates*

Karena database menyimpan tanggal sebagai angka-angka, Anda dapat melakukan perhitungan menggunakan operator aritmatika seperti penambahan dan pengurangan. Anda dapat menambahkan dan mengurangkan konstanta angka begitu pula pada tanggal.

Anda dapat melakukan operasi-operasi berikut :

Operasi	Hasil	Keterangan
date + number	Date	Menambahkan suatu angka dari hari ke suatu tanggal
date – number	Date	Mengurangikan suatu angka dari hari dari suatu tanggal
date – date	Number of days	Mengurangkan suatu tanggal dari tanggal yang lain
date + number/24	Date	Menambahkan suatu angka dari jam ke suatu tanggal

Menggunakan Operator Aritmatika *Dates*

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102539
De Haan	453.245395

ORACLE

Aritmatika Dates (lanjutan)

Contoh pada slide menampilkan nama belakang dan jumlah minggu kerja untuk semua pegawai di departement 90. Contoh tersebut mengurangi tanggal dimana pegawai mulai bekerja dari tanggal saat ini (`SYSDATE`) dan hasilnya dibagi 7 untuk menghitung jumlah minggu dimana pegawai telah bekerja.

Catatan : `SYSDATE` adalah fungsi SQL yang mengembalikan tanggal dan waktu saat ini. Hasil Anda mungkin berbeda dengan contoh diatas

Jika tanggal saat ini dikurangkan dengan suatu tanggal sebelumnya, selisihnya adalah angka negatif.

Fungsi-Fungsi *Date*

Function	Hasil
MONTHS_BETWEEN	Angka suatu bulan diantara dua tanggal
ADD_MONTHS	Menambah bulan-bulan kalender ke tanggal
NEXT_DAY	Hari berikutnya dari suatu tanggal tertentu
LAST_DAY	Hari terakhir dari suatu bulan
ROUND	Pembulatan hari
TRUNC	Pemotongan hari

ORACLE

Dates Functions

Date function bekerja pada tanggal –tanggal Oracle. Semua *date function* mengembalikan suatu nilai dengan tipe data `DATE` kecuali `MONTHS_BETWEEN`, yang mengembalikan nilai numerik.

- **`MONTHS_BETWEEN`** (***`date1, date2`***) : Mencari jumlah bulan diantara *`date1`* dan *`date2`*. Hasilnya bisa jadi positif atau negatif. Jika *`date1`* lebih awal daripada *`date2`*, hasilnya adalah positif ; jika *`date1`* lebih awal daripada *`date2`*, hasilnya adalah negatif . Sebagian dari hasil bukan bilangan bulat (*noninteger*) menunjukkan suatu bagian dari bulan.
- **`ADD_MONTHS`** (***`date, n`***) : Menambahkan *`n`* jumlah suatu bulan kalender ke *`date`*. Nilai dari *`n`* harus bilangan bulat (*integer*) dan bisa negatif.
- **`NEXT_DATE`** (*`date, 'char'`*) : (*`'char'`*) setelah *`date`* menemukan suatu tanggal dari suatu hari tertentu pada suatu minggu. Nilai dari *`char`* bisa angka yang mewakili suatu hari atau suatu karakter *string*.
- **`LAST_DAY`** (*`date`*) : Mencari hari terakhir dari suatu tanggal dalam suatu bulan yang berisi *`date`*.
- **`ROUND`** (***`date[,'fmt']`***) : Mengembalikan pembulatan *date* ke suatu unit yang ditentukan oleh model format *`fmt`*. Jika model format *`fmt`* dihilangkan, *`date`* dibulatkan ke hari terdekat.
- **`TRUNC`** (***`date`*[,*`'fmt'`*]**) : Mengembalikan *`date`* dengan bagian suatu waktu dari suatu hari yang dipotong ke unit yang ditentukan oleh model format *`fmt`*. Jika model format *`fmt`* dihilangkan , *`date`* dipotong ke hari terdekat.

Daftar ini adalah sebagian dari *date function* yang ada. Model-model format selanjut dibahas dalam pelajaran ini. Contoh-contoh dari model-model format adalah bulan dan tahun.

Menggunakan Fungsi-Fungsi *Date*

Fungsi	Hasil
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

ORACLE

Fungsi-Fungsi *Date* (lanjutan)

Sebagai contoh, tampilkan nomor pegawai, tanggal masuk, jumlah bulan bekerja, tanggal *review* 6 bulan, Jumat pertama setelah tanggal masuk, dan hari terakhir dari bulan masuk kerja untuk semua pegawai yang telah bekerja kurang dari 70 bulan

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date,'FRIDAY'), LAST_DAY (hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) <70;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(	LAST_DAY(
107	07-FEB-99	31.6982407	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	22.4079182	16-MAY-00	19-NOV-99	30-NOV-99
149	29-JAN-00	19.9885633	29-JUL-00	04-FEB-00	31-JAN-00
178	24-MAY-99	28.1498536	24-NOV-99	28-MAY-99	31-MAY99

Menggunakan Fungsi-Fungsi *Date*

Anggap SYSDATE = '25-JUL-03' :

Fungsi	Hasil
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE ,'YEAR')	01-JAN-04
TRUNC(SYSDATE ,'MONTH')	01-JUL-03
TRUNC(SYSDATE ,'YEAR')	01-JAN-03

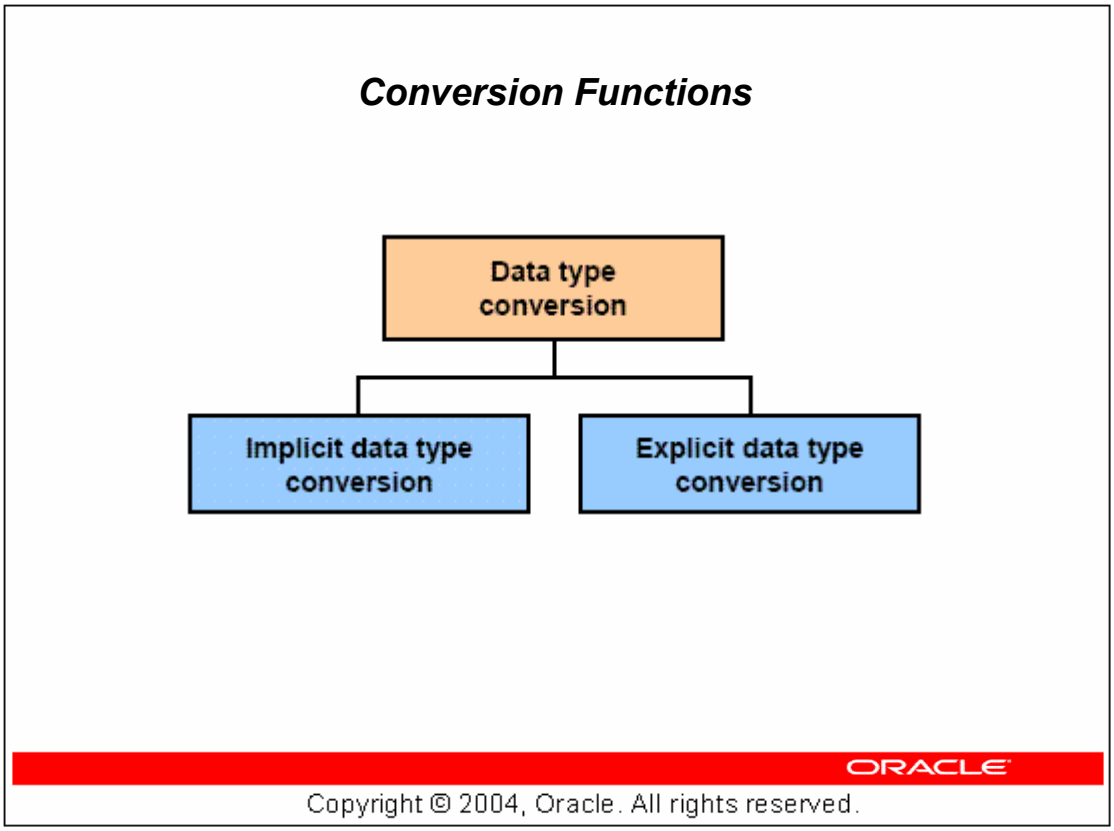
Menggunakan Fungsi-Fungsi *Date* (lanjutan)

Fungsi-fungsi ROUND dan TRUNC dapat digunakan untuk nilai-nilai angka dan tanggal. Ketika digunakan pada tanggal-tanggal, fungsi-fungsi ini membulatkan atau memotong ke model format tertentu. Oleh karena itu, Anda dapat membulatkan tanggal-tanggal ke tahun atau bulan terdekat.

Contoh :
Bandingkan tanggal-tanggal masuk untuk semua pegawai yang dimulai tahun 1997. Tampilkan nomor pegawai, tanggal masuk dan bulan awal menggunakan fungsi-fungsi ROUND dan TRUNC.

```
SELECT   employee_id, hire_date,
         ROUND(hire_date,'MONTH'), TRUNC(hire_date, 'MONTH')
FROM     employees
WHERE    hire_date LIKE '% 97';
```

EMPLOYEE_ID	HIRE_DATE	ROUND(HIR	TRUNC(HIR
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-97



Conversion Functions

Disamping tipe data-tipe data Oracle, kolom-kolom dari tabel-tabel di dalam Oracle dapat didefinisikan menggunakan tipe data-tipe data ANSI, DB2, dan SQL/DS. Meskipun demikian, server Oracle secara internal merubah tipe data-tipe data yang sama dengan tipe data-tipe data Oracle.

Dalam beberapa kasus, server Oracle menggunakan data dari suatu tipe data dimana server Oracle memperkirakan data dari suatu tipe data yang lain. Ketika ini terjadi, server Oracle dapat secara otomatis mengubah suatu data ke tipe data yang diperkirakan. Perubahan tipe data ini bisa jadi dilakukan secara *implisit* (*implicitly*) oleh server Oracle atau secara *eksplisit* (*explicitly*) oleh user.

Konversi-konversi tipe data implisit bekerja menurut aturan-aturan yang dijelaskan dalam slide berikutnya.

Konversi-konversi tipe data eksplisit dilakukan dengan menggunakan *conversion functions*. *Conversion functions* merubah suatu nilai dari suatu tipe data ke tipe data lain. Umumnya, bentuk dari nama-nama fungsi mengikuti konvensi (kesepakatan) `data type TO data type`. Tipe data yang pertama adalah tipe data input; tipe data yang kedua adalah output.

Catatan : Meskipun disediakan konversi tipe data implisit, Anda diijinkan melakukan konversi tipe data eksplisit untuk memastikan kebenaran pernyataan SQL Anda.

Konversi Tipe Data Implisit

Untuk penugasan-penugasan, server Oracle dapat secara otomatis mengkonversi berikut ini :

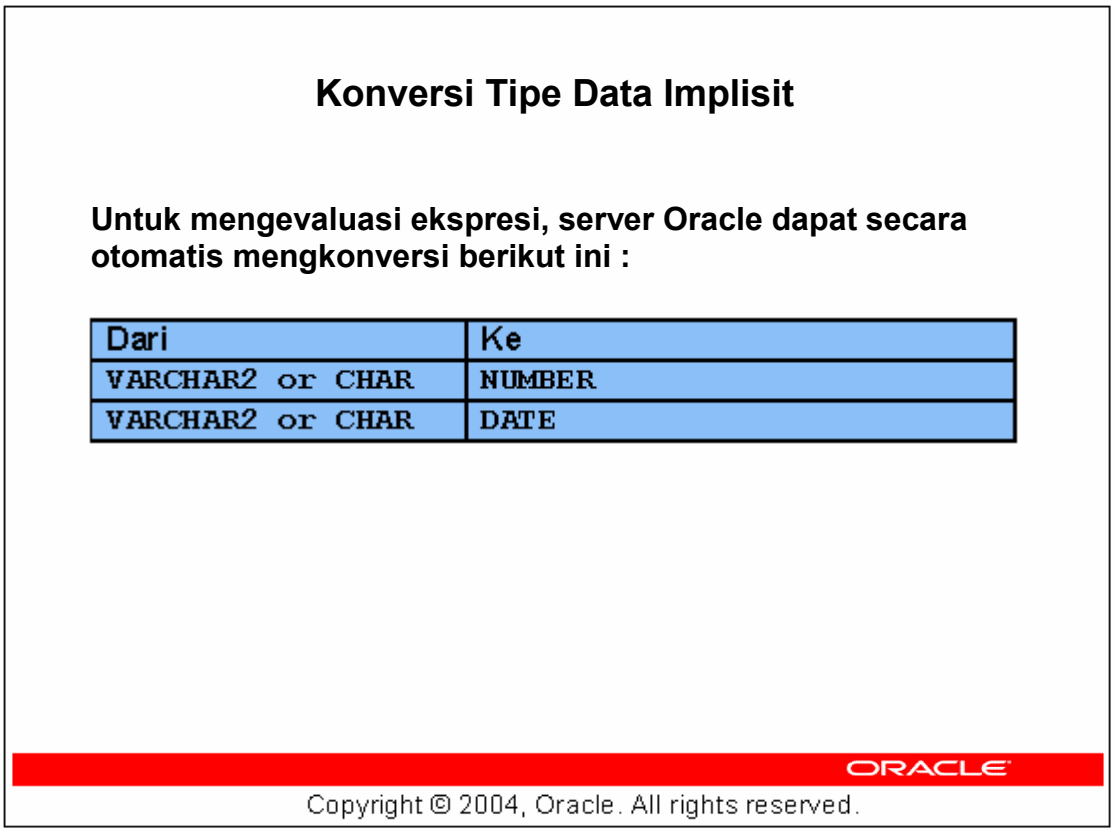
Dari	Ke
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

ORACLE

Konversi Tipe Data Implisit

Penugasan (*assignment)* berhasil jika server Oracle dapat mengkonversi tipe data dari suatu nilai yang digunakan dalam penugasan terhadap suatu target penugasan.

Sebagai contoh, hasil ekspresi `hire_date > '01-JAN-90'` dalam suatu konversi implisit dari suatu *string* '01-JAN-90' ke suatu tanggal.

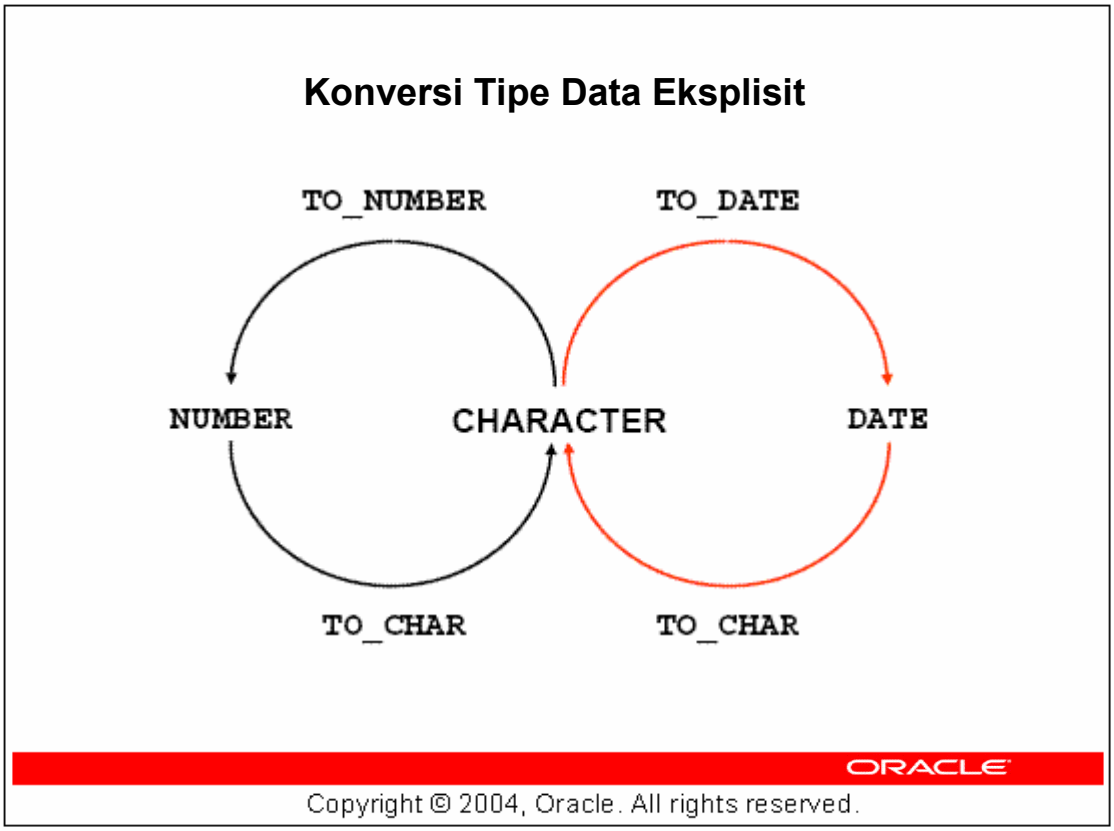


Konversi Tipe Data Implisit (lanjutan)

Pada umumnya, server Oracle menggunakan aturan untuk ekspresi-ekspresi ketika peng-konversi-an suatu tipe data diperlukan dalam bagian-bagian yang tidak tercakup oleh suatu aturan bagi konversi-konversi penugasan.

Sebagai contoh, hasil suatu ekspresi `salary = '20000'` dalam suatu konversi implisit dari suatu *string* `'20000'` ke bilangan 20000.

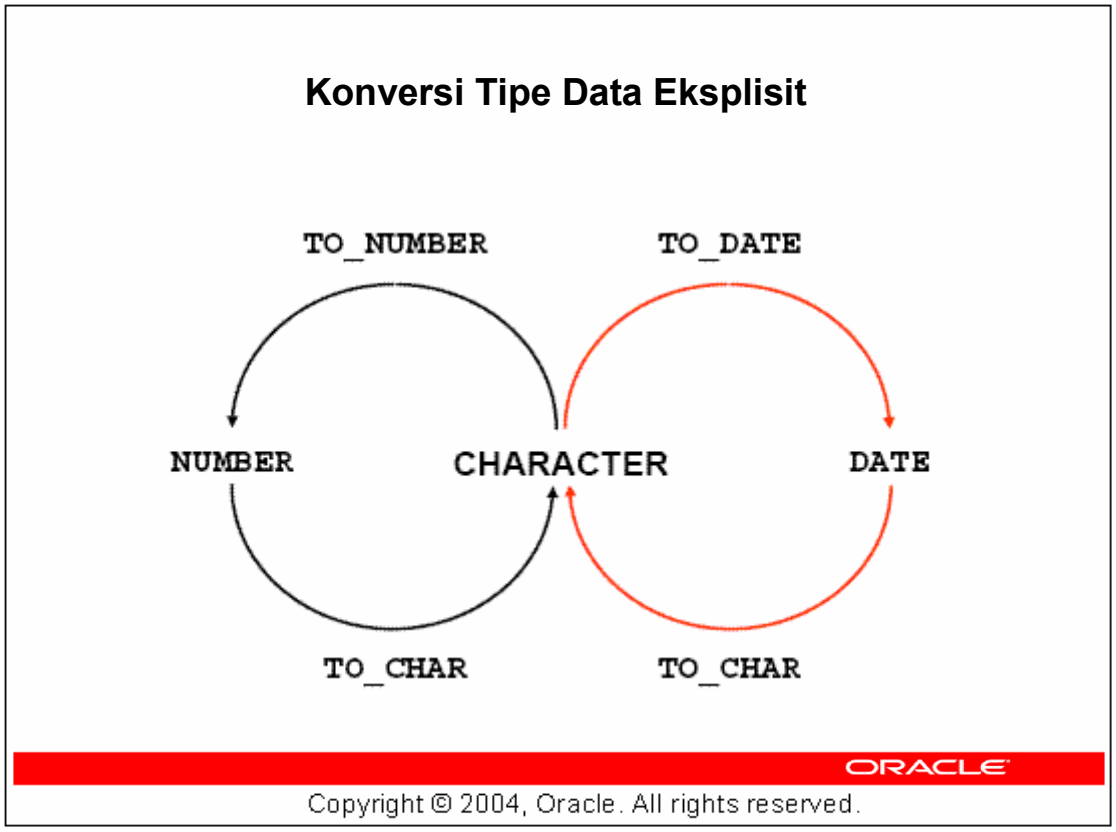
Catatan : Konvers-konversi `CHAR` ke `NUMBER` berhasil jika hanya karakter *string* menunjukkan suatu bilangan yang valid.



Konversi Tipe Data Eksplisit

SQL menyediakan tiga fungsi untuk mengkonversi suatu nilai dari satu tipe data ke tipe data lain :

Fungsi	Melakukan
`TO_CHAR(number|date,[ fmt], [nlsparams])`	Konversi-konversi nilai suatu bilangan atau tanggal ke suatu karakter *string* `VARCHAR2` dengan model format `fmt` **Konversi bilangan** : Suatu parameter `nlsparams` menentukan karakter-karakter berikut, yang dikembalikan oleh bagian format bilangan : - Karakter desimal - Pemisah kelompok - Simbol mata uang lokal - Simbol mata uang internasional Jika `nlsparams` atau parameter-parameter lainnya dihilangkan, fungsi ini menggunakan nilai parameter *default* untuk suatu sesi.



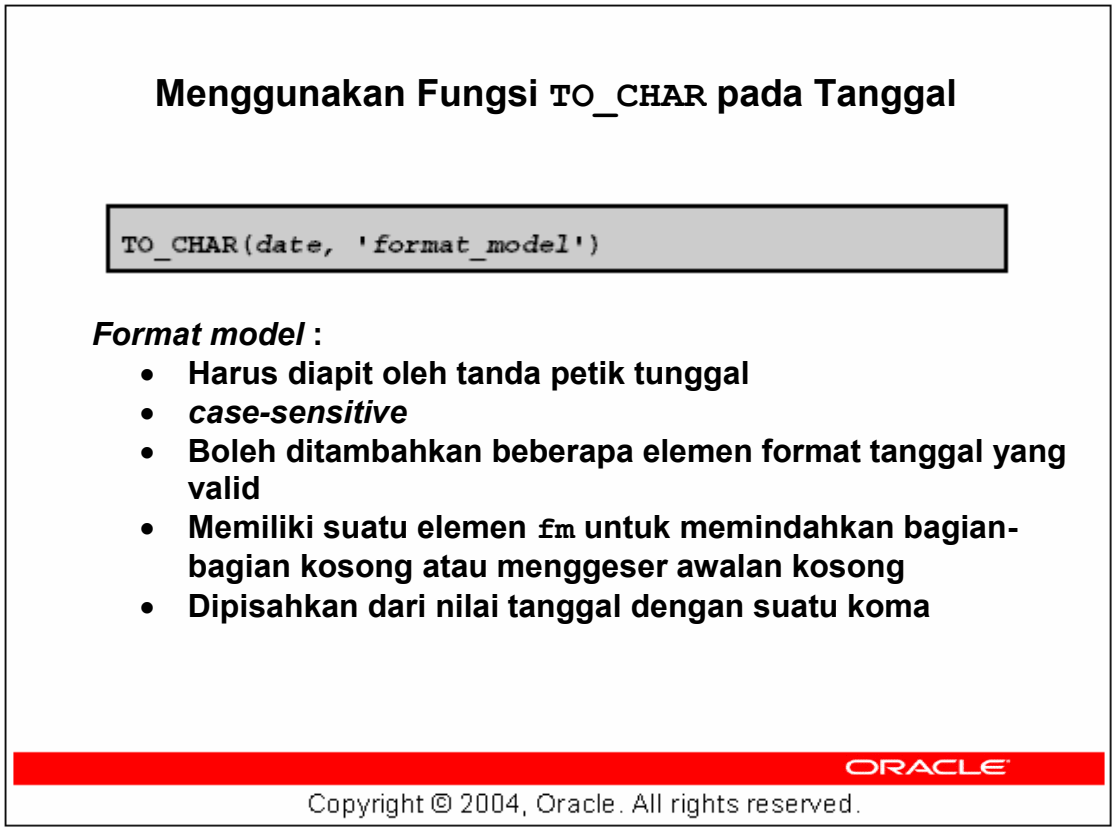
Konversi Tipe Data Eksplisit (lanjutan)

Fungsi	Melakukan
`TO_CHAR(number|date, [fmt], [nlsparams])`	**Konversi tanggal** : parameter-parameter `nlsparams` menentukkan bahasa dalam nama bulan dan nama hari serta dikembalikan berupa singkatan-singkatan. Jika parameter ini dihilangkan, fungsi ini menggunakan *default* bahasa-bahasa tanggal untuk suatu sesi.
`TO_NUMBER (char, [fmt], [nlsparams])`	Konversi-konversi suatu karakter *string* yang berisi digit-digit bilangan dalam suatu format yang ditentukan dengan model format optional *fmt*. Parameter `nlsparams` memiliki tujuan yang sama dalam fungsi ini seperti fungsi `TO_CHAR` untuk konversi bilangan.
`TO_DATE(char, [fmt], [nlsparams])`	Konversi-konversi suatu karakter *string* yang menunjukkan suatu tanggal untuk suatu tanggal sesuai dengan yang ditentukan pada *fmt*. Jika *fmt* dihilangkan, formatnya adalah DD-MON-YY. Parameter `nlsparams` memiliki tujuan yang sama dalam fungsi ini seperti fungsi `TO_CHAR` untuk konversi tanggal.

Konversi Tipe Data Eksplisit (lanjutan)

Catatan : Daftar dari fungsi-fungsi yang disebutkan dalam pelajaran ini hanya mencakup beberapa fungsi-fungsi konversi yang ada.

Untuk informasi lebih lanjut, lihat ”Conversion Functions” dalam *Oracle SQL Reference*.



Menampilkan suatu Tanggal dalam Suatu Format Khusus

Sebelumnya, semua nilai tanggal Oracle ditampilkan dalam format DD-MON-YY. Anda dapat menggunakan fungsi `TO_CHAR` untuk mengkonversi suatu tanggal dari format *default* tersebut ke suatu format yang Anda tentukan.

Pedoman-pedoman :

- Suatu *format model* harus diapit dengan tanda petik tunggal dan bersifat *case-sensitive*.
- Suatu *format model* dapat ditambahi beberapa elemen format tanggal yang valid. Pastikan untuk memisahkan nilai tanggal dari *format model* dengan sebuah koma.
- Nama-nama hari dan bulan pada output secara otomatis diisi dengan kosong.
- Untuk memindah isian kosong atau menggeser awalan kosong, gunakan elemen mode pengisian *fm*.
- Anda dapat mem-format suatu hasil *field* karakter dengan perintah *i*SQL*Plus `COLUMN`. (dibahas pada pelajaran selanjutnya).

```
SELECT employee_id, TO_CHAR(hired_date, 'MM/YY')Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH_HIRED
205	06/94

Elemen-Elemen Model Format Suatu Tanggal

Elemen	HASIL
YYYY	Tahun utuh dalam angka
YEAR	Pengejaan tahun (dalam bahasa Inggris)
MM	Dua digit nilai untuk nama bulan
MONTH	Nama utuh suatu bulan
MON	Tiga huruf singkatan nama suatu bulan
DY	Tiga huruf singkatan nama suatu hari dari suatu minggu
DAY	Nama utuh suatu hari dari suatu minggu
DD	Bilangan/angka hari suatu bulan

Contoh Elemen-Elemen Format yang Berlaku Format-Format Tanggal

Elemen	Keterangan
SCC atau CC	Abad; menyediakan awalan tanggal B.C. (= S.M.) dengan -
Tahun dalam Tanggal YYYY atau SYYYY	Tahun; menyediakan awalan tanggal B.C. (= S.M.) dengan -
YYY atau YY atau Y	Tiga, dua atau satu digit terakhir suatu tahun
Y, YYY	Tahun dengan koma di posisi ini
IYYY, IYY, IY, I	Empat-, tiga-, dua-, atau satu- digit tahun berdasarkan standar ISO
SYEAR atu YEAR	Ejaan tahun; menyediakan awalan tanggal B.C. (= S.M.) dengan -
BC atau AD	Menunjukkan tahun B.C. (= S.M.) atau A.D. (= M)
B.C. atau A.D.	Menunjukkan tahun B.C. (= S.M.) atau A.D. (= M) menggunakan periode
Q	Seperempat tahun
MM	Bulan; dua digit nilai
MONTH	Nama suatu bulan diisi dengan kosong pada panjang karakter ke sembilan
MON	Nama suatu bulan, disingkat tiga huruf
RM	Angka Romawi bulan
WW atau W	Minggu ke suatu tahun atau bulan
DDD atau DD atau D	Hari ke suatu tahun, bulan atau minggu
DAY	Nama suatu hari diisi dengan kosong pada panjang karakter ke sembilan
DY	Nama suatu hari; disingkat tiga huruf
J	Hari kalender Julian (kalender Julius Caesar); jumlah hari sejak 31 Desember 4713 Sebelum Masehi

Elemen-Elemen pada Model Format Tanggal

- **Elemen-elemen waktu memformat bagian waktu dari suatu tanggal :**

HH24:MI:SS AM	15:45:32 PM

- **Menambah karakter-karakter *string* dengan mengapitnya dalam tanda petik ganda :**

DD "of" MONTH	12 of OCTOBER

- **Akhiran angka menyebutkan bilangan**

ddspth	fourteenth

ORACLE

Elemen-Elemen Format Tanggal : Format-format Waktu

Gunakan format-format yang terdaftar dalam tabel berikut ini untuk menampilkan informasi waktu dan literal-literal dan untuk mengubah bilangan-bilangan ke ejaan bilangan-bilangan.

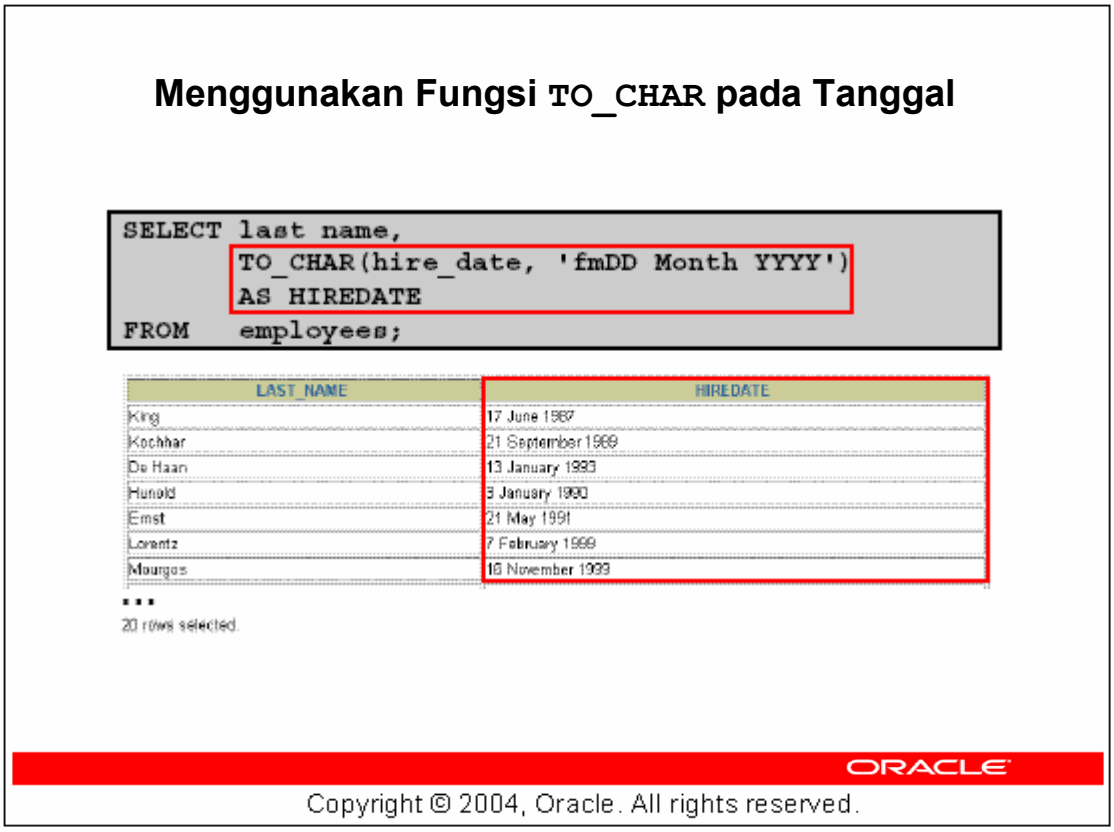
Element	Deskripsi
AM atau PM	Indikator *Meridian* (garis bujur)
A.M. or P.M.	Indikator *Meridian* dengan periode
HH atau HH12 atau HH24	Jam suatu hari, atau jam (1-12), atau jam (0-23)
MI	Menit (0-59)
SS	Detik (0-59)
SSSSS	Lewat detik-detik tengah malam (0-86399)

Format-Format Lain

Elemen	Deskripsi
/ . ,	Memberikan tanda baca pada hasil
“of the”	Memberikan tanda kutip rangkaian pada hasil

Menentukan Akhiran untuk Merubah Bilangan yang Ditampilkan

Elemen	Deskripsi
TH	Urutan bilangan (contoh, DDTH untuk 4TH)
SP	Pengejaan Bilangan (contoh, DDSP untuk FOUR)
SPTH or THSP	Pengejaan urutan bilangan (contoh, DDSTH untuk FOURTH)



Menggunakan Fungsi TO_CHAR pada Tanggal

Pernyataan SQL pada slide menampilkan nama belakang dan tanggal masuk untuk semua pegawai. Suatu tanggal masuk yang muncul 17 Juni 1987.

Contoh

Ubah contoh pada slide untuk menampilkan tanggal-tanggal dalam suatu format yang muncul seperti "*Seventeenth of June 1987 12:00:00 AM.*"

```
SELECT Last_name,
    TO_CHAR(hire_date, 'fmDdspth "OF" Month YYYY fmHH:MI:SS AM')
    HIREDATE
FROM   employees;
```

LAST_NAME	HIREDATE
King	Seventeenth of June 1987 12:00:00 AM
Kochhar	Twenty-First of September 1989 12:00:00 AM

. . .

Perhatikan bahwa bulan ditentukan menggunakan format model; dengan kata lain, bulan diawali huruf besar dan selanjutnya menggunakan huruf kecil.

Menggunakan Fungsi TO_CHAR pada Bilangan

```
TO_CHAR(number, 'format_model')
```

Berikut adalah beberapa dari elemen-elemen format yang dapat Anda gunakan pada suatu fungsi TO_CHAR untuk menampilkan sebuah nilai bilangan sebagai suatu karakter :

Elemen	Hasil
9	Menunjukkan bilangan
0	Memaksa nol supaya ditampilkan
$	Menempatkan tanda dollar
L	Menggunakan simbol mata uang local
.	Mencetak titik desimal
,	Mencetak tanda koma sebagai indicator ribuan

Menggunakan Fungsi TO_CHAR Pada Bilangan

Saat sedang bekerja dengan nilai-nilai bilangan seperti karakter-karakter *string*, Anda harus mengubah bilangan-bilangan ke tipe data karakter menggunakan fungsi TO_CHAR, yang menerjemahkan suatu nilai dari tipe data NUMBER ke tipe data VARCHAR2. Teknik ini berguna terutama pada penggabungan (*concatenation*).

Menggunakan Fungsi `TO_CHAR` pada Bilangan (lanjutan)

Elemen-elemen Format Bilangan

Jika Anda mengubah suatu bilangan ke tipe data karakter, Anda bisa menggunakan elemen-elemen format berikut ini :

Elemen	Keterangan	Contoh	Hasil
9	Posisi numerik (jumlah dari 9 menentukan lebar tampilan)	999999	1234
0	Menampilkan awalan nol	099999	001234
$	Simbol dollar	$999999	$1234
L	Simbol mata uang lokal	L999999	FF1234
D	Mengembalikan suatu posisi karakter desimal tertentu. Defaultnya adalah (.).	99D99	99.99
.	Titik desimal pada posisi tertentu	999999.99	1234.00
G	Mengembalikan pemisah kelompok (*group separator*) pada posisi tertentu. Anda dapat menentukan pemisah kelompok-kelompok (*multiple group*) pada model format bilangan.	9,999	9G999
,	Koma pada posisi tertentu	999,999	1,234
MI	Tanda minus di sebelah kanan (nilai negatif)	999999MI	1234-
PR	Kurung bilangan-bilangan negatif	999999PR	<1234>
EEEE	Notasi ilmiah (format harus ditentukan empat E)	99.999EEEE	1.234E+03
U	Mengembalikan suatu posisi tertentu mata uang“*Euro*” (atau yang lain)	U9999	€1234
V	*n* kali kelipatan 10 (*n*= bilangan 9 setelah V)	9999V99	123400
S	Mengembalikan nilai negatif atau positif	S9999	-1234 atau +1234
B	Menampilan nilai-nilai nol sebagai kosong, bukan 0	B9999.99	1234.00

Menggunakan Fungsi TO_CHAR pada Bilangan

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

SALARY
$6,000.00

ORACLE

Pedoman-Pedoman

- Server Oracle menampilkan suatu *string* (rangkaian) sejumlah tanda (#) di tempat dari seluruh bilangan yang melampaui digit-digit bilangan pada digit-digit yang disediakan dalam model format.
- Server Oracle membulatkan nilai desimal yang disimpan ke bilangan pada posisi desimal yang disediakan dalam model format

Menggunakan Fungsi-Fungsi TO_NUMBER dan TO_DATE

- **Mengubah suatu rangkaian karakter ke suatu format bilangan menggunakan fungsi TO_NUMBER :**

```
TO_NUMBER(char[, 'format_model'])
```

- **Mengubah suatu rangkaian karakter ke suatu format tanggal menggunakan fungsi TO_DATE :**

```
TO_DATE(char[, 'format_model'])
```

- **Fungsi-fungsi ini memiliki sebuah *modifier* (pengubah) fx. *Modifier* ini menentukan kesesuaian yang pasti pada argumen karakter dan model format tanggal dari suatu fungsi TO_DATE.**

ORACLE

Menggunakan Fungsi-Fungsi TO_NUMBER dan TO_DATE

Mungkin Anda ingin mengubah suatu rangkaian karakter baik ke suatu bilangan atau tanggal. Untuk melakukannya, gunakan fungsi-fungsi TO_NUMBER atau TO_DATE. Model format yang Anda pilih sesuai dengan format elemen-elemen yang ditunjukkan sebelumnya.

Modifier fx menentukan kesesuaian yang pasti pada argumen karakter dan model format tanggal dari suatu fungsi TO_DATE:

- Pemberian tanda baca dan tanda petik dalam argumen karakter harus benar-benar sesuai dengan (kecuali untuk bentuk/*case*) bagian-bagian yang berhubungan pada model format.
- Suatu argumen karakter tidak boleh memiliki ruang kosong berlebihan. Tanpa fx, Oracle mengabaikan ruang kosong berlebih.
- Data numeric dalam argumen karakter harus mempunyai jumlah digit-digit yang sama dengan elemen yang berhubungan pada model format. Tanpa fx, angka-angka dalam argumen karakter dapat menghilangkan awalan kosong.

Contoh

Tampilkan nama dan tanggal masuk untuk semua pegawai yang mulai bekerja pada tanggal 24 Mei 1999. Karena digunakan fx modifier, suatu kesesuaian yang pasti diperlukan dan spasi setelah kata May tidak dikenali :

```
SELECT   last_name, hire_date
FROM     employees
WHERE    hire_date=TO_DATE ('May 24 1999', 'fxMonth DD, YYYY');
*
WHERE    hire_date =TO_DATE ('May 24 1999', 'fxMonth DD, YYYY');
                            *
ERROR at line 3:
ORA-01858 : a non-numeric character was found where a numeric was
expected.
```

Format Tanggal RR

Tahun sekarang	Tanggal Tertentu	Format RR	Format YY
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		Dua digit suatu tahun tertentu :	
		0-49	50-99
Dua digit pada tahun sekarang :	0-49	Tanggal yang dikembalikan adalah abad sekarang	Tanggal yang dikembalikan adalah abad sebelum abad sekarang
	50-99	Tanggal yang dikembalikan adalah abad setelah abad sekarang	Tanggal yang dikembalikan adalah abad sekarang

Elemen Format Tanggal RR

Format tanggal RR sama dengan elemen YY, tetapi Anda dapat menggunakannya untuk menentukan perbedaan abad. Gunakan elemen format tanggal RR daripada YY sehingga abad pada nilai balik merubah nilai yang dikembalikan sesuai dengan dua digit tahun yang ditentukan dan dua digit terakhir pada tahun sekarang. Tabel dalam slide meringkas sifat elemen RR.

Tahun Sekarang	Tanggal yang diberikan	Tafsiran (RR)	Tafsiran (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

Contoh Format Tanggal RR

Untuk menemukan pegawai-pegawai yang masuk sebelum tahun 1990, gunakan format tanggal RR, yang menghasilkan hasil yang sama apakah perintah dijalankan pada tahun 1999 atau sekarang :

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIR
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

ORACLE

Contoh Format Tanggal RR

Untuk menemukan pegawai yang masuk sebelum tahun 1990, dapat digunakan format RR. Karena tahun sekarang lebih besar dari 1999, format RR menafsirkan bagian tahun dari suatu tanggal dari 1950 sampai 1999.

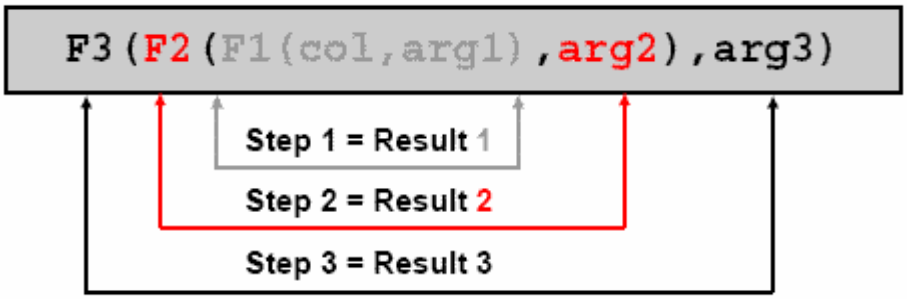
Perintah berikut ini, dilain pihak, tidak ada baris yang dihasilkan akan dipilih karena format YY menafsirkan bagian tahun dari suatu tanggal pada abad sekarang (2090).

```
SELECT last_name, TO_CHAR (hire_date,'DD-Mon-YYYY')
FROM   employees
WHERE  TO_DATE(hire_date,'DD-Mon-YY') < '01-Jan-1990' ;

No rows selected
```

Nested Functions (Fungsi-Fungsi Bersarang)

- ***Single-row functions*** **dapat disarangkan (*nested*) dalam beberapa level.**
- ***Nested functions*** **dievaluasi dari level terdalam ke level terluar.**



Nested Functions (Fungsi-Fungsi Bersarang)

Single-row functions dapat disarangkan sampai beberapa kedalaman. *Nested functions* dievaluasi dari level terdalam ke level yang terluar. Beberapa contoh berikut menunjukkan Anda fleksibilitas dari fungsi-fungsi ini.

Nested Functions
(Fungsi-Fungsi Bersarang)

```
SELECT last name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8
Hunold	HUNOLD_US
Ernst	ERNST_US
Lorentz	LORENTZ_US

ORACLE

Nested Function (lanjutan)

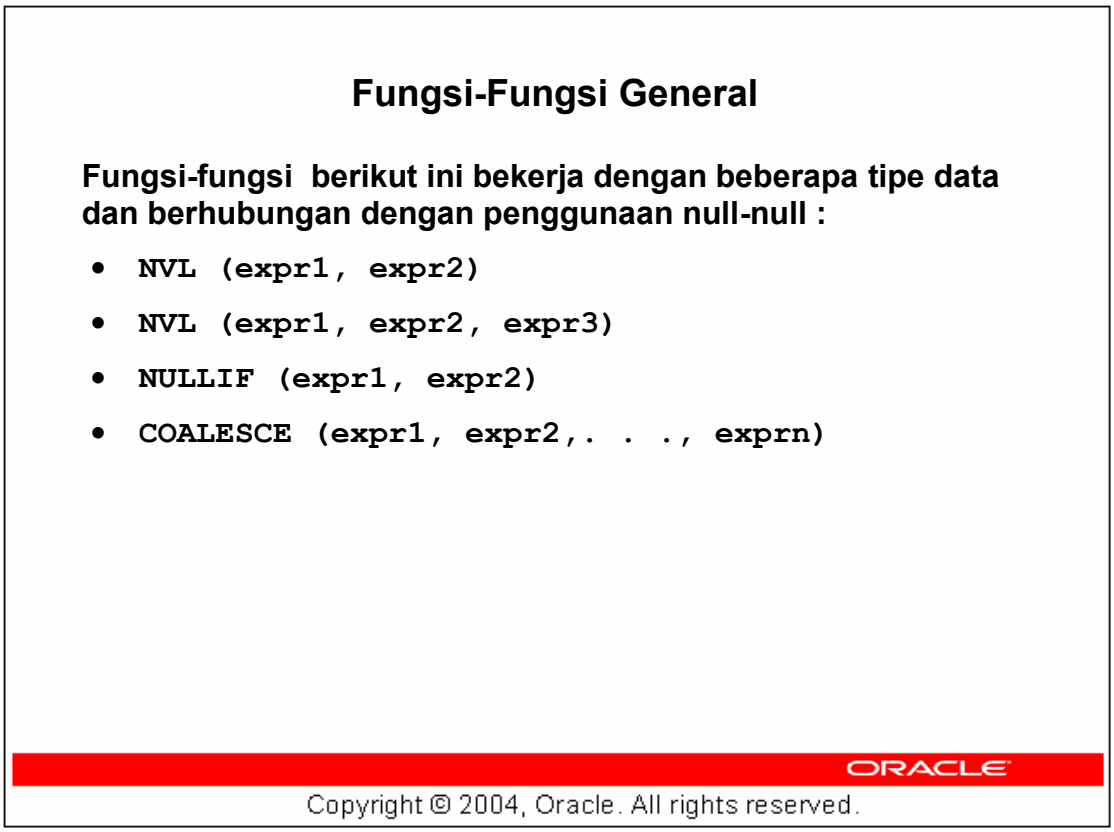
Contoh pada slide menampilkan nama-nama belakang pegawai yang bekerja pada departemen 60. Evaluasi pernyataan SQL melibatkan tiga tahap:

1. Fungsi terdalam mengambil delapan karakter pertama dari nama belakang.
   `Result1 = SUBSTR (LAST_NAME, 1, 8)`
2. Fungsi terluar menggabungkan hasil dengan _US.
   `Result2 = CONCAT (Result1, '_US')`
3. Fungsi yang paling terluar mengubah hasil menjadi huruf besar.

Seluruh ekspresi menjadi judul kolom karena tidak ada kolom alias yang diberikan.

Contoh
Tampilkan tanggal hari Jumat berikutnya enam bulan dari tanggal masuk. Tanggal yang dihasilkan harus menunjukkan *Friday, August 13th, 1999*. Urutkan hasil berdasarkan tanggal masuk.

```
SELECT     TO_CHAR (NEXT DAY (ADD_MONTHS
           (hire_date, 6), 'FRIDAY'),
           'fmDay, Month DDth, YYYY')
           "Next 6 Month Review"
FROM       employees
ORDER BY   hire_date;
```



Fungsi-Fungsi General

Fungsi-fungsi ini bekerja dengan beberapa tipe data dan berhubungan dengan penggunaan nilai-nilai *null* dalam daftar ekspresi.

FUNGSI	Keterangan
NVL	Mengubah suatu nilai *null* menjadi nilai actual.
NVL2	Jika expr1 bukan *null*, NVL2 mengembalikan expr2. Jika expr1 *null*, NVL2 mengembalikan expr3. Argument expr1 dapat berupa beberapa tipe data.
NULLIF	Membandingkan dua ekspresi dan mengembalikan *null* jika sama; mengembalikan ekspresi pertama jika tidak sama.
COALESCE	Mengembalikan ekspresi bukan *null* yang pertama dalam daftar ekspresi.

Catatan : Untuk informasi lebih lanjut mengenai ratusan fungsi-fungsi yang ada, lihat "Functions" dalam *Oracle SQL Reference*.

Fungsi NVL

Mengubah nilai *null* menjadi nilai aktual :

- **Tipe data – tipe data yang dapat digunakan adalah tanggal, karakter, dan angka.**
- **Tipe data-tipe data harus sesuai :**
  - **NVL (commission_pct, 0)**
  - **NVL (hire_date, '01-Jan-97')**
  - **NVL (job_id, 'No Job Yet')**

NVL *Function*

Untuk mengubah nilai *null* menjadi nilai aktual, gunakan fungsi NVL.

Sintak

```
NVL (expr1, expr2)
```

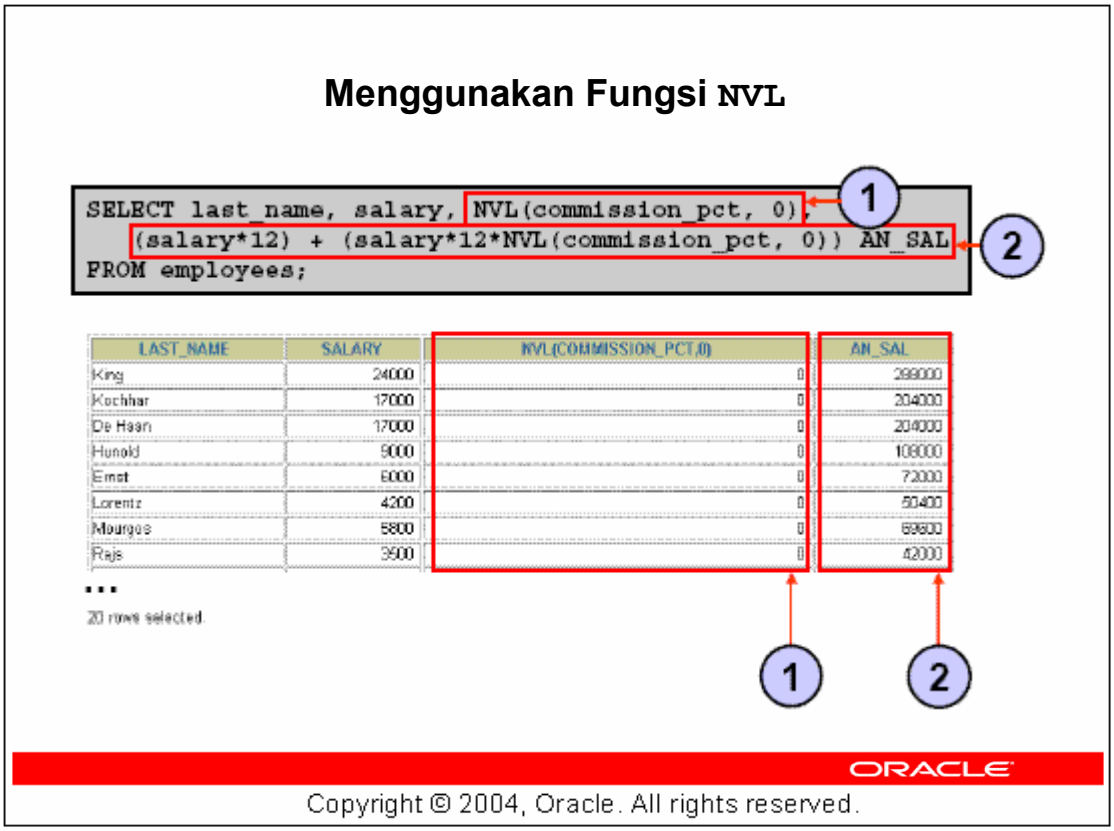
Dalam sintak :

- *expr1* adalah nilai asal atau ekspresi yang mungkin mengandung *null*.
- *expr2* adalah nilai yang diinginkan untuk mengubah *null*.

Anda dapat menggunakan fungsi NVL untuk mengubah beberapa tipe data, tetapi nilai kembalian selalu sama dengan tipe data dari *expr1*.

Konversi-konversi NVL untuk beberapa tipe data

Tipe Data	Contoh konversi
NUMBER	NVL (number_column, 9)
DATE	NVL (date_column, '01-Jan-95')
CHAR atau VARCHAR2	NVL(character_column,'Unavailable')



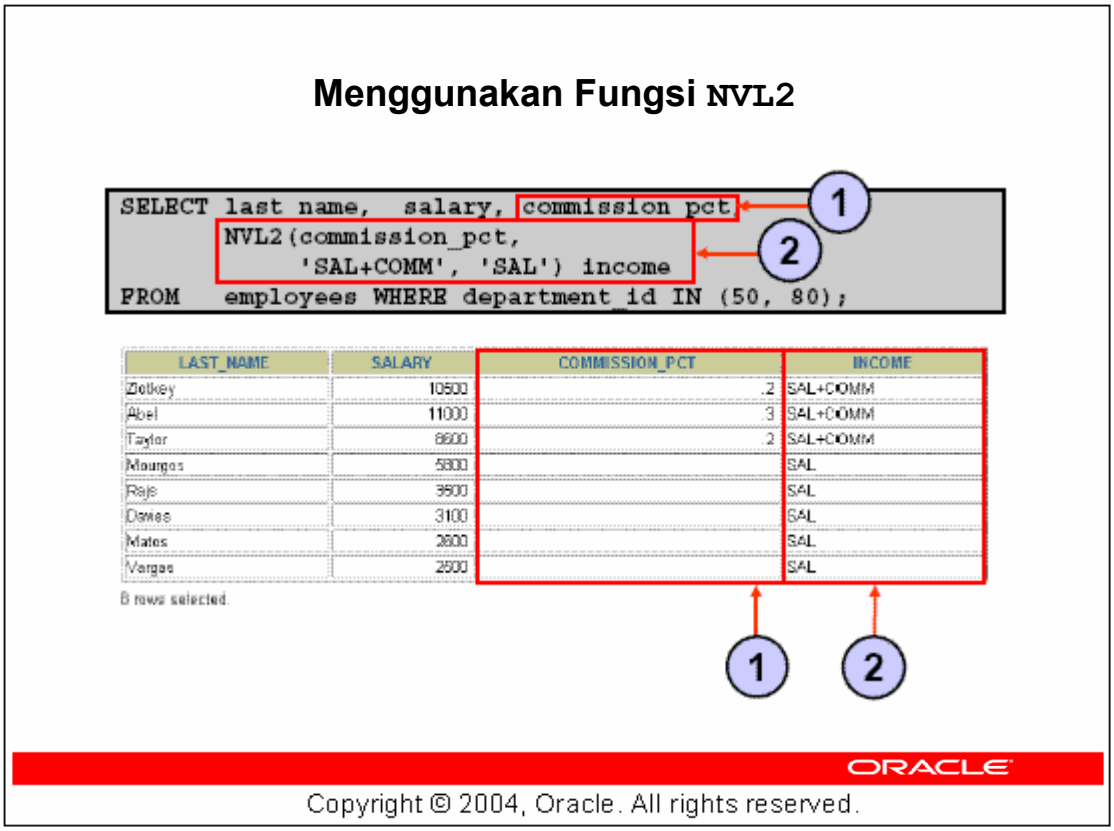
Menggunakan Fungsi NVL

Untuk menghitung kompensasi tahunan dari semua pegawai, Anda membutuhkan perkalian antara penghasilan bulanan dengan 12 kemudian menambah prosentase komisi kepada hasil :

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

LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
. . .			
Vargas	2500		
Zlotkey	10500	.2	151200
Abel	11000	.3	171600
Taylor	8600	.2	123840
. . .			

Perhatikan bahwa kompensasi tahunan yang dihitung hanya untuk pegawai yang mendapatkan komisi. Jika beberapa nilai kolom dalam ekspresi adalah *null*, maka hasilnya adalah *null*. Untuk menghitung nilai-nilai dari semua pegawai, Anda harus mengubah nilai *null* menjadi angka sebelum menggunakan operator aritmatika. Contoh pada slide, fungsi NVL digunakan untuk mengubah nilai *null* menjadi nol.



Menggunakan Fungsi NVL2

Fungsi NVL2 memeriksa ekspresi pertama. Jika ekspresi pertama bukan *null*, maka fungsi NVL2 mengembalikan ekspresi kedua. Jika ekspresi pertama *null*, maka ekspresi ketiga yang akan dikembalikan.

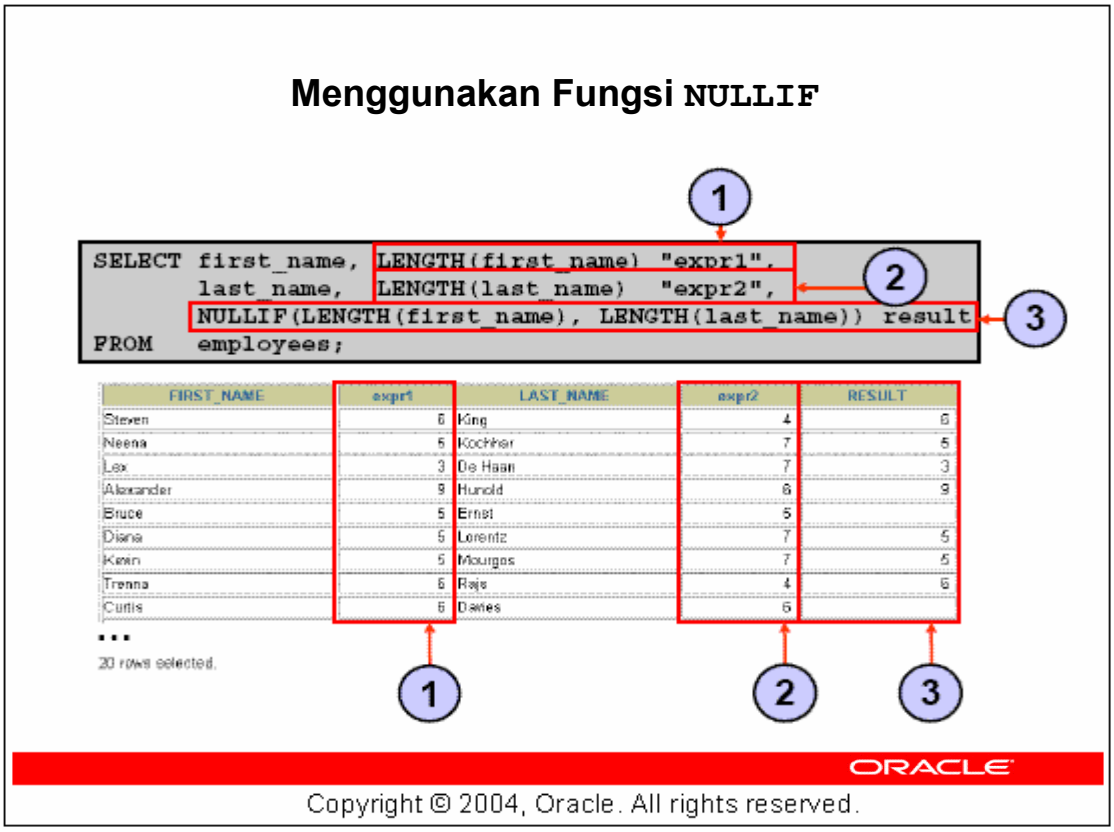
Sintak

```
NVL (expr1, expr2, expr3)
```

Dalam sintak :

- *expr1* adalah nilai atau ekspresi asal yang mungkin berisi *null*.
- *expr2* adalah nilai yang dikembalikan jika *expr1* bukan *null*.
- *expr3* adalah nilai yang dikembalikan jika *expr1* adalah *null*.

Contoh pada slide menampilkan, kolom COMMISSION_PCT diperiksa. Jika sebuah nilai terdeteksi, ekspresi kedua dari SAL+COMM akan dikembalikan. Jika kolom COMMISSION_PCT mangandung nilai *null*, ekspresi ketiga dari SAL akan dikembalikan.
Argumen *expr1* dapat bertipe data apapun. Argumen *expr2* dan *expr3* dapat bertipe data apapun kecuali LONG. Jika tipe data *expr2* dan *expr3* berbeda, server Oracle mengubah *expr3* menjadi tipe data *expr2* sebelum membandingkannya, kecuali jika *expr3* adalah konstanta *null*. Dalam kasus berikutnya, suatu konversi tipe data tidak dipentingkan. Tipe data mengembalikan nilai selalu sama dengan tipe dari *expr2*, kecuali *expr2* adalah data karakter, dalam hal ini nilai data yang dikembalikan adalah bertipe VARCHAR2.



Penggunaan Fungsi NULLIF

Fungsi NULLIF membandingkan dua ekspresi. Jika sama, fungsi akan mengembalikan *null*. Jika tidak sama, fungsi akan mengembalikan ekspresi pertama. Anda tidak dapat menyebutkan literal NULL sebagai ekspresi pertama.

Sintak

```
NULLIF(expr1, expr2)
```

Dalam sintak :

- *expr1* adalah nilai asal yang dibandingkan dengan *expr2*.
- *expr2* adalah nilai asal yang dibandingkan dengan *expr1* (Jika tidak sama dengan *expr1*, maka *expr1* yang dikembalikan).

Contoh pada slide menampilkan, panjang dari nama depan dalam tabel EMPLOYEES dibandingkan dengan panjang dari nama belakang dalam tabel EMPLOYEES. Ketika panjang nama-nama tidak sama, maka panjang dari nama depan akan ditampilkan.

Catatan : Fungsi NULLIF secara logika sama dengan ekspresi CASE berikut ini. Ekspresi CASE didiskusikan dalam halaman berikutnya :

```
CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
```

Menggunakan COALESCE *Function*

- **Keuntungan dari fungsi COALESCE dibandingkan dengan fungsi NVL adalah bahwa fungsi COALESCE dapat mengambil beberapa nilai alternatif.**
- **Jika ekspresi pertama adalah bukan *null*, fungsi COALESCE mengembalikan ekspresi tersebut; jika tidak ia akan melakukan COALESCE dari ekspresi yang tersisa.**

ORACLE

Menggunakan Fungsi COALESCE

Fungsi COALESCE mengembalikan ekspresi bukan *null* (*non-null*) yang pertama yang terdapat dalam daftar.

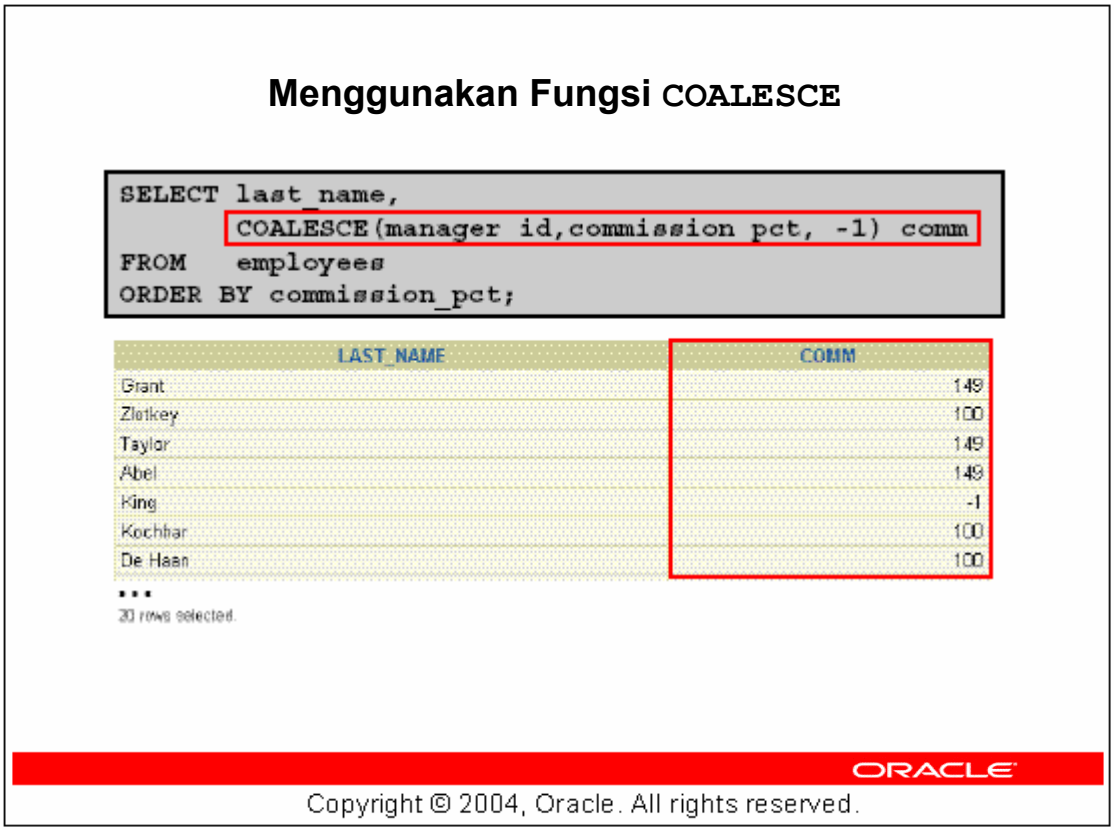
Sintak

```
COALESCE (epr1, expr2,…exprn)
```

Dalam sintak:

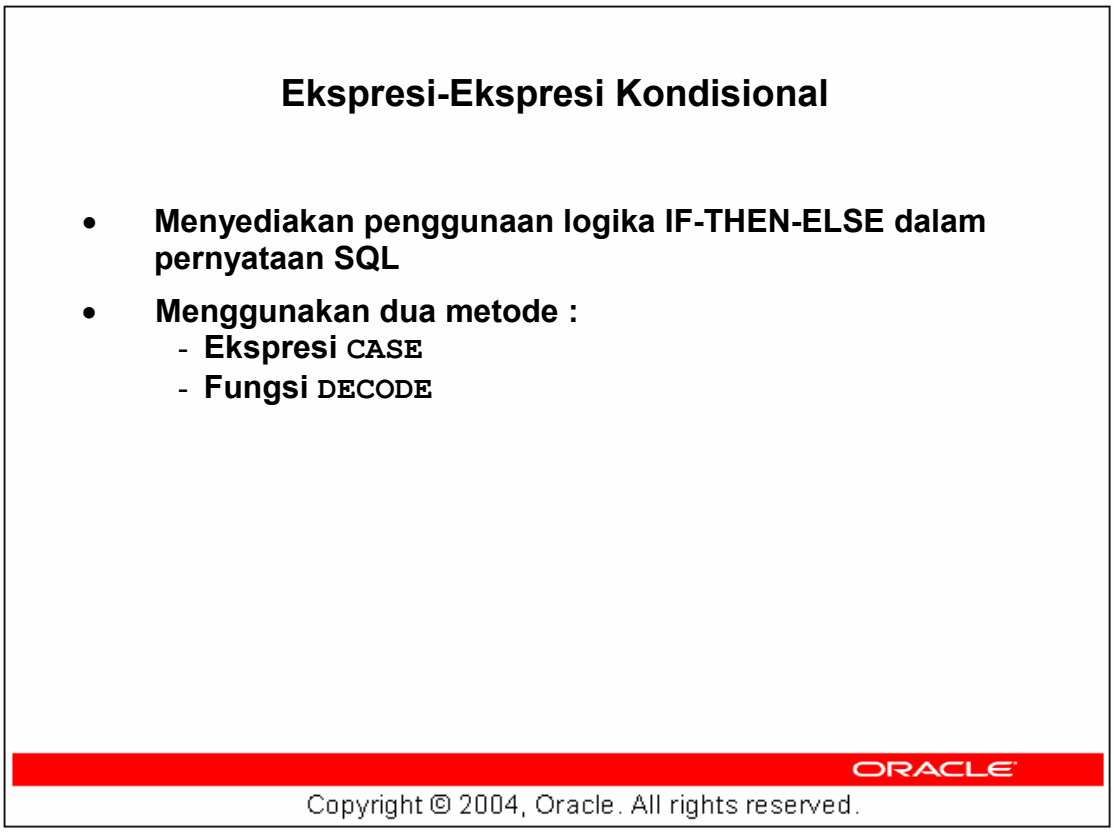
- *expr1* mengembalikan ekspresi *expr1* jika ia bukan *null*.
- *expr2* mengembalikan ekspresi *expr2* jika ekspresi pertama *null* dan ekspresi *expr2* bukan *null*.
- *exprn* mengembalikan ekspresi *exprn* jika ekspresi-ekspresi sebelumnya *null*.

Semua ekspresi harus bertipe data sama.



Menggunakan Fungsi COALESCE (lanjutan)

Contoh pada slide menampilkan, jika nilai `MANAGER_ID` bukan *null*, ia akan ditampilkan. Jika nilai `MANAGER_ID` *null*, maka `COMMISSION_PCT` ditampilkan. Jika nilai `MANAGER_ID` dan `COMMISSION_PCT` *null*, maka nilai -1 yang akan ditampilkan.



Ekspresi-Ekspresi Kondisional

Dua metode yang digunakan untuk implementasi pemrosesan kondisi (logika IF_THEN_ELSE) dalam pernyataan SQL adalah ekspresi `CASE` dan fungsi `DECODE`.

Catatan : Ekspresi `CASE` sesuai dengan **ANSI SQL**. Fungsi `DECODE` adalah khusus untuk sintak Oracle.

Ekspresi CASE

Kemudahan-kemudahan pencarian kondisional dengan melakukan pekerjaan pada suatu pernyataan IF-THEN-ELSE :

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


Ekspresi CASE

Ekspresi `CASE` mengijinkan Anda untuk menggunakan logika IF-THEN-ELSE dalam pernyataan SQL tanpa meminta prosedur-prosedur.

Dalam ekspresi `CASE` sederhana, server Oracle mencari pasangan `WHEN . . . THEN` pertama dimana `expr` adalah sama dengan `comparison_expr` dan mengembalikan `return_expr`. Jika tidak ada dari pasangan `WHEN . . . THEN` sesuai dengan kondisi ini, dan jika sebuah klausa `ELSE` ada, maka server Oracle akan mengembalikan `else_expr`. Jika tidak, server Oracle akan mengembalikan *null*. Anda tidak bisa menentukan literal `NULL` untuk semua `return_exprs dan else_expr`.

Semua ekspresi-ekspresi (`expr, comparison_expr dan return_expr`) harus bertipe data sama, yang bisa jadi `CHAR, VARCHAR2, NCHAR,` atau `NVARCHAR2.`

Menggunakan Ekspresi CASE

Kemudahan-kemudahan pencarian kondisional dengan melakukan suatu pekerjaan pada suatu pernyataan IF-THEN-ELSE :

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE       salary END          "REVISED_SALARY"
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Raja	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

Menggunakan Ekspresi CASE

Dalam pernyataan SQL pada slide, nilai dari JOB_ID dikodekan. Jika JOB_ID adalah IT_PROG, kenaikan gaji adalah 10%; jika JOB_ID adalah ST_CLERK, kenaikan gaji adalah 15%; jika JOB_ID adalah SA_REP, kenaikan gaji adalah 20%. Selain job-job itu, tidak ada kenaikan gaji.

Pernyataan yang sama dapat ditulis dengan fungsi DECODE.

Berikut ini adalah contoh pencarian ekspresi CASE. Dalam pencarian ekspresi CASE, pencarian terjadi dari kiri ke kanan sampai suatu kejadian dari daftar kondisi ditemukan, lalu ia mengembalikan ekspresi yang dikembalikan. Jika tidak ada kondisi yang benar ditemukan, dan jika ada suatu klausa ELSE, ekspresi yang dikembalikan dalam klausa ELSE akan dikembalikan; jika tidak, NULL akan dikembalikan.

```
SELECT last_name, salary,
(CASE  WHEN salary<5000 THEN 'Low'
   WHEN salary<10000 THEN 'Medium'
   WHEN salary<20000 THEN 'Good'
   ELSE 'Excellent'
END)   qualified_salary
FROM   employees;
```

Fungsi DECODE

Kemudahan-kemudahan pencarian kondisional dengan melakukan suatu pekerjaan pada suatu ekspresi CASE atau suatu pernyataan IF-THEN-ELSE :

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

ORACLE

Fungsi DECODE

Fungsi DECODE mengkodekan suatu ekspresi dalam suatu cara yang sama dengan logika IF-THEN-ELSE yang digunakan dalam beragam bahasa. Fungsi DECODE mengkodekan *expression* setelah membandingkannya dengan setiap nilai *search*. Jika ekspresi sama dengan *search*, *result* akan dikembalikan.

Jika nilai default dihilangkan, suatu nilai *null* akan dikembalikan dimana nilai yang dicari tidak sesuai dengan beberapa nilai hasil.

Menggunakan Fungsi DECODE

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Grant	SA_REP	7000	8400

20 rows selected.

Menggunakan Fungsi DECODE

Dalam pernyataan SQL pada slide, nilai dari `JOB_ID` diuji. Jika `JOB_ID` adalah `IT_PROG`, kenaikan gaji adalah 10%; jika `JOB_ID` adalah `ST_CLERK`, kenaikan gaji adalah 15%; jika `JOB_ID` adalah `SA_REP`, kenaikan gaji adalah 20%. Selain job-job itu, tidak ada kenaikan gaji.

Pernyataan yang sama dapat diekspresikan dalam *pseudocode* seperti pernyataan IF-THEN-ELSE :

```
IF job_id = 'IT_PROG'     THEN salary = salary*1.10
IF job_id = 'IT_PROG'     THEN salary = salary*1.15
IF job_id = 'IT_PROG'     THEN salary = salary*1.20
ELSE salary = salary
```

Menggunakan Fungsi DECODE

Menampilkan tingkat tarif pajak yang sesuai untuk masing-masing pegawai dalam departemen 80 :

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                     0, 0.00,
                     1, 0.09,
                     2, 0.20,
                     3, 0.30,
                     4, 0.40,
                     5, 0.42,
                     6, 0.44,
                        0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

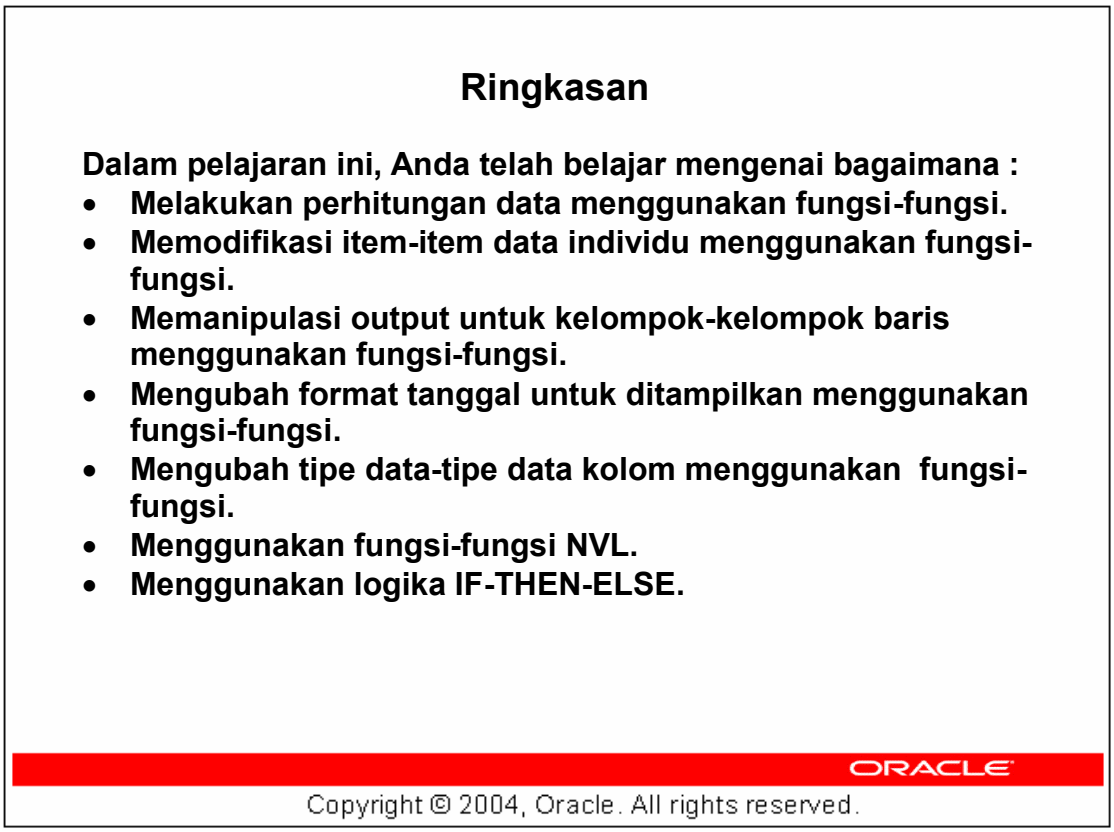
ORACLE

Menggunakan Fungsi DECODE (lanjutan)

Slide ini menunjukkan contoh lain menggunakan fungsi DECODE. Dalam contoh ini, kita menentukan tingkat tarif pajak untuk setiap pegawai dalam departemen 80 berdasarkan gaji bulanan. tingkat tarif pajak adalah sebagai berikut :

Range Gaji Bulanan	*Tarif Pajak*
$0.00 – 1,999.99	00%
$2,000.00-3,999.99	09%
$4,000.00 – 5,999.99	20%
$6,000.00 – 7,999.99	30%
$8,000.00 – 9,999.99	40%
$10,000.00 – 11,999.99	42%
$12,200.00 – 13,999.99	44%
$14,000.00 atau lebih	45%

LAST_NAME	SALARY	TAX_RATE
Zlotkey	10500	.42
Abel	11000	.42
Taylor	8600	.4



Ringkasan

Single-row functions dapat disarangkan (*nested*) ke dalam beberapa level. *Single-row functions* dapat memanipulasi berikut ini :

- Data Karakter : `LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH`
- Data Angka : `ROUND, TRUNC, MOD`
- Data Tanggal : `MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC`

Ingat hal-hal berikut ini :

- Nilai tanggal dapat menggunakan operator-operator aritmatika.
- Fungsi-fungsi konversi dapat mengubah karakter, tanggal, dan nilai angka : `TO_CHAR, TO_DATE, TO_NUMBER`.
- Ada beberapa fungsi berhubungan dengan *null-null*, terdiri dari `NVL, NVL2, NULLIF` dan `COALESCE`.
- Logika IF-THEN-ELSE dapat diaplikasikan dalam pernyataan SQL dengan menggunakan ekspresi `CASE` dan fungsi `DECODE`.

SYSDATE dan DUAL

`SYSDATE` adalah fungsi tanggal yang mengembalikan tanggal dan waktu saat ini. Biasanya dengan memilih `SYSDATE` dari tabel maya (*dummy table*) yang disebut `DUAL`.