9

Menggunakan Pernyataan DDL untuk Membuat dan Mengelola Tabel-Tabel

ORACLE

Tujuan

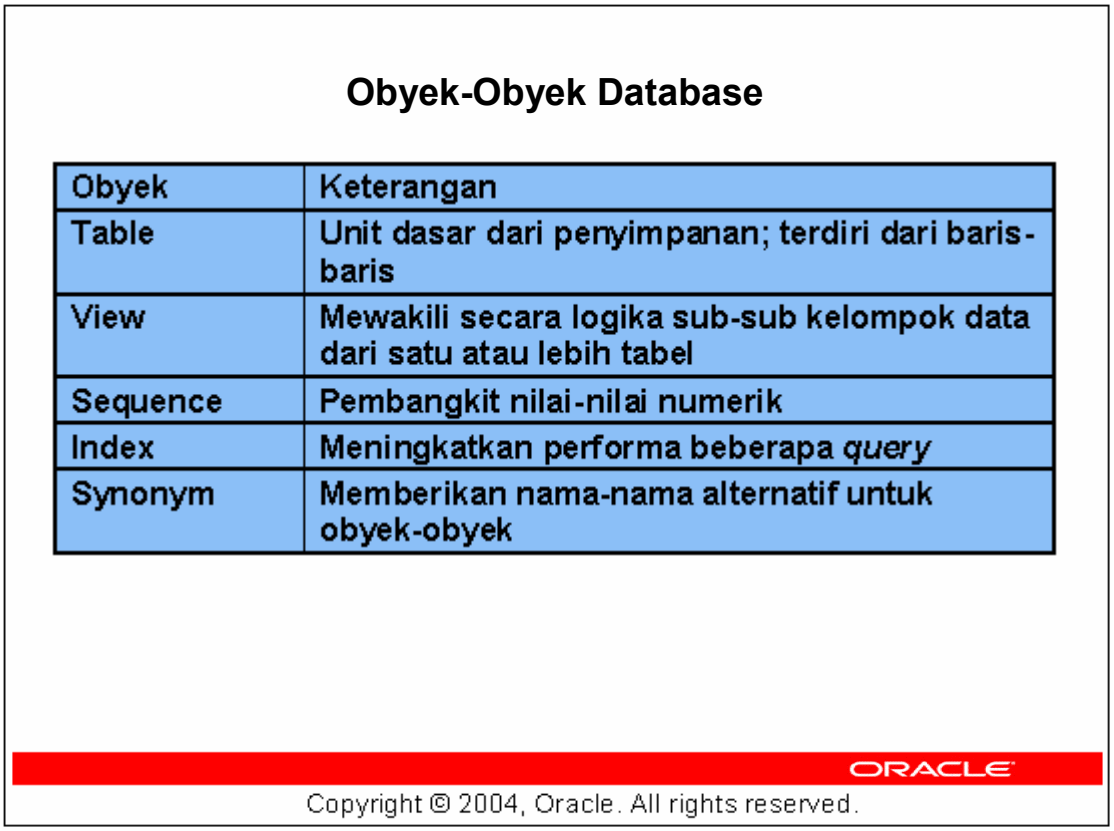
Setelah menyelesaikan pelajaran ini, Anda akan dapat melakukan hal-hal berikut :

- **Mengelompokkan obyek-obyek utama database**
- **Menampilkan struktur tabel**
- **Daftar tipe data-tipe data yang ada untuk kolom-kolom**
- **Membuat suatu tabel sederhana**
- **Memahami bagaimana *constraint-constraint* dibuat saat pembuatan tabel**
- **Menjelaskan bagaimana skema obyek-obyek bekerja**

ORACLE

Tujuan

Dalam pelajaran ini, Anda dikenalkan pada pernyataan *Data Definition Language* (DDL). Anda diajari dasar-dasar bagaimana untuk membuat tabel-tabel sederhana, mengubahnya dan menghapusnya. Ditunjukkan tipe data-tipe data yang ada dalam DDL dan dikenalkan konsep-konsep skema. *Constraint-constraint* terkait dalam materi ini. Ditunjukkan dan dijelaskan *exception messages* yang dibangkitkan dari *constraint-constraint* yang dilanggar pada waktu DML.



Obyek-obyek Database

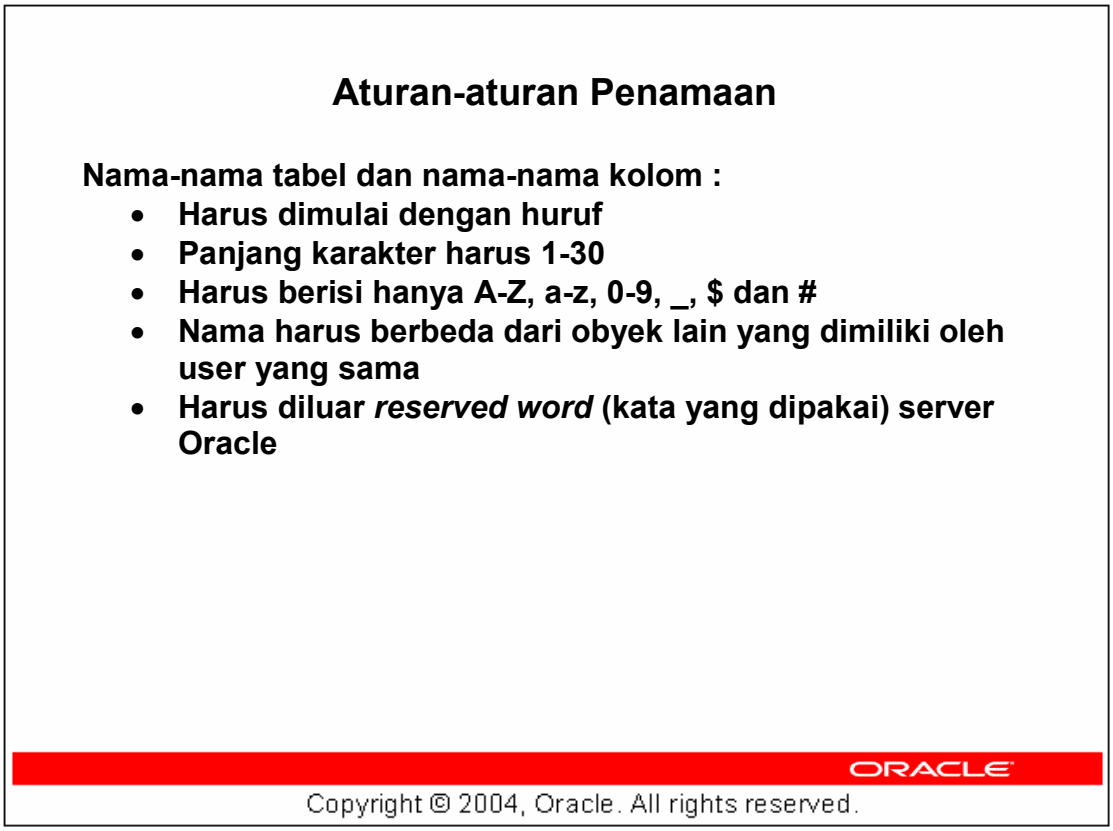
Sebuah database Oracle dapat berisi beberapa struktur data. Setiap struktur sebaiknya tersusun dalam desain database sehingga struktur itu bisa terbentuk ketika tahap pembangunan dari suatu pengembangan database.

- **Tabel** : Menyimpan data
- **View** : *Subset* data dari satu atau lebih tabel
- **Sequence** : Pembangkit nilai-nilai numerik
- **Index** : Meningkatkan performa dari beberapa query
- **Synonym** : Memberikan nama-nama alternatif untuk obyek-obyek

Struktur-Struktur Tabel Oracle

- Tabel dapat dibuat sewaktu-waktu, meskipun user-user sedang menggunakan database.
- Anda tidak perlu untuk menentukan ukuran suatu tabel. Ukuran ditentukan oleh banyaknya ruang yang dialokasikan untuk database secara keseluruhan. Ukuran suatu tabel penting, bagaimanapun, untuk memperkirakan berapa banyak ruang sebuah tabel akan digunakan sepanjang waktu.
- Struktur tabel dapat dimodifikasi secara online

Catatan : Obyek-obyek database banyak tersedia tetapi tidak dicakup dalam pelajaran ini.



Aturan-Aturan Penamaan

Anda memberi nama tabel-tabel database dan kolom berdasarkan standar aturan untuk penamaan obyek database Oracle :

- Nama tabel dan nama kolom harus dimulai dengan huruf dan panjang karakter antara 1 – 30
- Nama harus mengandung hanya karakter-karakter A-Z, a-z, 0-9, _ (*underscore*), $, dan # (karakter-karakter legal, tetapi penggunaannya tidak disarankan)
- Nama tidak boleh sama dengan nama obyek lain yang dimiliki oleh user server Oracle yang sama.
- Nama harus diluar *reserved word* (kata yang dipakai) server Oracle

Pedoman-Pedoman Penamaan
Gunakan nama yang jelas untuk tabel-tabel dan obyek database yang lain.

Catatan : Nama bersifat *case-insensitive*. Sebagai contoh, `EMPLOYEES` adalah diperlakukan sama dengan `eMPloyees` atau `eMpLOYEES`.

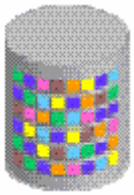
Untuk informasi lebih lanjut, lihat “Object Names and Qualifiers” dalam *Oracle Database SQL Reference*.

Pernyataan CREATE TABLE

- **Anda harus mempunyai :**
  - **CREATE TABLE privilege (hak)**
  - **Sebuah area penyimpanan**

```
CREATE TABLE [schema.]table
          (column datatype [DEFAULT expr][, ...]);
```

- **Anda menentukan :**
  - **Nama tabel**
  - **Nama kolom, tipe data kolom dan ukuran kolom**



ORACLE

Pernyataan CREATE TABLE

Anda membuat tabel untuk menyimpan data dengan mengeksekuis penyataan SQL CREATE TABLE. Pernyataan ini adalah salah satu pernyataan DDL, yang mana suatu *subset* dari pernyataan SQL untuk membuat, mengubah, atau menghapus struktur-struktur database Oracle. Pernyataan ini mempunyai efek segera pada database, dan pernyataan tersebut juga mencatat informasi dalam *data dictionary*.

Untuk membuat suatu tabel, seorang user harus mempunyai CREATE TABLE privilege dan area penyimpanan dimana obyek-obyek dibuat. Administrator database menggunakan pernyataan bahasa pengontrol data untuk memberi hak-hak kepada para user (Pernyataan DCL dibahas dalam pelajaran selanjutnya).

Dalam sintak :

schema	adalah sama dengan nama pemilik
table	adalah nama tabel
DEFAULT *expr*	adalah menentukan suatu nilai default jika suatu nilai dihilangkan dalam pernyataan INSERT
column	adalah nama kolom
datatype	adalah tipe data dan panjang kolom

Mereferensi Tabel-Tabel Milik User Lain

- **Tabel-tabel yang dimiliki oleh user lain adalah tidak berada di dalam skema milik user.**
- **Anda harus menggunakan nama pemiliknya sebagai awalan untuk tabel-tabel tersebut.**


```
SELECT *
FROM userB.employees;
```

```
SELECT *
FROM userA.employees;
```

ORACLE

Mereferensi Tabel-tabel Milik User Lain

Sebuah skema adalah kumpulan dari obyek -obyek. Obyek-obyek skema adalah struktur-struktur logika yang secara langsung menunjuk ke data dalam suatu database. Obyek – obyek skema termasuk *table*s, *views*, *synonyms*, *sequences*, *stored procedures*, *indexes*, *clusters*, dan *database links*.

Jika suatu tabel tidak dimiliki user, nama pemilik harus mengawali tabel. Sebagai contoh, jika ada skema-skema bernama `USERA` dan `USERB`, dan keduanya mempunyai tabel `EMPLOYEES`, kemudian jika `USERA` ingin mengakses tabel `EMPLOYEES` yang dimiliki `USERB`, dia harus mengawali nama tabel pada nama skema :

```
SELECT *
FROM userb.employees;
```

Jika `USERB` ingin mengakses tabel `EMPLOYEES` yang dimiliki `USERA`, dia harus mengawali nama tabel pada nama skema :

```
SELECT *
FROM usera.employees;
```

Opsi DEFAULT

- **Menentukan suatu nilai *default* untuk suatu kolom saat *insert* (penyisipan).**

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- **Nilai-nilai *literal*, ekspresi-ekspresi, atau fungsi-fungsi SQL adalah nilai-nilai legal.**
- **Kolom lain yang mempunyai nama atau *pseudocolumn* (kolom khayal) adalah nilai-nilai ilegal.**
- **Suatu tipe data *default* harus sesuai dengan tipe data kolom.**

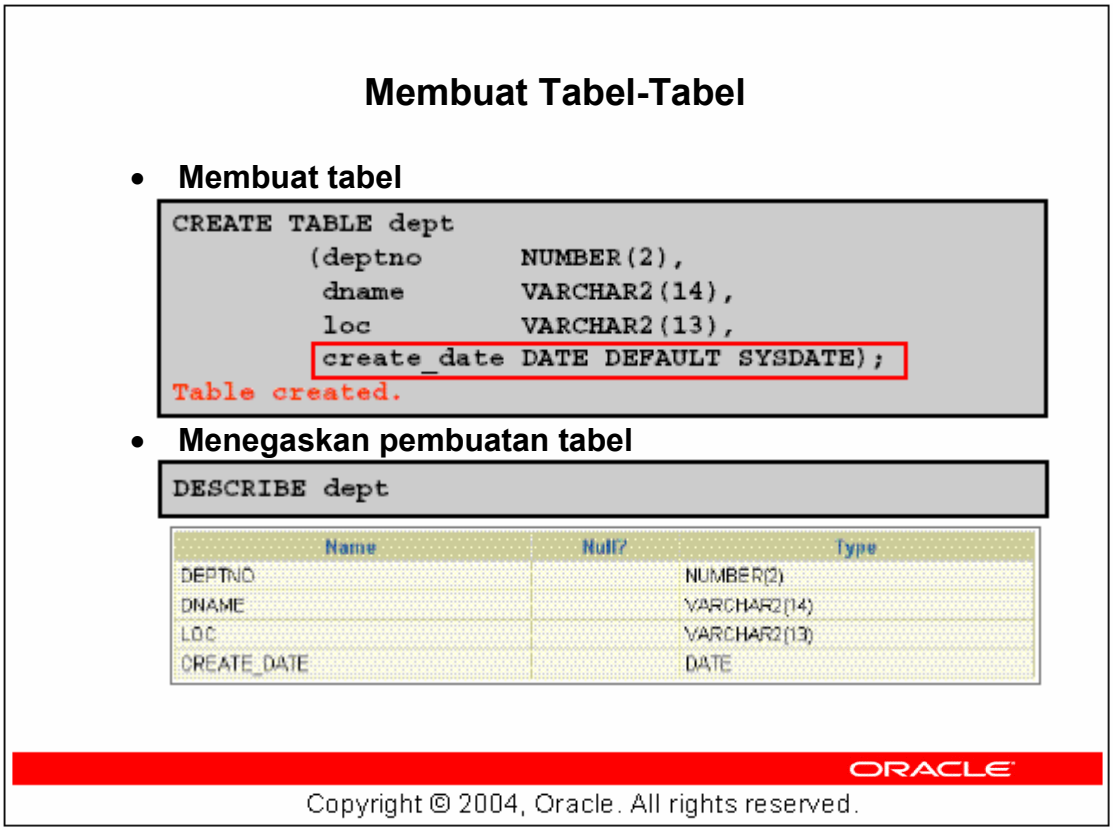
```
CREATE TABLE hire_dates
        (id          NUMBER(8),
        hire date DATE DEFAULT SYSDATE);
Table created.
```

ORACLE

Opsi DEFAULT

Ketika Anda mendefinisikan suatu tabel, Anda dapat menentukan bahwa suatu kolom diberikan suatu nilai default dengan menggunakan opsi `DEFAULT`. Opsi ini mencegah nilai-nilai *null* dari masukkan kolom jika suatu baris yang dimasukkan tanpa suatu nilai untuk kolom. Nilai *default* dapat berupa suatu literal, ekspresi, atau suatu fungsi SQL (seperti `SYSDATE` atau `USER`), tapi nilai tidak dapat berupa nama kolom lain atau *pseudocoloumn* (seperti `NEXTVAL` atau `CURRVAL`). Ekspresi *default* harus sesuai dengan tipe data kolom

Catatan : `CURRVAL` dan `NEXTVAL` dijelaskan kemudian dalam pelajaran ini.

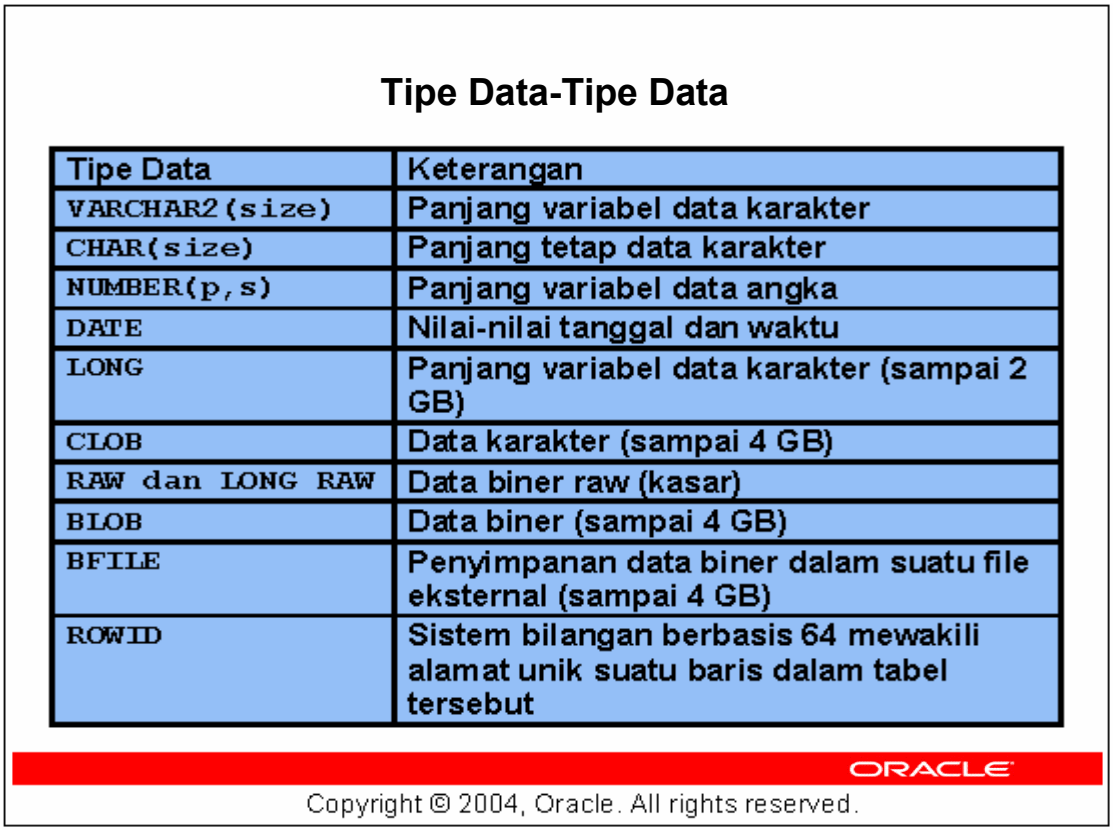


Membuat Tabel-Tabel

Contoh dalam slide membuat tabel `DEPT`, dengan empat kolom : `DEPTNO`, `DNAME`, `LOC`, dan `CREATE_DATE`. Kolom `CREATE_DATE` mempunyai nilai *default*. Jika nilai tidak disediakan untuk pernyataan `INSERT`, tanggal sistem secara otomatis dimasukkan.

Selanjutnya penegasan pembuatan tabel dengan mengeluarkan dari perintah `DESCRIBE`.

Karena membuat tabel adalah pernyataan DDL, secara otomatis *commit* mengambil bagian ketika pernyataan ini dijalankan.



Tipe Data-Tipe Data

Ketika Anda mengidentifikasi suatu kolom untuk suatu tabel, Anda perlu menyediakan suatu tipe data untuk kolom. Ada beberapa tipe data yang tersedia :

Tipe Data	Keterangan
VARCHAR2(*size*)	Panjang variable data karakter (Suatu *size* / ukuran maksimum harus ditentukan : *size* min adalah 1; *size* maks adalah 4,000)
CHAR [(*size*)]	Panjang tetap data karakter dari panjang bytes *size* (*size default* dan minimum adalah = 1; Maksimum = 2,000)
NUMBER [(*p*,*s*)]	Angka dengan presisi *p* dan skala *s* (Presisi adalah total jumlah dari digit desimal, dan skala adalah jumlah digit di kanan koma dari poin desimal; rentang presisi antara 1 – 38, dan rentang skala antara -84 – 127)
DATE	Nilai-nilai tanggal dan waktu ke detik terdekat antara 1 Januari 4712 sebelum masehi (B.C.) sampai 31 Desember 9999 masehi (A.D.).
LONG	Panjang variabel data karakter (sampai 2 GB)
CLOB	Data karakter (sampai 4 GB)
RAW (*size*)	Data biner raw (kasar) dari panjang *size* (maksimum *size* harus ditentukan : maks *size* = 2,000)
LONG RAW	Data biner raw dari panjang variabel (sampai 2 GB)
BLOB	Data biner (sampai 4 GB)
BFILE	Data biner disimpan dalam file eksternal (sampai 4 GB)
ROWID	Sistem bilangan berbasis 64 mewakili alamat unik dari suatu baris dalam tabel tersebut

Petunjuk

- Kolom LONG tidak disalin ketika tabel dibuat menggunakan suatu *subquery*.
- Kolom LONG tidak dapat dimasukkan dalam suatu klausa GROUP BY atau ORDER BY.
- Hanya satu kolom LONG yang dapat digunakan pada setiap tabel.
- Tidak ada *constraint-constraint* yang dapat didefinisikan pada kolom LONG.
- Anda sebaiknya menggunakan kolom CLOB daripada kolom LONG.

Tipe Data-Tipe Data *Datetime*

Anda dapat menggunakan beberapa tipe data-tipe data *datetime* :

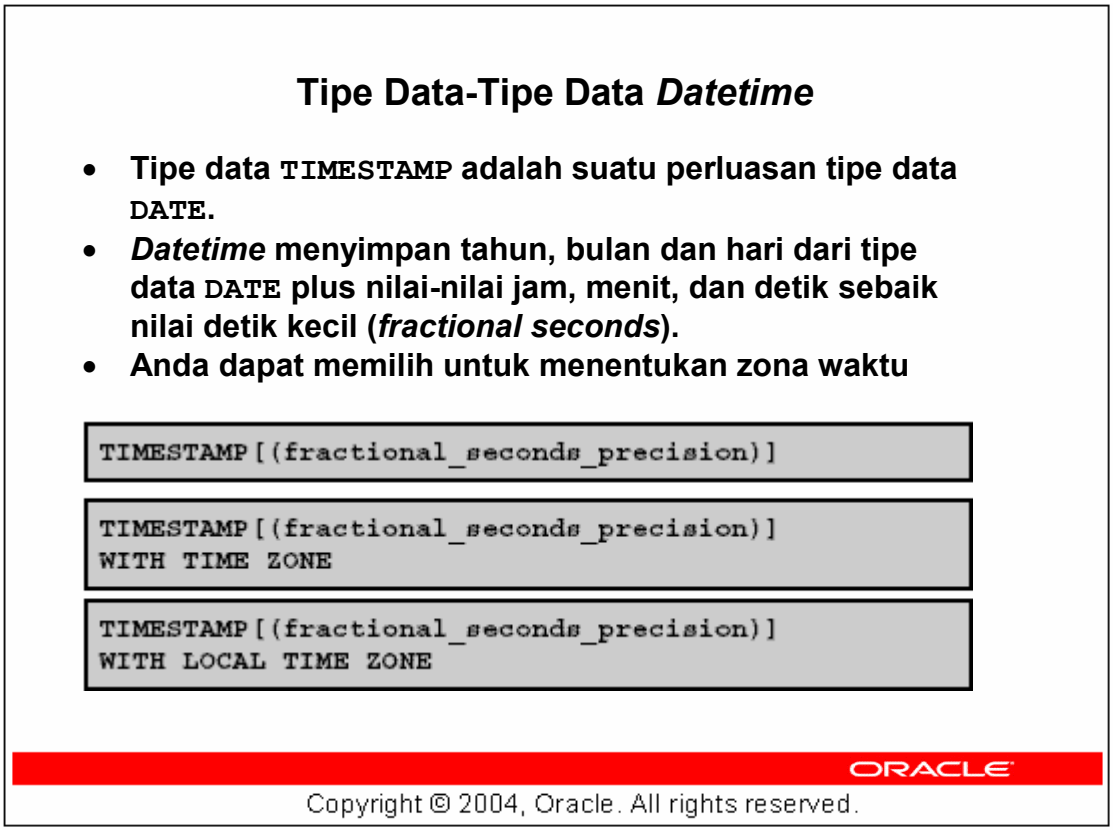
Tipe Data	Keterangan
TIMESTAMP	Tanggal dengan pembagian detik
INTERVAL YEAR TO MONTH	Disimpan sebagai suatu interval tahun dan bulan
INTERVAL DAY TO SECOND	Dsimpan sebagai suatu interval dari hari, jam, menit dan detik

ORACLE

Tipe Data-Tipe Data Datetime yang Lain

Tipe Data	Penjelasan
`TIMESTAMP`	Memungkinkan waktu disimpan sebagai tanggal dengan sebagian detik. Ada beberapa variasi pada tipe data ini.
`INTERVAL YEAR TO MONTH`	Memungkinkan waktu disimpan sebagai interval tahun dan bulan. Digunakan untuk menunjukkan perbedaan antara dua nilai *datetime* yang mana hanya bagian-bagian penting tahun dan bulan.
`INTERVAL DAY TO SECOND`	Memungkinkan waktu yang disimpan sebagai suatu interval hari, jam, menit dan detik. Digunakan untuk menunjukkan perbedaan yang tepat antara dua nilai-nilai *datetime.*

Catatan : Tipe data-tipe data datetime ini ada pada Oracle9*i* dan keluaran selanjutnya. Untuk informasi lebih rinci tentang tipe data *datetime*, lihat topic-topik "`TIMESTAMP` Datatype," "`INTERVAL YEAR TO MONTH` Datatype", dan "`INTERVAL DAY TO SECOND` Datatype" pada *Oracle SQL Reference.*



Tipe Data TIMESTAMP

Tipe data TIMESTAMP adalah suatu perluasan dari tipe data DATE. TIMESTAMP menyimpan nilai-nilai tahun, bulan, dan hari dari tipe data DATE plus nilai-nilai jam, menit dan detik. Tipe data ini digunakan untuk menyimpan nilai-nilai waktu yang tepat.

Secara optional fractional_second_precision menentukan jumlah digit dalam sebagian kecil dari *field datetime* SECOND dan bisa berupa suatu bilangan antara 0 sampai 9. *Default*-nya adalah 6.

Contoh

Dalam contoh ini, suatu tabel diberi nama NEW_EMPLOYEES, dengan suatu kolom START_DATE yang mempunyai suatu tipe data TIMESTAMP :

```
CREATE TABLE new_employees
          (employee_id NUMBER,
          first_name VARCHAR2(15),
          last_name VARCHAR2(15),
          . . .
          start_date TIMESTAMP(7),
          . . . );
```

Anggap bahwa dua baris disisipkan dalam tabel NEW_EMPLOYEES. Keluaran yang ditampilkan menunjukkan perbedaan. (Suatu *default* tipe data DATE untuk tampilan menggunakan format DD-MON-RR):

```
SELECT start_date
FROM new_employees;

17-JUN-03 12.00.00.000000 AM
21-SEP-03 12.00.00.000000 AM
```

Tipe Data `TIMESTAMP` (lanjutan)

Tipe Data `TIMESTAMP WITH TIME ZONE`

`TIMESTAMP WITH TIME ZONE` adalah suatu varian dari `TIMESTAMP` yang mencakup suatu perbedaan wilayah waktu (*time-zone*) dalam nilainya. Perbedaan wilayah waktu adalah perbedaan (dalam jam dan menit) antara waktu setempat dan dan UTC (*Universal Time Coordinate*, yang dikenal sebagai waktu *Greenwich Mean Time*). Tipe data ini digunakan untuk mengumpulkan dan memeriksa informasi tanggal lintas wilayah geografis.

Sebagai contoh:

```
TIMESTAMP '2003-04-15 8:00:00 -8:00'
```

adalah sama dengan

```
TIMESTAMP '2003-04-15 11:00:00 -5:00'
```

Karena pada saat pukul 8:00 a.m. di *Pacific Standard Time* adalah menunjukkan pukul 11:00 a.m. di *Eastern Standard Time*.

Hal ini dapat juga dinyatakan sebagai berikut:

```
TIMESTAMP '2003-04-15 8:00:00 US/Pacific'
```

Tipe Data `TIMESTAMP WITH LOCAL TIME ZONE`

`TIMESTAMP WITH LOCAL TIME ZONE` adalah varian lain dari `TIMESTAMP` yang mencakup suatu perbedaan wilayah waktu dalam nilainya. `TIMESTAMP WITH LOCAL TIME ZONE` berbeda dengan `TIMESTAMP WITH TIME ZONE` di dalam data yang disimpan dalam database disesuaikan dengan wilayah waktu database, dan perbdaan wilayah waktu tidak disimpan sebagai bagian dari data kolom. Saat user me-*retrieve* data, data yang dikembalikan dalam bagian wilayah waktu user berada.Perbedaan wilayah waktu adalah perbedaan (dalam jam dan menit) antara waktu setempat dan UTC.

Tidak seperti `TIMESTAMP WITH TIME ZONE`, Anda dapat menentukan tipe kolom-kolom dari `TIMESTAMP WITH LOCAL TIME ZONE` sebagai bagian dari suatu *primary* atau *unique key*, seperti contoh berikut :

```
CREATE TABLE time_example
          (order_date TIMESTAMP WITH LOCAL TIME ZONE);

INSERT INTO time_example VALUES ('15-JAN-04 09:34:28 AM');

SELECT *
FROM   time_example;

ORDER_DATE
---------------------------------------
15-JAN-04 09.34.28.000000 AM
```

Tipe `TIMESTAMP WITH LOCAL TIME ZONE` adalah cocok untuk aplikasi-aplikasi *two-tier* dimana Anda ingin menampilkan tanggal dan waktu menggunakan wilayah waktu dari sistem di client.

Tipe Data-Tipe Data *Datetime*

- **Tipe Data `INTERVAL YEAR TO MONTH` menyimpan suatu periode waktu menggunakan *field-field datetime* `YEAR` dan `MONTH` :**

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

- **Tipe data `INTERVAL DAY TO SECOND` menyimpan suatu periode dari waktu dengan masa hari, jam, menit, dan detik :**

```
INTERVAL DAY [(day_precision)]
   TO SECOND [(fractional_seconds_precision)]
```


Tipe Data `INTERVAL YEAR TO MONTH`

`INTERVAL YEAR TO MONTH` menyimpan suatu periode waktu menggunakan *field-field datetime* `YEAR` dan `MONTH`.

Gunakan `INTERVAL YEAR TO MONTH` untuk menunjukkan perbedaan antara nilai-nilai dua *datetime,* dimana hanya bagian-bagian penting tahun dan bulan saja. Sebagai contoh, Anda mungkin menggunakan nilai ini untuk mengatur suatu peringatan pada suatu tanggal 120 bulan ke depan, atau memeriksa apakah 6 bulan yang berlalu dari suatu tanggal tertentu.

Dalam sintak:

`year_precision` adalah jumlah digit dalam *field datetime* `YEAR`. Nilai *default* dari `year_precision` adalah 2.

Contoh

- `INTERVAL '123-2' YEAR(3) TO MONTH`
  Menandakan suatu interval dari tahun 123, bulan 2
- `INTERVAL '123-2' YEAR (3)`
  Menandakan suatu interval dari tahun 123 bulan 0
- `INTERVAL '300' MONTH (3)`
  Menandakan suatu interval dari bulan 300
- `INTERVAL '123' YEAR`
  Mengembalikan suatu kesalahan sebab nilai tepat *default* adalah 2 dan 123 mempunyai 3 digit

```
CREATE TABLE time_example2
(loan_duration INTERVAL YEAR (3) TO MONTH);

INSERT INTO time example2 (loan_duration)
  VALUES (INTERVAL '120' MONTH(3));

SELECT TO_CHAR(sysdate+loan_duration, 'dd-mon-yyyy')
FROM time_example2;    --today's date is 26-Sep-2001
```

TO_CHARSYS
26-sep-2011

Tipe Data `INTERVAL DAY TO SECOND`

`INTERVAL DAY TO SECOND` menyimpan suatu periode waktu dengan masa hari, jam, menit dan detik.

Gunakan `INTERVAL DAY TO SECOND` untuk menunjukkan perbedaan yang tepat antara nilai-nilai dua *datetimes*. Sebagai contoh, Anda mungkin menggunakan nilai ini untuk mengatur suatu peringatan pada suatu waktu 36 jam ke depan, atau untuk mencatat waktu antara dimulai dan berakhirnya suatu perlobaan. Untuk menunjukkan lamanya waktu yang dihabiskan, termasuk tahun yang berbeda, dengan ketepatan yang tinggi, Anda bisa menggunakan suatu nilai yang besar untuk bagian-bagian hari.

Dalam sintak:

`day_precision`	adalah jumlah digit dalam *field datetime* `DAY`. Nilai-Nilai yang diterima adalah 0 sampai 9. *Default*-nya adalah 2
`fractional_seconds_precision`	adalah jumlah digit dalam sebagian kecil dari *filed datetime* `SECOND`. Nilai-Nilai yang diterima adalah 0 sampai 9. *Default*-nya adalah 6.

Contoh

- `INTERVAL '4 5:12:10.222' DAY TO SECOND(3)`
  Menandakan 4 hari, 5 jam, 12 menit, 10 detik dan 222 ribu dari suatu detik.
- `INTERVAL '180' DAY(3)`
  Menandakan 180 hari
- `INTERVAL '4 5:12:10.222' DAY TO SECOND(3)`
  Menandakan 4 hari, 5 jam, 12 menit, 10 detik dan 222 ribu dari suatu detik
- `INTERVAL '4 5:12' DAY TO MINUTE`
  Menandakan 4 hari, 5 jam dan 12 menit
- `INTERVAL '400 5' DAY(3) TO HOUR`
  Menandakan 400 hari dan 5 jam.
- `INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)`
  Menandakan 11 jam, 12 menit dan 10.2222222 detik

Contoh

```
CREATE TABLE time_example3
(day_duration INTERVAL DAY (3) TO SECOND);

INSERT INTO time_example3 (day_duration)
VALUES (INTERVAL '180' DAY(3));

SELECT sysdate + day_duration "Half Year"
FROM time_example3;    --today's date is 26-Sep-2001
```

Half Year
25-MAR-02

Menyertakan *Constraint-Constraint*

- ***Constraint-constraint*** **menjalankan aturan-aturan (*rules*)pada tingkat tabel.**
- ***Constraint-constraint*** **mencegah penghapusan suatu tabel jika ada ketergantungan-ketergantungan (*dependencies*).**
- **Berikut ini adalah tipe-tipe *Constraint* yang valid :**
  - **NOT NULL**
  - **UNIQUE**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**

Constraint-Constraint

Server Oracle menggunakan *constraint-constraint* untuk mencegah masukan data tidak valid ke dalam tabel-tabel.

Anda dapat menggunakan *constraint-constraint* untuk melakukan berikut ini:

- Melaksanakan aturan pada data dalam suatu tabel kapan pun suatu baris disisipkan, di-*update*, atau dihapus dari tabel. *Constraint* harus dipenuhi pada operasi agar berhasil.
- Mencegah penghapusan suatu tabel jika ada ketergantungan dari tabel lain.
- Menyediakan aturan-aturan untuk *tool-tool* Oracle, seperti Oracle Developer.

Data Integrity Constraints

Constraint	Keterangan
NOT NULL	Menentukan bahwa kolom tidak boleh berisi suatu nilai *null*
UNIQUE	Menentukan suatu kolom atau kombinasi kolom-kolom yang nilai-nilainya harus unik untuk semua bari di dalam tabel
PRIMARY KEY	Identifikasi secara unik setiap baris pada tabel
FOREIGN KEY	Menentukan dan melaksanakan suatu hubungan kunci tamu (*foreign key*) antara suatu kolom dan kolom yang direferensikan
CHECK	Menentukan suatu kondisi yang harus benar

Pedoman-Pedoman *Constraint*

- **Anda dapat menamai suatu *constraint*, atau server Oracle sendiri memberikan suatu nama dengan menggunakan format `SYS_Cn`.**
- **Buat suatu *constraint* pada waktu berikut ini :**
  - **Pada waktu yang sama ketika tabel dibuat**
  - **Setelah tabel telah dibuat**
- **Definiskan suatu *constraint* di tingkat tabel atau kolom.**
- **Tampilan suatu *constraint* di dalam *data dictionary*.**

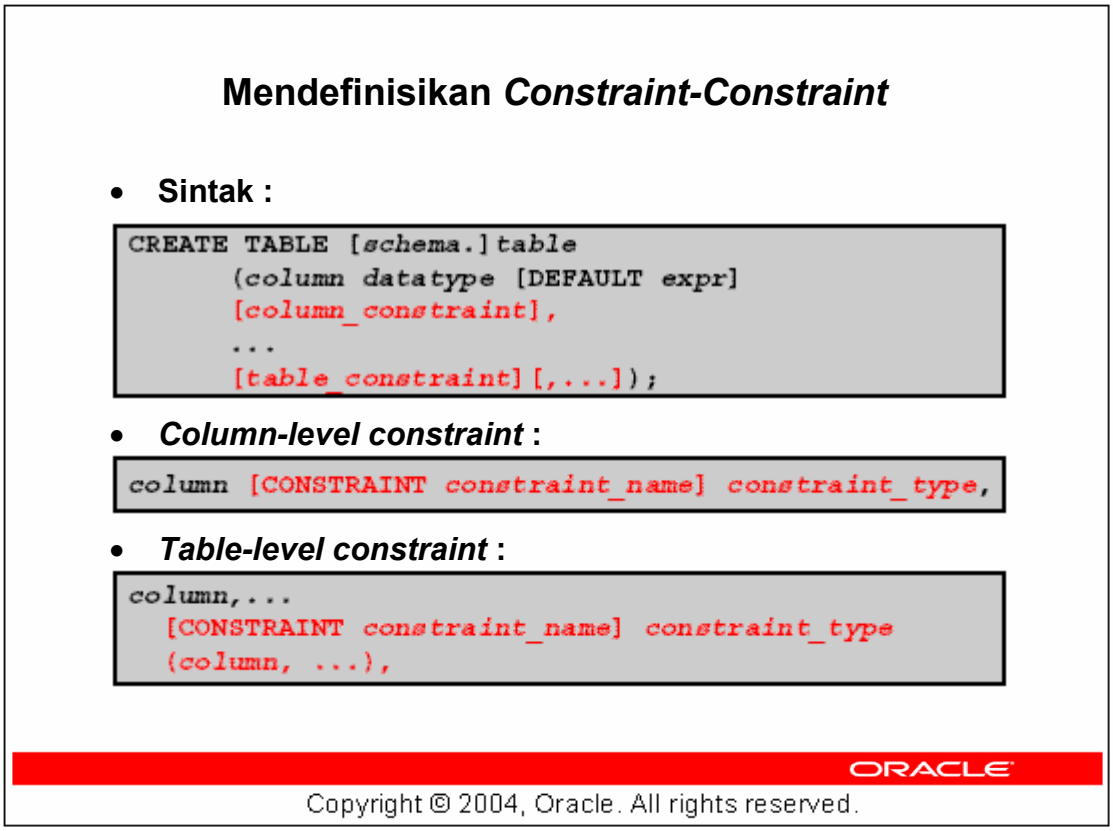
ORACLE

Pedoman-Pedoman *Constraint*

Semua *constraint* disimpan didalam *data dictionary*. *Constraint-constraint* mudah untuk dikenali jika Anda memberikan suatu nama yang memiliki arti. Nama-nama *constraint* harus mengikuti aturan-aturan standar *object-naming*. Jika Anda tidak memberikan nama *constraint* Anda, server Oracle akan memberikan suatu nama dengan format `SYS_Cn`, di mana `n` adalah suatu bilangan bulat (*integer*) sehingga nama *constraint* adalah nama yang unik.

Constraint-constraint dapat didefiniskan pada saat pembuatan tabel atau setelah tabel selesai dibuat.

Untuk informasi lebih lanjut, lihat "Constraint" pada *Oracle Database SQL Reference.*



Mendefinisikan *Constraint-Constraint*

Pada slide diberikan sintak untuk mendefinisikan *constraint-constraint* ketika membuat suatu tabel. Anda dapat membuat *constraint-constraint* baik pada tingkat kolom (*column level*) atau tingkat tabel (*table level*). *Constraint-constraint* yang didefinisikan pada tingkat kolom disertakan pada saat kolom didefiniskan. *Table-Level constraint* didefiniskan pada bagian akhir dari pendefinisian tabel dan harus merujuk ke kolom atau kolom-kolom dimana *constraint* disebutkan didalam suatu kelompok kurung.

Constraint-constraint `NOT NULL` harus didefiniskan pada tingkat kolom.

Constraint-constraint yang diterapkan pada lebih dari satu kolom harus didefinisikan pada tingkat tabel.

Dalam sintak :

`schema`	adalah sama seperti nama pemilik
`table`	adalah nama tabel
`DEFAULT expr`	menentukan suatu nilai *default* untuk digunakan jika suatu nilai dihilangkan dalam pernyataan `INSERT`
`column`	adalah nama kolom
`datatype`	adalah tipe data untuk kolom dan panjangnya
`column_constraint`	adalah suatu *integrity constraint* sebagai bagian dari definisi kolom
`table_constraint`	adalah suatu *integrity constraint* sebagai bagian dari definisi tabel

Mendefiniskan *Constraint-Constraint*

- ***Column-level constraint*** :

```
CREATE TABLE employees(
  employee_id  NUMBER(6)
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,
  first_name   VARCHAR2(20),
  ...);
```

1

- ***Table-level constraint*** :

```
CREATE TABLE employees(
  employee_id  NUMBER(6),
  first_name   VARCHAR2(20),
  ...
  job_id       VARCHAR2(10) NOT NULL,
  CONSTRAINT emp_emp_id_pk
    PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

Mendefiniskan *Constraint-Constraint* (lanjutan)

Constraint-constraint biasanya dibuat pada waktu yang sama saat tabel dibuat. *Constraint-constraint* bisa ditambahkan pada suatu tabel setelah tabel dibuat dan untuk sementara bisa tidak digunakan.

Kedua contoh pada slide membuat suatu *constraint primary key* pada kolom `EMPLOYEE_ID` dari tabel `EMPLOYEES`.

1. Contoh pertama menggunakan sintak *column-level* untuk mendefiniskan constraint.
2. Contoh kedua menggunakan sintak *table-level* untuk mendefinisikan constraint.

Lebih detil tentang *constraint primary key* dibahas pada pelajaran berikutnya.

Constraint NOT NULL

Memastikan bahwa nilai-nilai *null* tidak diijinkan pada kolom :

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

. . .
20 rows selected.

Constraint NOT NULL (Baris tidak bisa berisi suatu nilai *null* untuk kolom ini.)

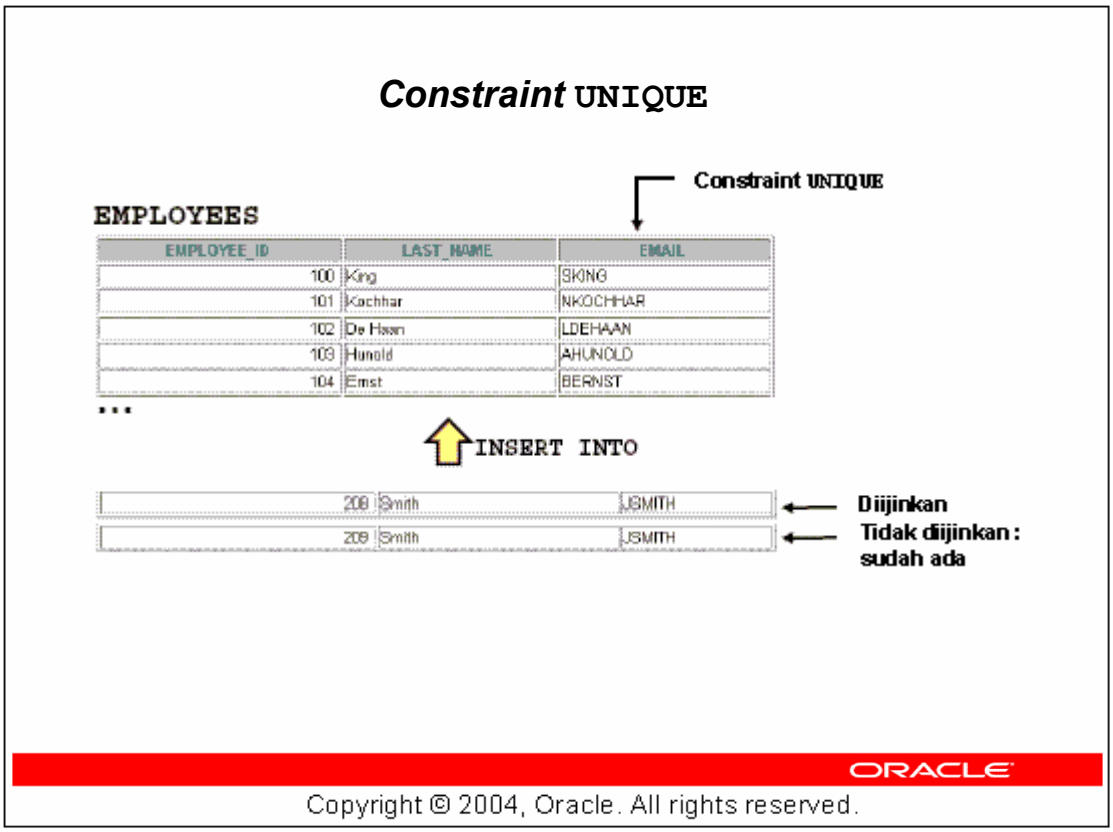
Constraint NOT NULL

Tidak adanya constraint NOT NULL (Beberapa baris dapat berisi suatu nilai *null* untuk kolom ini.)

ORACLE

Constraint NOT NULL

Constraint `NOT NULL` memastikan bahwa kolom tidak berisi nilai-nilai *null*. Kolom-kolom tanpa *constraint* `NOT NULL` secara default dapat berisi nilai-nilai *null*. *Constraint* `NOT NULL` harus didefinisikan pada tingkat kolom (*column-level*).



Constraint UNIQUE

Suatu kunci *integrity constraint* `UNIQUE` meminta bahwa nilai dalam suatu kolom atau kelompok kolom (kunci) adalah unik, dimana tidak ada dua baris dari suatu tabel bisa memiliki nilai-nilai yang sama dalam suatu kolom tertentu atau sekelompok kolom. Kolom (atau sekelompok kolom) disertakan dalam definisi dari *constraint* kunci `UNIQUE` yang disebut *unique key*. Jika constraint `UNIQUE` terdiri dari lebih satu kolom, kelompok kolom-kolom itu disebut suatu *composite unique key*.

Constraint `UNIQUE` memperbolehkan inputan nilai *null* kecuali Anda juga mendefinisikan *constraint-constraint* `NOT NULL` pada kolom yang sama. Pada kenyataannya, setiap nomor dari baris-baris bisa menyertakan *null-null* pada kolomnya tanpa constraint `NOT NULL` karena *null-null* dianggap tidak sama dengan apapun. Sebuah *null* dalam suatu kolom (atau dalam semua kolom dari suatu *composite* `UNIQUE` *key*) selalu memenuhi suatu *constraint* `UNIQUE`.

Catatan : Karena mekanisme pencarian untuk *constraint* `UNIQUE` pada lebih dari satu kolom, Anda tidak dapat memiliki nilai-nilai yang sama pada kolom-kolom non-null dari suatu *constraint composite* `UNIQUE` *key* null secara parsial.

Constraint UNIQUE

Ditentukan baik pada level tabel atau level kolom :

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(9,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

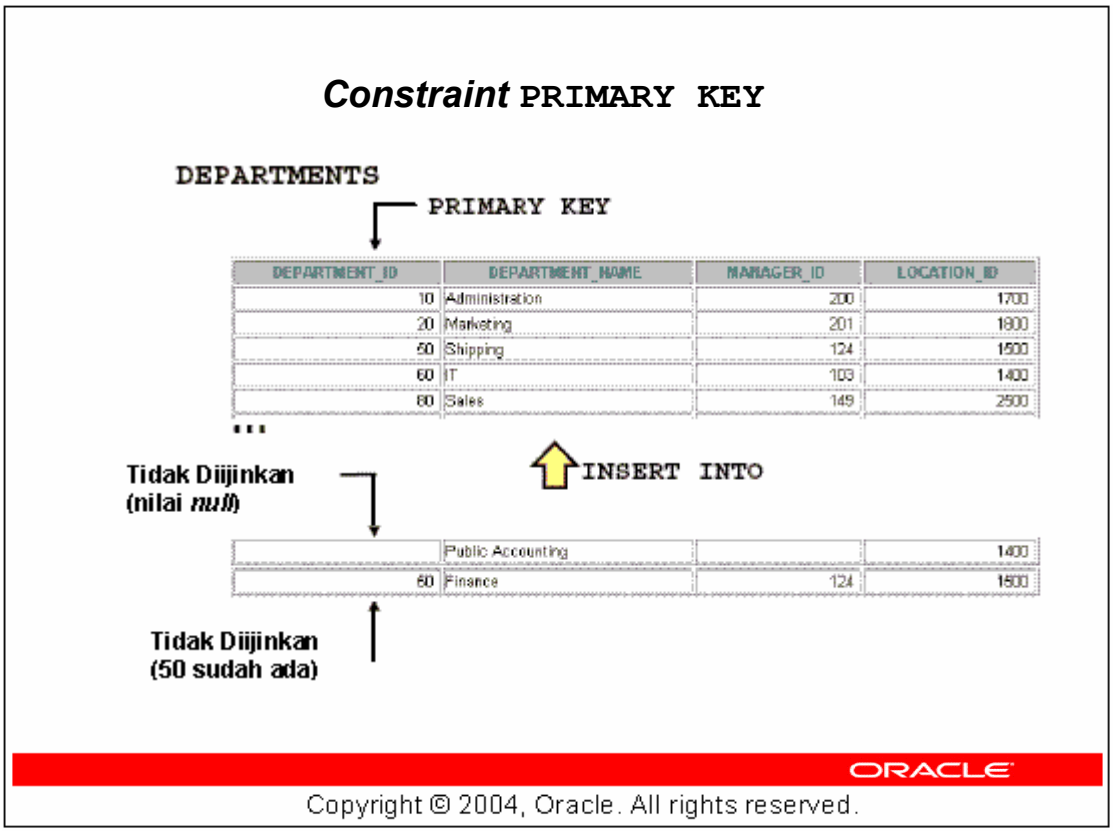
ORACLE

Constraint UNIQUE (lanjutan)

Constraint-constraint UNIQUE dapat didefinisikan pada tingkat kolom atau tingkat tabel. Suatu *composite unique key* dibuat dengan menggunakan definisi tingkat tabel.

Contoh pada silde menerapkan *constraint* `UNIQUE` pada kolom `EMAIL` dari tabel `EMPLOYEES`. Nama constraintnya ialah `EMP_EMAIL_UK`.

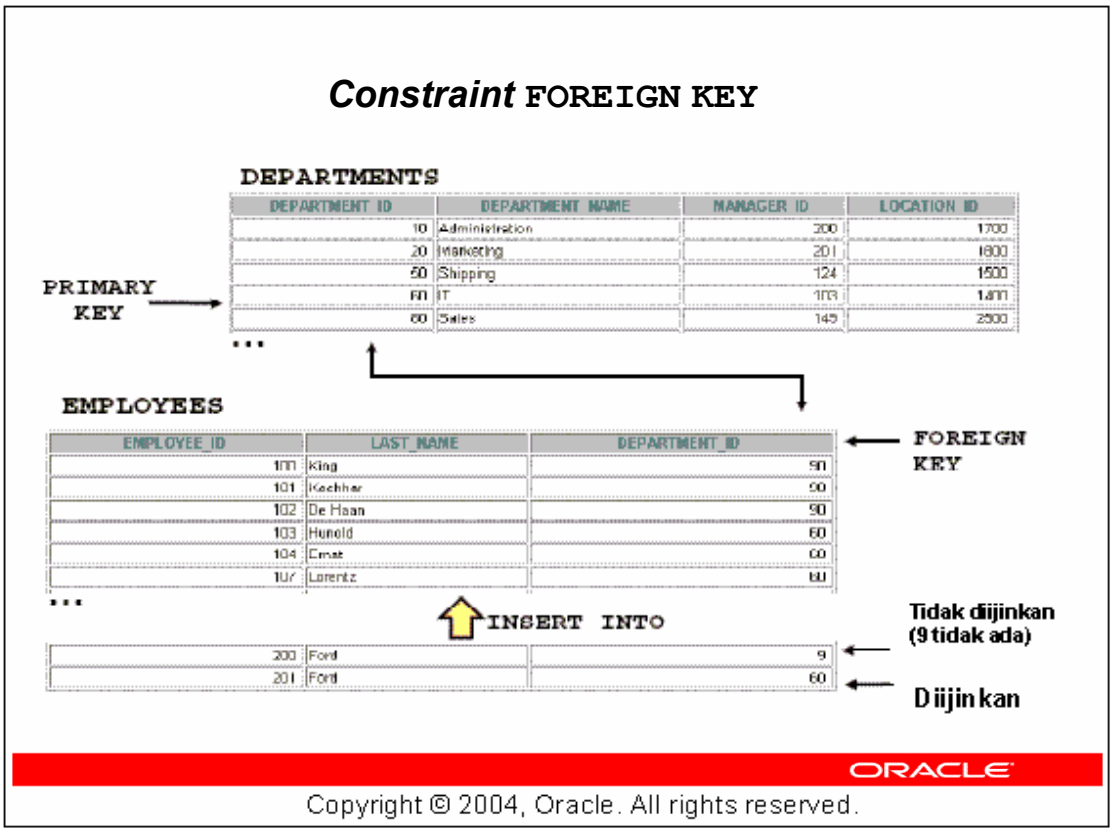
Catatan : Server Oracle menjalankan *constraint* `UNIQUE` dengan membuat secara implisit suatu indeks unik pada kolom kunci unik (*unique key column*) atau kolom-kolom.



Constraint `PRIMARY KEY`

Suatu *constraint* `PRIMARY KEY` membuat suatu *primary key* untuk suatu tabel. Setiap tabel hanya bisa dibuat satu primary key. *Constraint* `PRIMARY KEY` adalah suatu kolom atau sekelompok kolom yang teridentifikasi secara unik pada setiap baris dalam suatu tabel. Constraint menjalankan keunikan dari suatu kolom atau kombinasi kolom dan memastikan tidak ada kolom yang merupakan bagian dari *primary key* dapat berisi nilai *null*.

Catatan : Karena keunikan adalah bagian dari definisi *constraint primary key*, server Oracle menjalankan keunikan dengan membuat secara implisit suatu indeks unik pada kolom atau kolom-kolom *primary key*.



Constraint FOREIGN KEY

Suatu *constraint* FOREIGN KEY (atau *referential integrity*) mendesain suatu kolom atau kombinasi dari kolom-kolom sebagai suatu *foreign key* (kunci tamu) dan menjalankan sebuah hubungan antara *primary key* atau suatu *unique key* dalam tabel yang sama atau pada tabel yang berbeda.

Contoh pada slide ,DEPARTMENT_ID telah didefinisikan sebagai *foreign key* dalam tabel EMPLOYEES (tabel *dependent* atau *child*); *foreign key* mereferensikan kolom DEPARTMENT_ID dari tabel DEPARTMENT (tabel *referenced* atau *parent*).

Pedoman-Pedoman

- Suatu nilai *foreign key* harus sesuai dengan nilai yang ada dalam tabel *parent* (induk) atau menjadi NULL.
- *Foreign key-foreign key* adalah berdasarkan pada nilai-nilai data dan betul-betul bersifat logika, daripada fisik, berupa *pointers* (penunjuk).

Constraint FOREIGN KEY

Ditentukan baik pada tingkat tabel atau tingkat kolom :

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

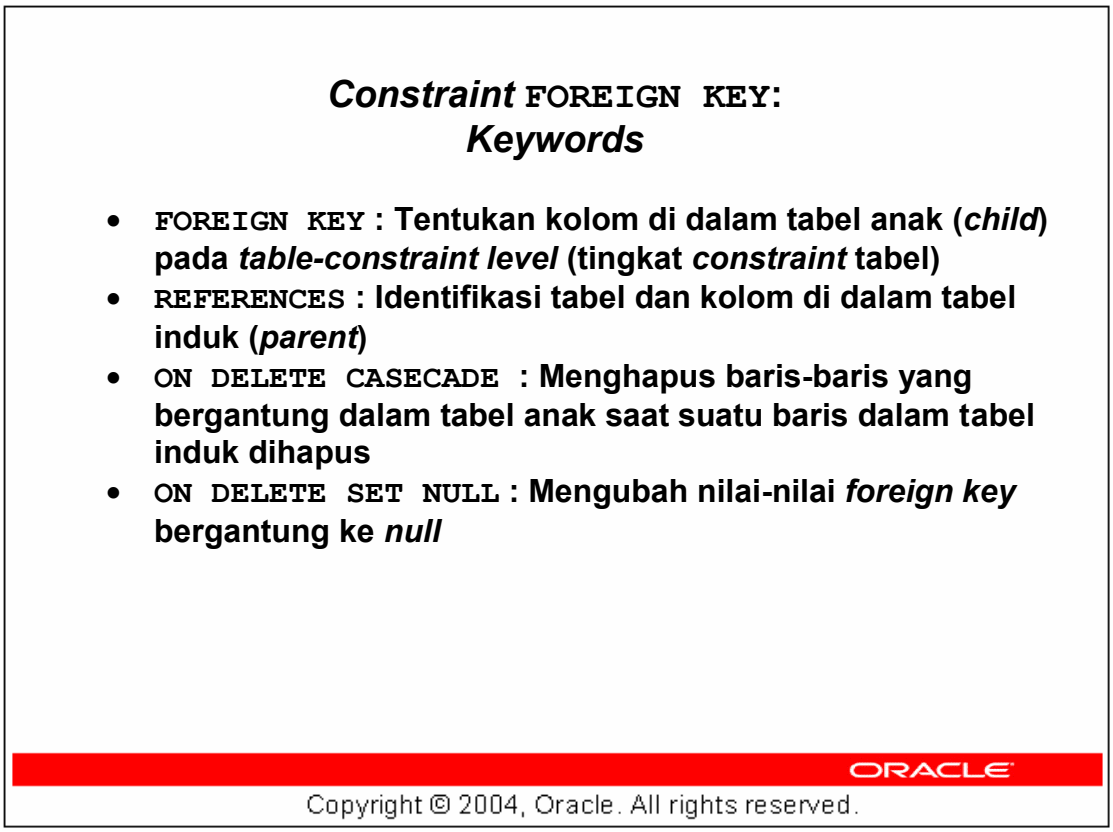
Constraint FOREIGN KEY (lanjutan)

Constraint-constraint FOREIGN KEY dapat didefinisikan pada tingkat *constraint* kolom atau tabel. Suatu *composite* (gabungan) *foreign key* harus dibuat dengan menggunakan definisi *level-table* (tingkat tabel).

Contoh pada slide mendefinisikan suatu *constraint* FOREIGN KEY pada kolom DEPARTMENT_ID dari tabel EMPLOYEES, menggunakan sintak *table-level*. Nama dari *constraint* itu ialah EMP_DEPTID_FK.

Foreign key juga dapat didefinisikan pada tingkat kolom, sediakan *constraint* berdasarkan pada suatu kolom tunggal. Perbedaan sintak dalam *keywords* FOREIGN KEY tidak muncul. Sebagai contoh :

```
CREATE TABLE employees
(. . .
 department_id NUMBER(4) CONSTRAINT emp_deptid_fk
 REFERENCES departments(department_id),
 . . .
 )
```



Constraint `FOREIGN KEY` : Keywords

Foreign key didefinisikan pada tabel anak, dan tabel yang berisi kolom yang direferensikan merupakan tabel induk. *Foreign key* didefinisikan menggunakan suatu kombinasi dari *keywords* (kata kunci) berikut :

- `FOREIGN KEY` digunakan untuk mendefinisikan kolom dalam tabel anak pada *table-constraint level* (tingkat *constraint* tabel).
- `REFERENCES` mengidentifikasikan tabel dan kolom pada tabel induk.
- `ON DELETE CASCADE` menandakan bahwa ketika baris dalam tabel induk dihapus , baris-baris yang bergantung pada tabel anak juga akan dihapus.
- `ON DELETE SET NULL` mengubah nilai-nilai *foreign key* ke null ketika nilai induk di hapus.

Perilaku *default* ini disebut *restrict rule*, dimana tidak mengijinkan *update* atau penghapusan dari data yang direferensikan.

Tanpa pilihan the `ON DELETE CASCADE` atau `ON DELETE SET NULL`, baris dalam tabel induk tidak bisa dihapus jika tabel induk direferensikan dalam tabel anak.

Constraint CHECK

- **Tentukan suatu kondisi bahwa setiap baris harus sesuai**
- **Ekspresi-ekspresi berikut adalah tidak diijinkan :**
  - **Merujuk pada *pseudocolumns* (kolom maya) CURRVAL, NEXTVAL, LEVEL dan ROWNUM**
  - **Memanggil ke fungsi-fungsi SYSDATE, UID, USER dan USERENV**
  - **Query-query yang merujuk ke nilai-nilai lain pada baris-baris lain**

```
..., salary   NUMBER(2)
     CONSTRAINT emp_salary_min
            CHECK (salary > 0),...
```

ORACLE

Constraint CHECK

Constraint CHECK mendefinisikan suatu kondisi bahwa setiap baris harus sesuai. Kondisi dapat menggunakan konstruksi yang sama sebagai kondisi-kondisi query, dengan pengecualian sebagai berikut :

- Merujuk pada pseudocolumns (kolom maya) CURRVAL, NEXTVAL, LEVEL dan ROWNUM
- Memanggil fungsi-fungsi SYSDATE, UID, USER dan USERRENV
- Query-query yang merujuk pada nilai-nilai lainnya pada baris yang lain

Suatu kolom tunggal dapat memiliki beberapa *constraint* CHECK yang merujuk pada kolom dalam definisinya. Tidak ada batas jumlah *constraint* CHECK yang dapat Anda definisikan pada suatu kolom.

Constraint-constraint CHECK dapat didefinsisikan pada tingkat kolom ataupun tingkat tabel .

```
CREATE TABLE employees
  (. . .
   salary NUMBER(8,2) CONSTRAINT emp_salary_min
                      CHECK (salary > 0),
   . . .
```

CREATE TABLE : Contoh

```
CREATE TABLE employees
    ( employee_id    NUMBER(6)
        CONSTRAINT     emp_employee_id   PRIMARY KEY
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
        CONSTRAINT     emp_last_name_nn  NOT NULL
    , email          VARCHAR2(25)
        CONSTRAINT     emp_email_nn      NOT NULL
        CONSTRAINT     emp_email_uk      UNIQUE
    , phone_number   VARCHAR2(20)
    , hire_date      DATE
        CONSTRAINT     emp_hire_date_nn  NOT NULL
    , job_id         VARCHAR2(10)
        CONSTRAINT     emp_job_nn        NOT NULL
    , salary         NUMBER(8,2)
        CONSTRAINT     emp_salary_ck     CHECK (salary>0)
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
        CONSTRAINT     emp_dept_fk       REFERENCES
           departments (department_id));
```

ORACLE

Contoh CREATE TABLE

Pada contoh ditunjukkan penggunaan pernyataan untuk membuat tabel `EMPLOYEES` dalam skema **HR**.

Constraint-Constraint yang Melanggar

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
UPDATE employees
       *
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

Departemen 55 tidak ada.

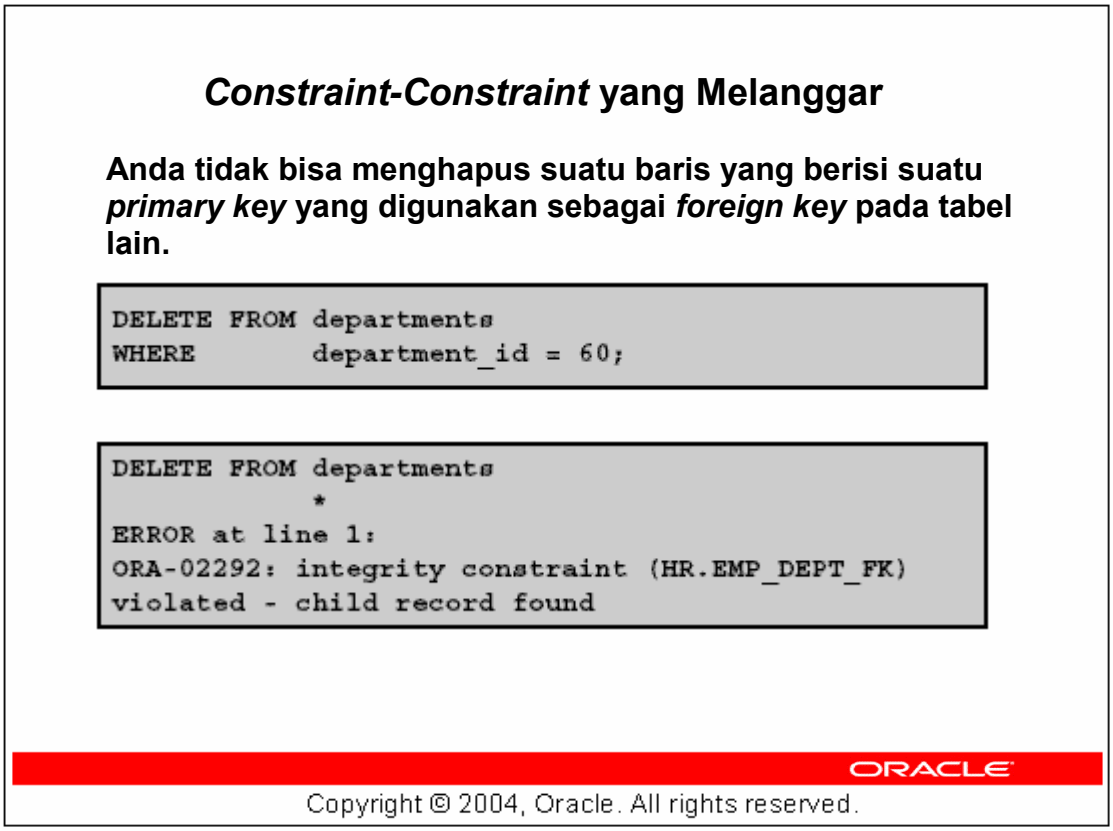
ORACLE

Kesalahan *Intregity Constraint*

Ketika Anda sudah meletakkan *constraint-constraint* pada kolom-kolom, sebuah *error* akan dikembalikan kepada Anda jika Anda mencoba untuk melanggar aturan *constraint*.

Sebagai contoh, jika Anda berusaha untuk meng-*update* suatu catatan dengan suatu nilai yang terkait pada suatu *integrity constraint* , maka menghasilkan *error*.

Contoh pada slide, departemen 55 tidak ada pada tabel induk, `DEPARTMENTS`, dan Anda juga menerima pesan pelanggaran *parent key* `ORA-02291`.



Kesalahan *Integrity Constraint* (lanjutan)

Jika anda mencoba untuk menghapus suatu *record* pada suatu nilai yang terkait pada suatu *integrity constraint*, suatu kesalahan akan dikembalikan.

Contoh pada slide mencoba untuk menghapus departemen 60 dari tabel `DEPARTMENTS`, tapi hasilnya suatu kesalahan karena nomor departemen itu digunakan sebagai *foreign key* dalam tabel `EMPLOYEES`. Jika *record* induk yang Anda coba untuk hapus memiliki *record* anak, maka Anda menerima pesan pelanggaran *child record found* `ORA-02292`.

Pernyataan berikut dikerjakan karena tidak ada pegawai-pegawai pada departemen 70 :

```
DELETE FROM departments
WHERE       department_id = 70;

1 row deleted.
```

Membuat suatu Tabel dengan Menggunakan suatu *Subquery*

- **Buat suatu tabel dan sisipkan baris-baris dengan mengkombinasikan pernyataan `CREATE TABLE` dan pilihan `AS subquery.`**

```
CREATE TABLE table
            [(column, column...)]
AS subquery;
```

- **Sesuaikan jumlah dari kolom-kolom tertentu dengan jumlah kolom-kolom dari *subquery*.**
- **Tentukan kolom-kolom dengan nama-nama kolom dan nilai-nilai *default*.**

ORACLE

Membuat Suatu Tabel dari Baris-Baris dalam Tabel Lain

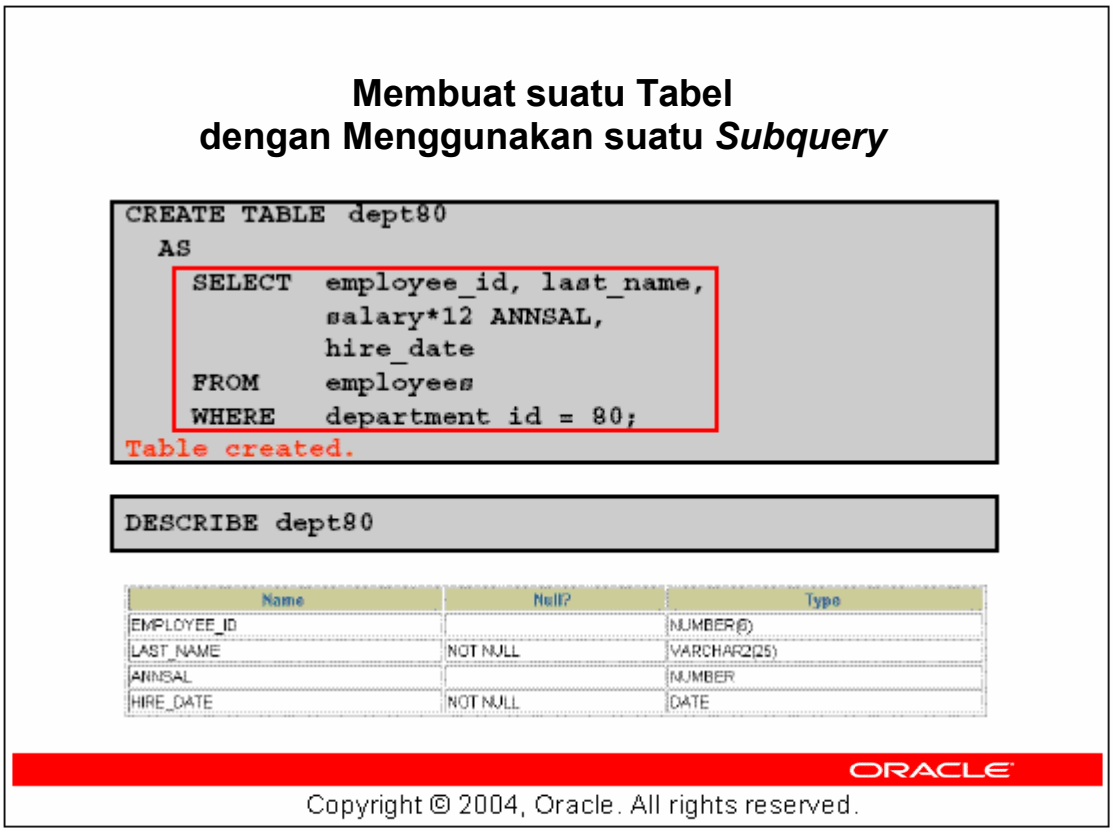
Suatu metode kedua untuk membuat suatu tabel adalah dengan menerapkan klausa `AS subquery`, dimana kedua-duanya membuat tabel dan menyisipkan baris-baris yang dikembalikan dari *subquery*.

Dalam sintak :

`table`	adalah nama dari tabel
`column`	adalah nama kolom, nilai *default* dan *integrity constraint*
`subquery`	adalah pernyataan `SELECT` yang menentukan sekelompok baris-baris untuk disisipkan ke dalam tabel baru

Pedoman-Pedoman

- Tabel dibuat dengan menentukan nama-nama kolom dan mengambil baris-baris dengan pernyataan `SELECT` yang disisipkan ke dalam tabel.
- Pendefinisian kolom boleh berisi hanya nama kolom dan nilai *default*.
- Jika kolom-kolom tertentu tidak diberikan, nama-nama kolom dari tabel adalah sama seperti nama-nama kolom dalam *subquery*.
- Pendefinisian tipe data kolom dan *constraint* `NOT NULL` adalah dilalui pada tabel baru. Aturan-aturan *constraint* lain adalah tidak dilalui pada tabel baru. Bagaimanapun, Anda boleh menambahkan *constraint-constraint* pada pendefinisian kolom.



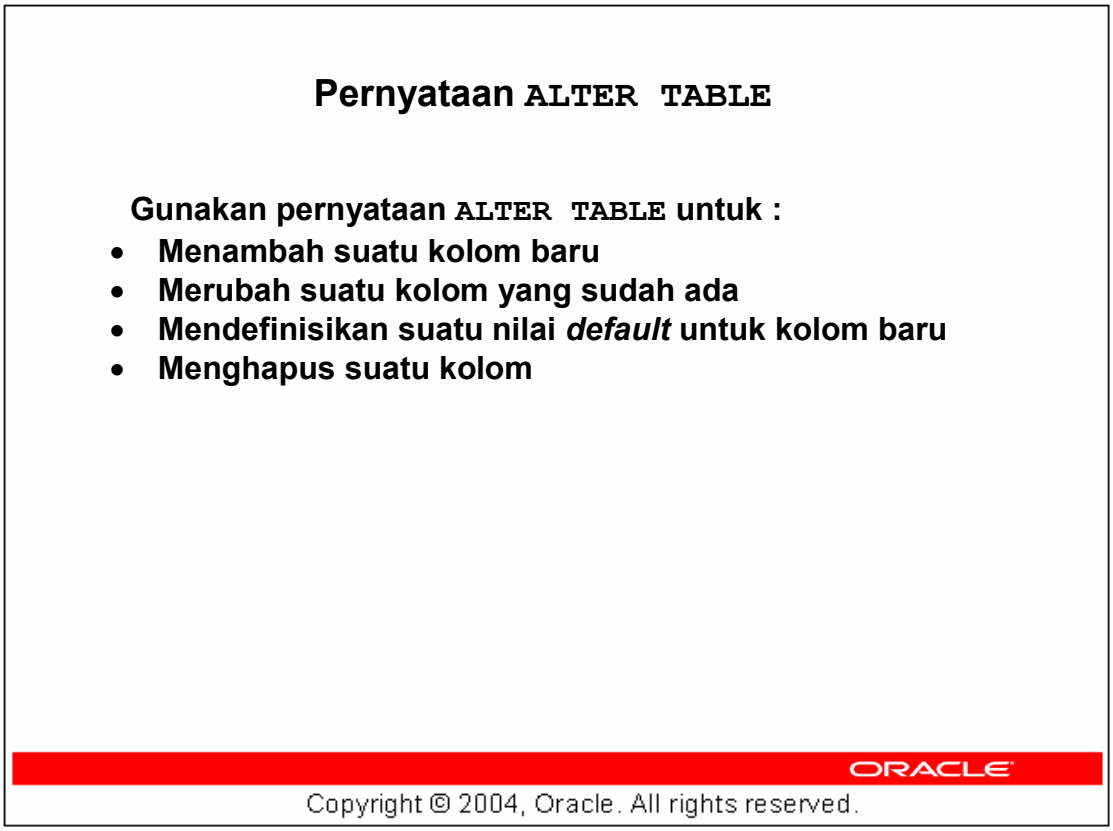
Membuat Suatu Tabel dari Dari Baris-baris Tabel Lain (lanjutan)

Contoh pada slide membuat suatu tabel bernama `DEPT80`, yang berisi rincian dari seluruh pegawai yang bekerja di departemen 80. Perhatikan bahwa data untuk tabel `DEPT80` berasal dari tabel `EMPLOYEES`.

Anda bisa menguji keberadaan dari suatu tabel database dan memeriksa definisi-definisi kolom dengan menggunakan perintah *i*SQL*Plus `DESCRIBE`.

Pastikan untuk menyediakan suatu alias kolom saat memilih suatu ekspresi. Ekspresi `SALARY*12` adalah memberikan alias `ANNSAL`. Tanpa alias, kesalahan berikut ditampilkan :

```
ERROR at line 3:
ORA-00998: must name this expression with a column alias
```

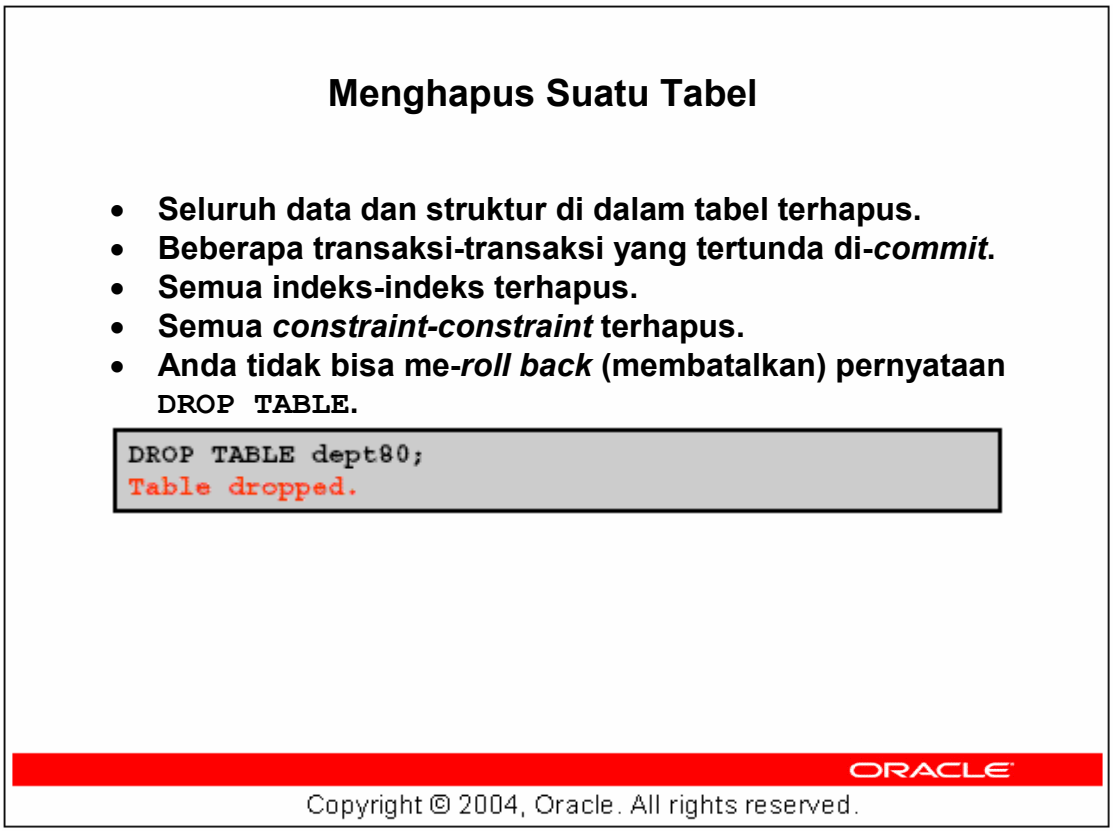


Pernyataan ALTER TABLE

Setelah Anda membuat suatu tabel, mungkin Anda ingin untuk merubah struktur tabel untuk beberapa alasan berikut :

- Anda menghilangkan suatu kolom.
- Definisi kolom Anda perlu untuk dirubah.
- Anda ingin untuk menghapus kolom-kolom.

Anda dapat melakukan hal tersebut dengan menggunakan pernyataan `ALTER TABLE`. Untuk informasi tentang pernyataan `ALTER TABLE`, lihat pelajaran *Oracle Database 10g SQL Fundamentals II* .



Menhapus Suatu Tabel

Pernyataan `DROP TABLE` menghapus pendefinisan suatu tabel Oracle. Saat Anda menghapus (*drop*) suatu tabel, database kehilangan semua data di dalam tabel dan semua indeks-indeks yang terhubung dengannya.

Sintak

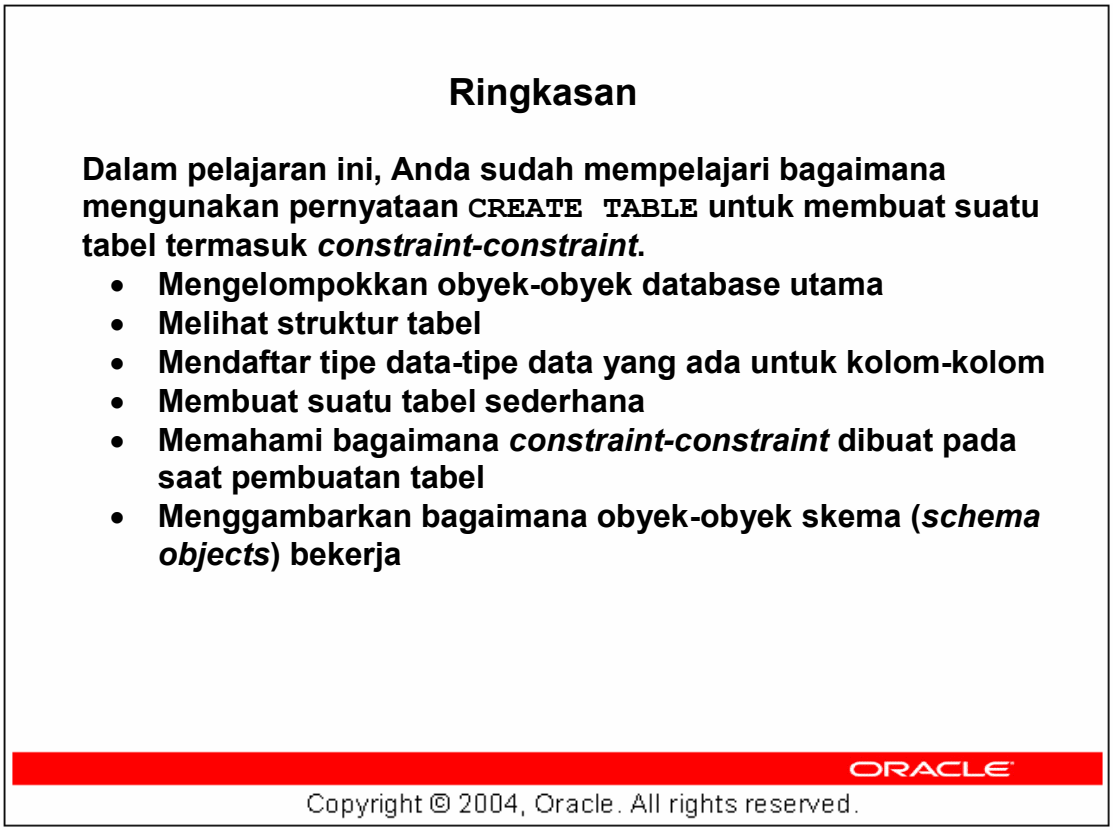
```
DROP TABLE table
```

Dalam sintak, *table* adalah nama suatu tabel.

Pedoman-pedoman

- Semua data terhapus dari tabel.
- Beberapa *view-view* dan *synonym* tetap ada tapi tidak valid.
- Beberapa transaksi-transaksi yang tertunda di-*commit.*
- Hanya pembuat tabel atau seorang user dengan hak `DROP ANY TABLE` bisa menghapus suatu tabel.

Catatan : Pernyataan `DROP TABLE`, sekali dieksekusi, tidak dapat diubah. Server Oracle tidak bertanya tindakan saat Anda mengeluarkan pernyataan `DROP TABLE`. Jika Anda memiliki tabel tersebut atau memiliki tingkat hak tertinggi, maka tabel dengan segera terhapus. Seperti semua pernyataan-pernyataan DDL, `DROP TABLE` secara otomatis ter-commit.



Ringkasan

Dalam pelajaran ini, Anda sudah mempelajari bagaimana melakukan hal-hal berikut :

CREATE TABLE

- Gunakan pernyataan `CREATE TABLE` untuk membuat suatu tabel dan menyertakan *constraint-constraint*.
- Membuat suatu tabel berdasarkan tabel lain dengan menggunakan suatu *subquery*.

DROP TABLE

- Menghapus baris-baris dan struktur suatu tabel.
- Sekali dieksekusi, pernyataan ini tidak bisa di *roll back*.