8 Mengubah Data

Tujuan

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

- **Menjelaskan setiap pernyataan *data manipulation language* (DML)**
- **Menyisipkan baris-baris ke dalam suatu tabel**
- **Menghapus baris-baris dari suatu tabel**
- **Transaksi-transaksi kontrol**

ORACLE

Tujuan

Pada pelajaran ini, Anda belajar bagaimana menggunakan pernyataan DML untuk menyisipkan baris-baris ke dalam suatu tabel, *update* baris-baris yang ada dalam suatu tabel, dan menghapus baris-baris yang ada dari suatu tabel. Anda juga belajar bagaimana mengontrol transaksi-transaksi dengan pernyataan-pernyataan `COMMIT`, `SAVEPOINT`, dan `ROLLBACK`.

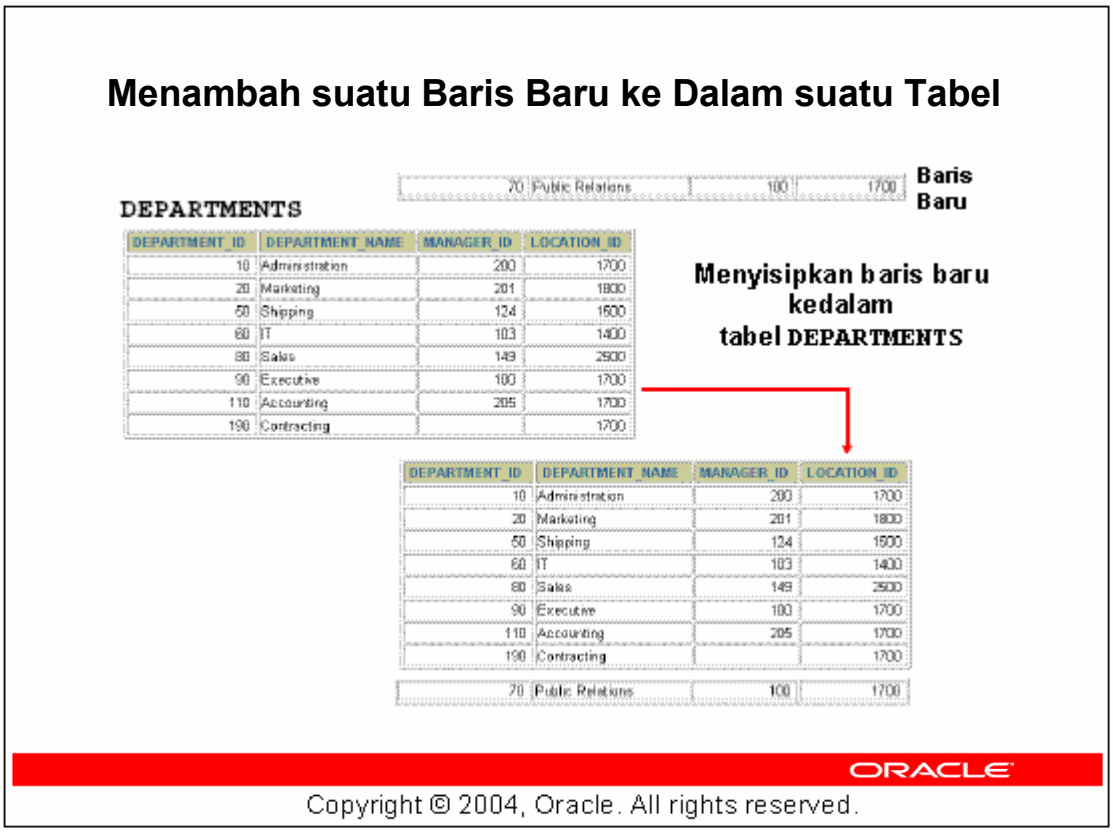
Data Manipulation Langguage

- **Suatu pernyataan DML dieksekusi saat Anda :**
  - **Menambah baris-baris baru ke suatu tabel**
  - **Memodifikasi baris-baris yang ada dalam suatu tabel**
  - **Menghapus baris-baris yang ada dari suatu tabel**
- **Suatu *transaksi* terdiri dari sekumpulan pernyataan-pernyataan DML yang membentuk suatu unit logika dari pekerjaan.**

Data Manipulating Language

Data Manipulating Language (DML) adalah suatu bagian inti dari SQL. Ketika Anda ingin menambah, *update*, atau menghapus data dalam suatu database, Anda mengeksekusi suatu pernyataan DML. Suatu kumpulan pernyataan-pernyataan DML yang membentuk suatu unit logika dari pekerjaan dinamakan *transaction* (transaksi).

Bayangkan suatu database perbankan. Ketika seorang nasabah bank mentransfer uang dari tabungannya ke rekening tujuan, transaksi mungkin terdiri dari tiga operasi terpisah : mengurangi saldo rekening, menambah rekening tujuan, dan mencatat transaksi di jurnal transaksi. Server Oracle harus menjamin bahwa ketiga pernyataan SQL dilakukan untuk menjaga rekening pada saldo yang sesuai. Ketika ada sesuatu yang menghalangi salah satu pernyataan di transaksi sejak dieksekusi, pernyataan yang lain dari transaksi harus dibatalkan.



Menambah suatu Baris Baru ke Dalam Tabel

Garis pada slide menggambarkan penambahan sebuah departemen baru ke tabel `DEPARTEMENTS`.

Sintak Pernyataan INSERT

- **Menambah baris-baris baru ke suatu tabel dengan menggunakan pernyataan INSERT :**

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- **Dengan sintak ini, hanya satu baris ynag disisipkan pada suatu waktu**

Menambah suatu Baris Baru ke suatu Tabel (lanjutan)

Anda dapat menambah baris baru ke suatu tabel dengan menggeluarkan pernyataan `INSERT`.

Dalam sintak:
table adalah nama dari suatu tabel
column adalah nama dari kolom dalam tabel yang ditambah
value adalah nilai yang sesuai untuk kolom

Note: Pernyataan ini dengan klausa `VALUE`-nya menambah hanya satu baris pada suatu waktu pada suatu tabel.

Menyisipkan Baris-Baris Baru

- **Menyisipkan suatu baris baru berisi nilai-nilai untuk setiap kolom.**
- **Daftar nilai-nilai dalam urutan *default* dari kolom-kolom dalam tabel.**
- **Daftar kolom-kolom dalam klausa `INSERT`, adalah optional.**

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 row created.
```

- **Apit nilai-nilai karakter dan tanggal dengan tanda petik tunggal.**

ORACLE

Menambah suatu Baris Baru ke suatu Tabel (lanjutan)

Karena Anda dapat menyisipkan satu baris baru yang berisi nilai-nilai untuk setiap kolomnya, daftar kolom tidak dibutuhkan pada klausa `INSERT`. Bagaimanapun, jika Anda tidak menggunakan daftar kolom, nilai-nilai harus didaftar sesuai dengan urutan *default* kolom dalam tabel, dan sebuah nilai harus disebutkan untuk masing-masing kolom.

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Untuk lebih jelasnya, gunakan daftar kolom pada klausa `INSERT`.

Apit nilai-nilai karakter dan tanggal dengan tanda petik tunggal, Anda tidak disarankan untuk mengapit nilai-nilai numerik dengan tanda petik tunggal.

Nilai-nilai bilangan tidak harus diapit dengan tanda petik tunggal, karena konversi secara implisit akan mengambil alih untuk nilai-nilai numerik yang diberikan ke kolom-kolom dengan tipe data `NUMBER` jika tanda petik tunggal disertakan.

Menyisipkan Baris-Baris dengan Nilai-Nilai *Null*

- **Secara implisit : Menghilangkan nama kolom dari daftar kolom.**

```
INSERT INTO   departments (department_id,
                           department_name )
VALUES        (30, 'Purchasing');
1 row created.
```

- **Secara eksplisit : Meyebutkan kata kunci (*keyword*) `NULL` dalam klausa `VALUES`.**

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 row created.
```

ORACLE

Cara-Cara untuk Menyisipkan Nilai-Nilai *Null*

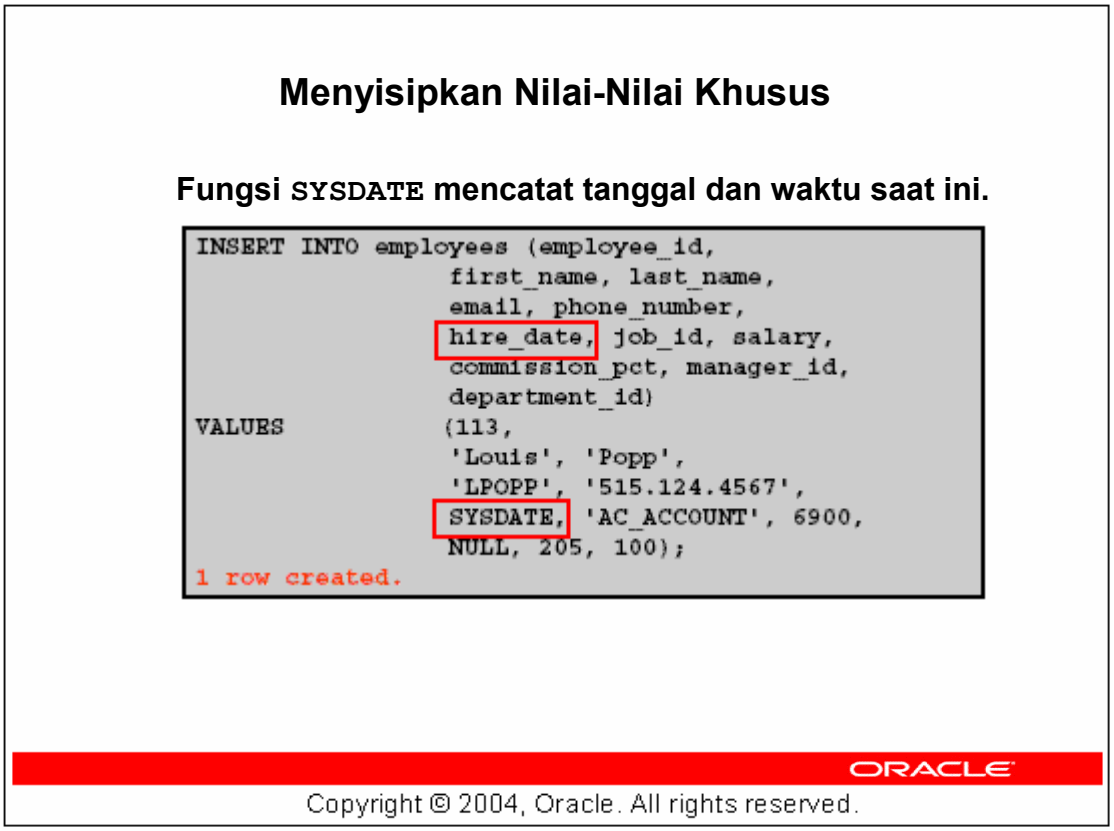
Method	Keterangan
Implisit	Menghilangkan nama kolom dari daftar kolom
Eksplisit	Menyebutkan kata kunci (*keyword*) `NULL` pada daftar `VALUES` Menetapkan *string* kosong (`' '`) pada daftar `VALUES` untuk rangkaian (*string*) karakter dan tanggal

Pastikan bahwa Anda bisa menggunakan nilai-nilai *null* pada kolom target dengan mengecek `NULL`? Dengan perintah *i*SQL*Plus `DESCRIBE`.

Server Oracle secara otomatis menjalankan semua tipe data, panjang data, dan *integrity constraints*. Beberapa kolom yang tidak didaftarkan secara eksplisit menghasilkan suatu nilai *null* dalam baris baru.

Kesalahan-kesalahan yang biasanya dapat terjadi selama input user :

- Nilai *mandatory* tidak ditemukan untuk suatu kolom `NOT NULL`
- Nilai yang sama malanggar *constraint-constraint* yang unik.
- *Constraint foreign key* dilanggar
- `CHECK` *constraint* dilanggar
- Tipe data tidak cocok
- Nilai terlalu lebar untuk diletakkan di dalam kolom



Menyisipkan Nilai-Nilai khusus dengan Menggunakan Fungsi-Fungsi SQL

Anda dapat menggunakan fungsi-fungsi (*functions*) untuk memasukkan nilai-nilai khusus dalam tabel Anda.

Contoh pada slide mencatat informasi untuk pegawai Popp dalam tabel `EMPLOYEES`. Informasi yang dicatat menyediakan tanggal dan waktu saat ini pada kolom `HIRE_DATE`. Informasi dicatat menggunakan fungsi `SYSDATE` untuk tanggal dan waktu saat ini.

Anda juga bisa menggunakan fungsi `USER` ketika menyisipkan baris-baris dalam suatu tabel. Fungsi `USER` mencatat *username* saat ini.

Menegaskan Penambahan ke Tabel

```
SELECT employee_id, last_name, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
113	Popp	AC_ACCOUNT	27-SEP-01	

Menyisipkan Nilai-Nilai Tanggal Tertentu

- **Menambah seorang pegawai baru.**

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

- **Memeriksa penambahan Anda.**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

ORACLE

Menyisipkan Nilai-Nilai Tanggal dan Waktu Tertentu

Format `DD-MON-YY` biasanya digunakan untuk menyisipkan suatu nilai tanggal. Dengan format ini, menyebabkan *default* abad pada abad sekarang. Karena tanggal juga berisi informasi waktu, *default* waktu adalah tengah malam (00:00:00).

Jika suatu tanggal harus dimasukkan ke dalam suatu format lain selain format *default* (sebagai contoh, dengan abad yang lain atau suatu waktu tertentu), Anda harus menggunakan fungsi `TO_DATE`.

Contoh pada slide informasi dicatat untuk pegawai bernama Raphaely dalam tabel `EMPLOYEES`. Informasi dicatat menjadikan kolom `HIRE_DATE` menjadi February 3, 1999. Jika Anda menggunakan pernyataan berikut daripada yang ditampilkan pada slide, tahun pada saat mulai bekerja dianggap sebagai tahun 2099.

```
INSER INTO  employees
VALUES      (1444, 'Den', 'Raphaely', 'DRAPHAEL',
            '515.127.4561','03-FEB-99', 'AC_ACCOUNT', 11000,
            NULL, 100, 30);
```

Jika format `RR` yang digunakan, sistem menyediakan abad yang tepat secara otomatis, jika bukan pada abad sekarang.

Membuat suatu *Script*

- **Gunakan subtitusi `&` dalam suatu pernyataan SQL untuk menunjukkan nilai-nilai.**
- **`&` adalah suatu penampung untuk nilai variabel.**

```
INSERT INTO departments
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department_name',&location);
```


```
1 row created.
```

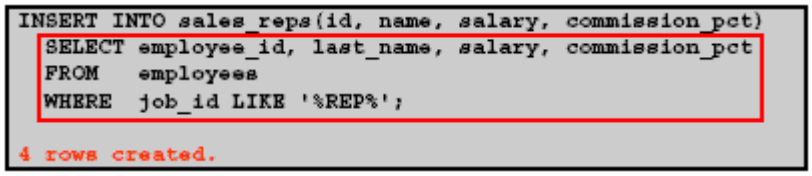

Membuat suatu *Script* untuk Mengubah Data

Anda dapat menyimpan perintah-perintah dengan *substitution variables* dalam suatu file dan mengeksekusi perintah-perintah di dalam file tersebut. Contoh pada slide mencatat informasi untuk suatu departemen dalam tabel `DEPARTMENTS`.

Jalankan file *script* dan Anda diperingatkan untuk memasukkan pada masing-masing *substitution variables* `&`. Setelah memasukkan suatu nilai ke *substitution variable*, klik tombol *Continue*. Nilai-nilai yang anda inputkan kemudian disubstitusikan ke dalam pernyataan. Hal ini memungkinkan Aanda untuk menjalankan file *script* yang sama berulang-ulang akan tetapi dengan menyediakan suatu nilai yang berbeda setiap saat Anda menjalankannya.

Menyalin Baris-Baris dari Tabel Lain

- **Tulis pernyataan `INSERT` Anda dengan suatu *subquery*.**



- **Jangan gunakan klausa `VALUES`.**
- **Sesuaikan jumlah kolom-kolom dalam klausa `INSERT` dengan kolom-kolom dalam *subquery*.**

ORACLE

Menyalin Baris-Baris dari Tabel Lain

Anda dapat menggunakan pernyataan `INSERT` untuk menambah baris-baris ke suatu tabel dimana nilai-nilainya diambil dari tabel-tabel yang ada. Dalam klausa `VALUES`, Anda gunakan suatu *subquery*.

Sintak

```
INSERT INTO table[ column (, column)] subquery;
```

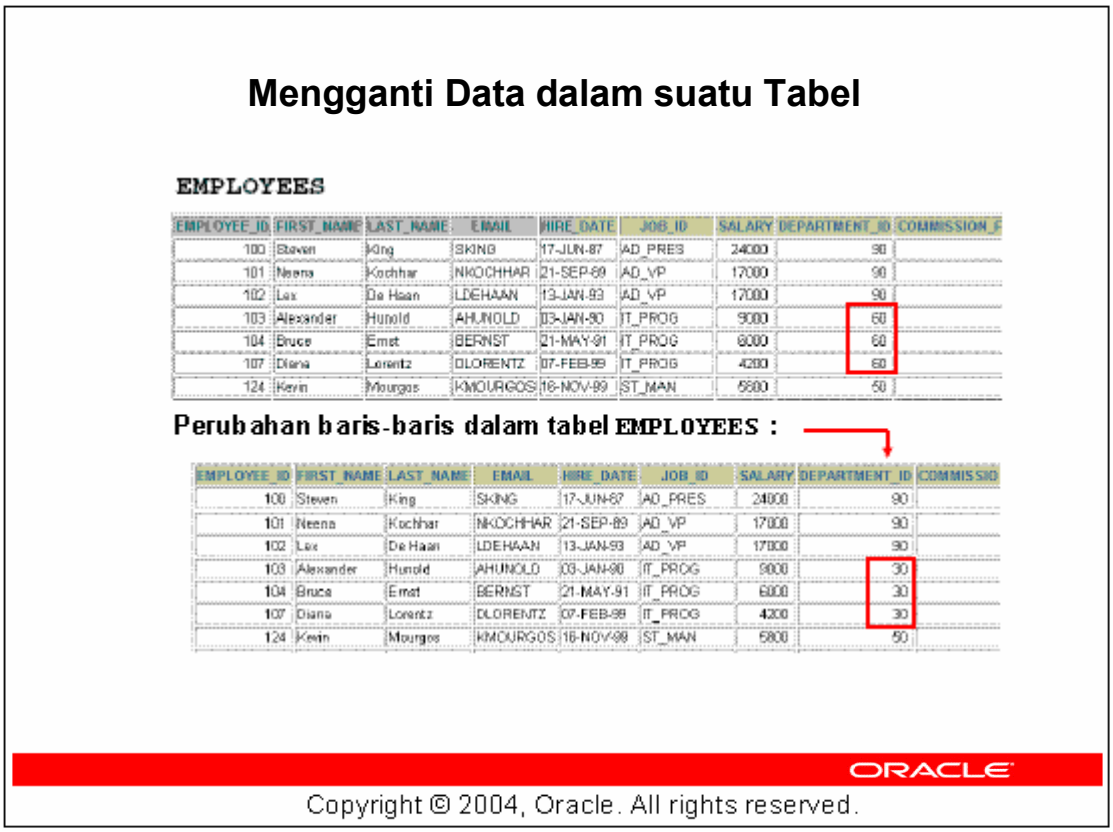
Dalam sintak :

table	adalah nama tabel
column	adalah nama kolom dalam tabel yang ditambah
subquery	adalah *subquery* yang mengembalikan baris-baris ke tabel

Jumlah kolom-kolom dan tipe data-tipe data dalam daftar kolom klausa `INSERT` harus sesuai dengan jumlah nilai-nilai dan tipe data-tipe data dalam *subquery*. Untuk membuat suatu salinan baris-baris dari suatu tabel, gunakan `SELECT *` pada *subquery*:

```
INSERT INTO copy_emp
SELECT *
FROM employees;
```

Untuk informasi lebih lanjut, lihat "`SELECT`" (bagian "*subqueries*") pada *Oracle Database SQL Reference*.



Mengganti Data dalam suatu Tabel

Pada slide digambarkan penggantian nomor departemen untuk pegawai di departemen 60 menjadi departemen 30.

Sintak Pernyataan UPDATE

- **Memodifikasi baris-baris yang ada dengan pernyataan UPDATE :**

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- **Perubahan lebih dari satu baris pada suatu waktu (jika diminta).**

Meng-*update* Baris-Baris

Anda dapat memodifikasi baris-baris yang ada dengan menggunakan pernyataan `UPDATE`.

Pada sintak **:**

`table`	adalah nama tabel
`column`	adalah nama kolom dalam tabel yang diisi
`value`	adalah nilai penghubung atau *subquery* untuk kolom
`condition`	Baris-baris yang diidentifikasi untuk di-*update* dan terdiri dari nama-nama kolom, ekspresi, konstanta, *subqueries*, dan operator-operator pembanding.

Menegaskan operasi *update* dengan meng-*query* tabel untuk menampilkan baris yang telah di-*update*.

Untuk informasi lebih lanjut, lihat ”`UPDATE`” pada *Oracle Database SQL Reference*.

Catatan : Pada umunya, *primary key* digunakan untuk mengidentifikasi suatu baris tunggal (*single row*). Menggunakan kolom-kolom lain dengan tidak benar menyebabkan beberapa haris ter-*update*. Sebagai contoh, mengidentifikasi suatu baris tunggal dalam tabel `EMPLOYEES` dengan menggunakan nama adalah berbahaya., karena lebih dari satu pegawai mungkin memiliki nama yang sama.

Meng-*Update* Baris-Baris dalam suatu Tabel

- **Baris tertentu atau baris-baris dimodifikasi jika Anda menentukan klausa `WHERE` :**

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- **Seluruh baris dalam tabel dimodifikasi jika Anda menghilangkan klausa `WHERE` :**

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated.
```

ORACLE

Meng-*Update*Baris-Baris (lanjutan)

Pernyataan `UPDATE` memodifikasi baris-baris tertentu jika klausa `WHERE` ditentukan. Contoh pada slide memindahkan pegawai bernomor 113 (Popp) ke departemen 70.

Jika Anda menghilangkan klausa `WHERE`, semua baris dalam tabel akan dimodifikasi.

```
SELECT last_name , department_id
FROM copy_emp;
```

LAST_NAME	DEPARTMENT_ID
King	110
Kochar	110
De Haan	110
Hunold	110
Ernst	110
Lorentz	110

. . .

22 rows selected

Catatan: tabel `COPY_EMP` memiliki data yang sama dengan tabel `EMPLOYEES`.

Meng-*Update* Dua Kolom dengan suatu *Subquery*

***Update* pegawai yang memiliki job ID 114 dan penghasilannya agar sesuai dengan pegawai 205.**

```
UPDATE   employees
SET      job_id   = (SELECT  job_id
                     FROM    employees
                     WHERE   employee_id = 205),
         salary   = (SELECT  salary
                     FROM    employees
                     WHERE   employee_id = 205)
WHERE    employee_id     =  114;
1 row updated.
```

ORACLE

Meng-*Update* Dua Kolom dengan suatu *Subquery*

Anda dapat meng-*update* lebih dari satu kolom pada klausa SET dari suatu pernyataan `UPDATE` dengan menulis *multiple subqueries*.

Sintak

```
UPDATE table
SET column =
             (SELECT column
              FROM table
              WHERE condition)
    [ ,
    column =
             (SELECT column
              FROM table
              WHERE condition)]
[WHERE condition];
```

Catatan: jika tidak ada baris yang di-*update*, akan dikembalikan pesan "`0 rows updated`".

Meng-*Update* Baris-Baris Berdasarkan pada Tabel Lain

Gunakan *subquery-subquery* dalam pernyataan `UPDATE` untuk merubah baris-baris dalam suatu tabel berdasarkan nilai-nilai dari tabel lain :

```
UPDATE  copy_emp
SET     department_id  =  (SELECT department_id
                           FROM employees
                           WHERE employee_id = 100)
WHERE   job_id         =  (SELECT job_id
                           FROM employees
                           WHERE employee_id = 200);
1 row updated.
```

ORACLE

Meng-*Update* Baris-Baris Berdasarkan pada Tabel Lain

Anda dapat menggunakan *subquery-subquery* dalam pernyataan `UPDATE` untuk meng-*update* baris-baris dalam suatu tabel. Contoh pada slide meng-*update* tabel `COPY_EMP` berdasarkan pada nilai-nilai dari tabel `EMPLOYEES`. Pernyataan `UPDATE` mengganti nomor departemen dari semua pegawai yang memiliki job ID 200 menjadi 100 nomor departemen saat ini.

Menghapus suatu Baris dari suatu Tabel

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

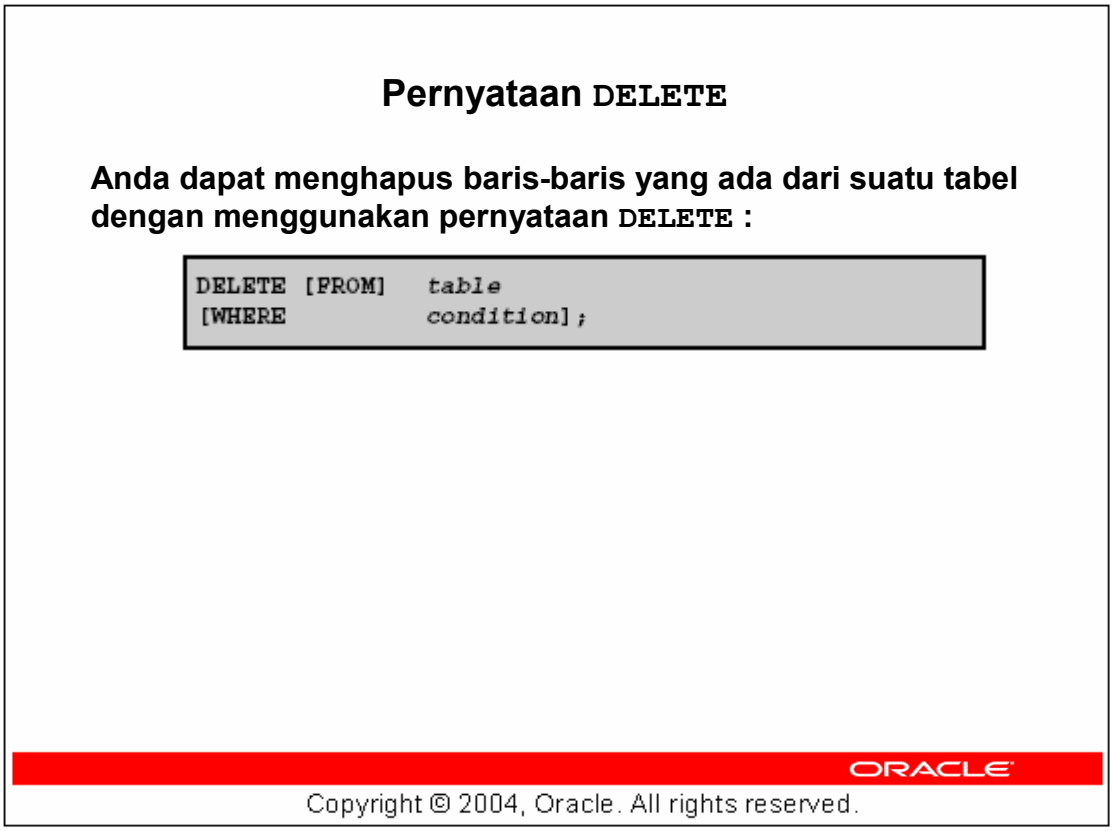
Hapus suatu baris dari tabel DEPARTMENTS :

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

ORACLE

Menghapus suatu Baris dari suatu Tabel

Graphic pada slide menghapus departemen Finance dari tabel `DEPARTMENTS` (asumsikan bahwa tidak ada *constraints* yang didefinisikan pada tabel `DEPARTMENTS`).



Menghapus Baris-Baris

Anda dapat menghapus baris-baris yang ada dengan menggunakan pernyataan `DELETE`.

Dalam sintak :

table	adalah nama tabel
condition	mengidentifikasi baris-baris untuk dihapus dan terdiri dari nama-nama kolom, ekspresi-ekspresi, konstanta-konstanta, *subquery-subquery*, dan operator-operator pembanding.

Catatan : Jika tidak ada baris terhapus, akan dekembalikan pesan "`0 rows are deleted`".

Untuk informasi lebih lanjut, lihat "`DELETE`" dalam *Oracle Database SQL Reference*.

Menghapus Baris-Baris dari suatu Tabel

- **Baris-baris tertentu terhapus jika Anda menentukan klausa `WHERE` :**

```
DELETE FROM departments
WHERE  department_name = 'Finance';
1 row deleted.
```

- **Semua baris dalam tabel terhapus jika Anda menghilangkan klausa `WHERE` :**

```
DELETE FROM  copy_emp;
22 rows deleted.
```


Menghapus Baris-Baris (lanjutan)

Anda dapat menghapus baris-baris tertentu dengan menentukan klausa `WHERE` pada pernyataan `DELETE`. Contoh pada slide menghapus departemen Finance dari tabel `DEPARTMENTS`. Anda dapat menegaskan operasi penghapusan dengan menampilkan baris-baris yang telah dihapus menggunakan pernyataan `SELECT`.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
No row selected.
```

Jika Anda menghilangkan klausa `WHERE`, semua baris dalam tabel dihapus. Contoh kedua pada slide menghapus semua baris dari tabel `COPY_EMP`, karena tidak ada klausa `WHERE` yang ditulis.

Contoh

Menghapus baris-baris yang diidentifikasikan di klausa `WHERE`.

```
DELETE FROM employees WHERE employee_id = 114;
1 row deleted

DELETE FROM departments WHERE department_id IN(30, 40 );
2 rows deleted.
```

Menghapus Baris-Baris Berdasarkan Tabel Lain

Gunakan *subquery-subquery* dalam pernyataan `DELETE` untuk menghapus baris-baris dari suatu tabel berdasarkan nilai-nilai dari tabel lain :

```
DELETE FROM employees
WHERE  department_id =
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
1 row deleted.
```

ORACLE

Menghapus Baris-Baris Berdasarkan pada Tabel Lain

Anda dapat menggunakan *subqueries* untuk menghapus baris-baris dari suatu tabel berdasarkan nilai-nilai dari tabel lain. Contoh pada slide menghapus semua pegawai yang berada di suatu departemen dimana nama departemen berisi rangkaian (*string*) `Public`. *Subquery* mencari tabel `DEPARTMENTS` untuk menemukan nomor departemen berdasarkan pada nama departemen yang berisi *string* `Public`. *Subquery* kemudian menjadikan nomor departemen untuk query utama, yang menghapus baris-baris data dari tabel `EMPLOYEES` berdasarkan nomor departement tersebut.

Pernyataan TRUNCATE

- **Menghilangkan semua baris dari suatu tabel, tabel kosong dan struktur tabel dibiarkan tetap utuh**
- **Adalah suatu pernyataan *data definition langguage* (DDL) dari pada suatu pernyataan DML; tidak bisa dibatalkan.**
- **Sintak :**

```
TRUNCATE TABLE table_name;
```

- **Contoh :**

```
TRUNCATE TABLE copy_emp;
```

ORACLE

Pernyataan TRUNCATE

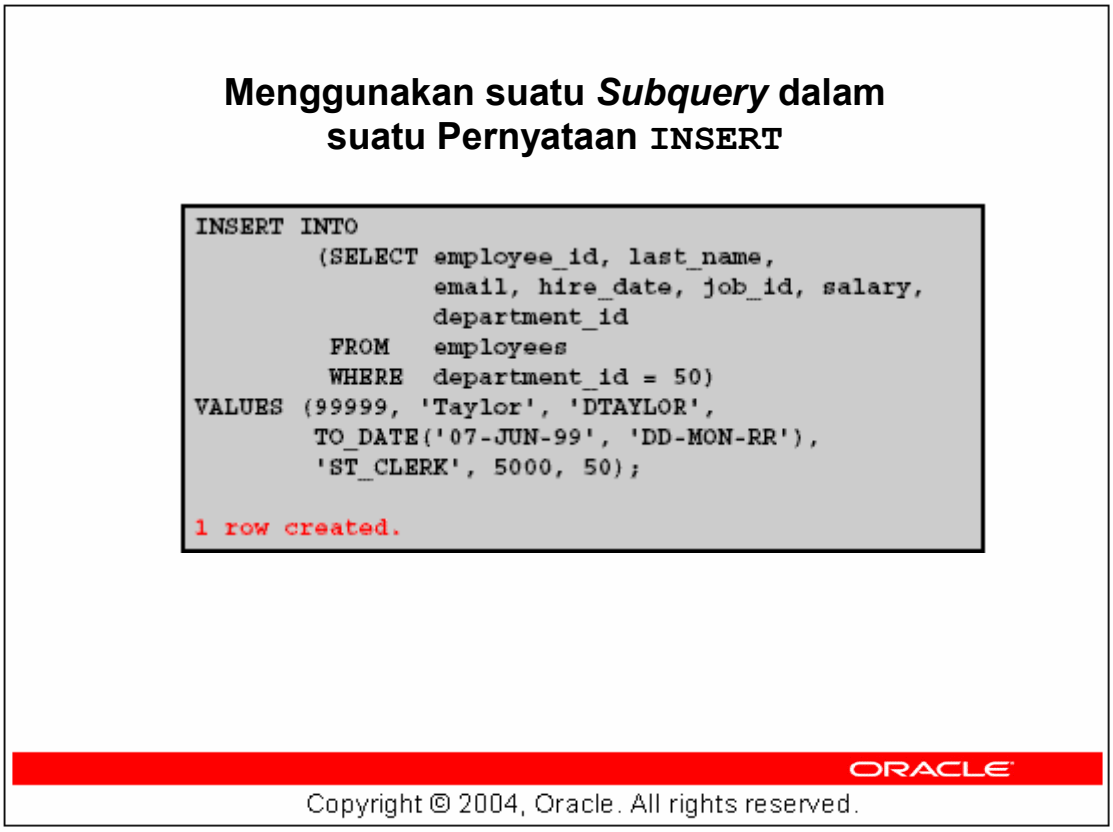
Sebuah method yang lebih efesien dalam mengosongkan sebuah table adalah dengan pernyataan TRUNCATE.

Anda dapat menggunakan pernyataan TRUNCATE untuk menghapus secara cepat semua baris dari suatu tabel atau *cluster* (kelompok).

Menghapus baris-baris dengan pernyataan TRUNCATE adalah lebih cepat dari pada menghapusnya dengan pernyataan DELETE dengan alasan sebagai berikut :

- Pernyataan TRUNCATE adalah suatu pernyataan *data definition language* (DDL) dan tidak membangkitkan informasi *rollback*. Informasi *rollback* dibahas pada pelajaran selanjutnya.
- Men-*truncate* suatu table tidak mengaktifkan picu-picu (*triggers*) penghapusan dalam table.
- Jika table adalah induk dari suatu *integrity constraint* referensial, Anda tidak dapat men-*truncate* tabel. Anda perlu untuk men-*disable constraint* sebelum mengeluarkan pernyataan TRUNCATE.

  Men-*disable constraint* dibahas dalam pelajaran berikutnya.



Menggunakan suatu *Subquery* dalam suatu Pernyataan `INSERT`

Anda dapat menggunakan suatu *subquery* dalam penempatan nama tabel pada klausa `INTO` dari pernyataan `INSERT`.

Daftar *select* dari *subquery* ini harus memiliki jumlah kolom-kolom yang sama dengan daftar kolom dari klausa `VALUES`. Beberapa aturan pada kolom-kolom dari tabel utama (*base table*) harus diikuti jika pernyataan `INSERT` dapat dikerjakan dengan sukses. Sebagai contoh, Anda tidak dapat menaruh suatu ID pegawai yang sama atau menghilangkan suatu nilai untuk suatu kolom dengan *mandatory not-null*.

Menggunakan suatu *Subquery* dalam suatu Pernyataan `INSERT`

Memeriksa hasil-hasil :

```
SELECT employee_id, last_name, email, hire_date,
       job_id, salary, department_id
FROM   employees
WHERE  department_id = 50;
```

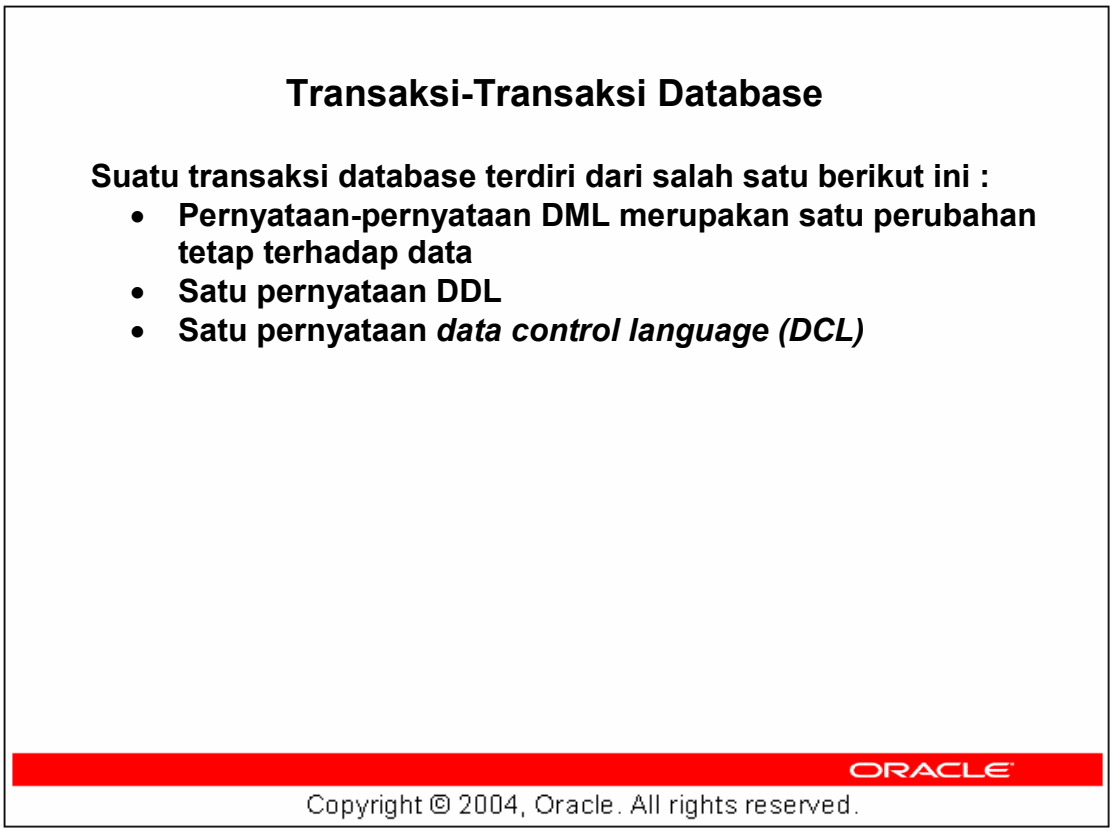
EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50
141	Rajs	TRAJS	17-OCT-95	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	9000	50

6 rows selected.

ORACLE

Menggunakan suatu *Subquery* dalam suatu Pernyataan `INSERT` (lanjutan)

Contoh menampilkan hasil dari *subquery* yang pernah digunakan untuk mengidentifikasi tabel pada pernyataan `INSERT`.



Transaksi-Transaksi Database

Server Oracle memastikan konsistensi data berdasarkan pada transaksi-transaksi. Transaksi-transaksi memberi Anda lebih fleksibel dan kontrol ketika mengubah data, dan server Oracle memastikan konsistensi data dalam even gagalnya proses user atau gagalnya sistem.

Transaksi-transaksi terdiri dari pernyataan-pernyataan DML yang membuat satu perubahan tetap terhadap data. Sebagai contoh, suatu transfer dana antara dua rekening akan dimasukkan ke debet pada satu rekening dan kredit pada rekening yang lain dalam jumlah yang sama. Kedua aksi akan gagal kedua-duanya atau sukses bersama-sama; kredit tidak akan di-*commit* tanpa debet.

Tipe-Tipe Transaksi

Tipe	Keterangan
Data manipulation language (DML)	Terdiri dari beberapa jumlah pernyataan DML yang diperlakukan server Oracle sebagai suatu entitas tunggal atau suatu unit logika dari kerja
Data definition language (DDL)	Terdiri dari hanya satu pernyataan DDL
Data control language (DCL)	Terdiri dari hanya satu pernyataan DCL

Transaksi-Transaksi Database

- **Dimulai saat pernyataan SQL DML pertama dieksekusi**
- **Diakhiri dengan salah satu dari even-even berikut ini :**
  - **Dikeluarkannya suatu pernyataan `COMMIT` atau `ROLLBACK`.**
  - **Eksekusi pernyataan-pernyataan DDL atau DCL (*commit* otomatis).**
  - **User keluar dari *i*SQL*Plus.**
  - **System terganggu (*crashes*).**

ORACLE

Kapan suatu Transaksi Dimulai dan Diakhiri ?

Suatu traksaksi dimulai ketika pernyataan DML pertama ditemukan dan diakhiri ketika muncul salah satu berikut ini :

- Dikeluarkannya pernyataan `COMMIT` atau `ROLLBACK`.
- Dikeluarkannya suatu pernyataan DDL, seperti `CREATE`.
- Dikeluarkannya suatu pernyataan DCL.
- User keluar dari *i*SQL*Plus.
- Mesin gagal atau sistem terganggu.

Setelah satu transaksi berakhir, *executable* pernyataan SQL berikutnya secara otomatis mengawali transaksi selanjutnya.

Keuntungan-Keuntungan dari Pernyataan COMMIT dan ROLLBACK

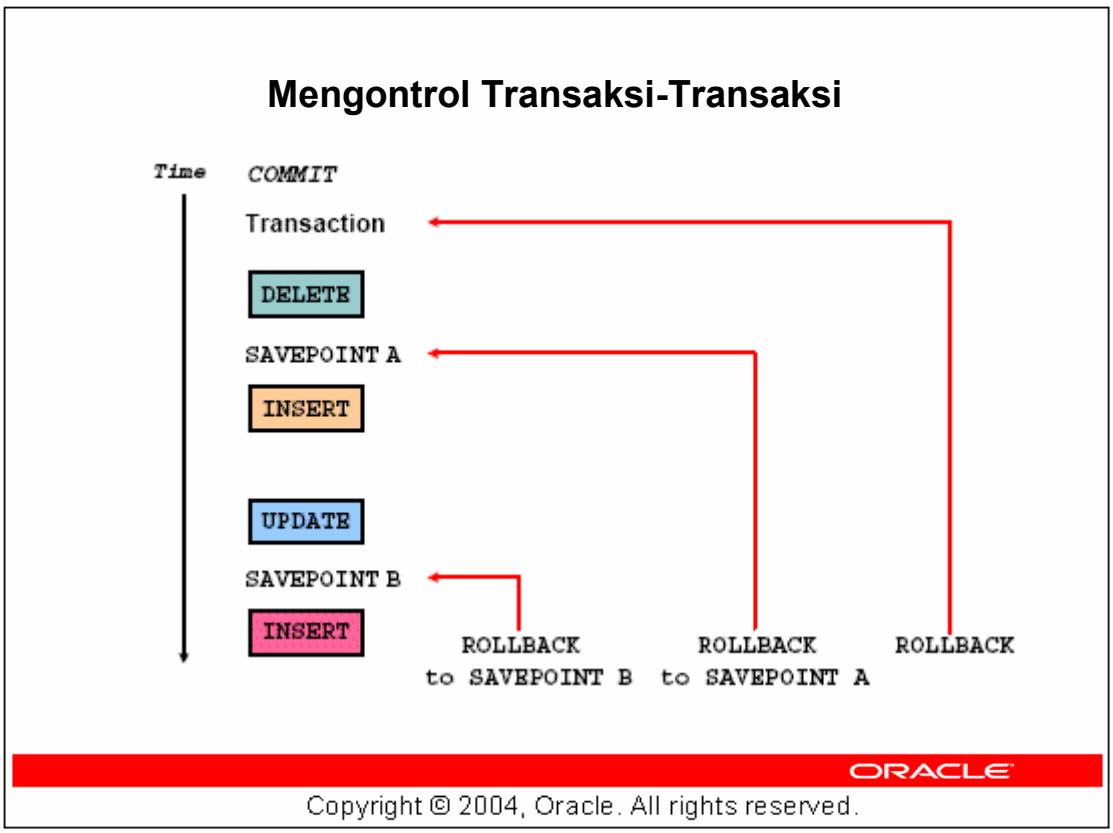
Dengan pernyataan COMMIT dan ROLLBACK, Anda dapat :

- **Memastikan konsistensi data**
- **Melihat perubahan-perubahan data sebelum membuat permanen perubahan**
- **Secara logika mengelompokkan operasi-operasi yang berkaitan**

ORACLE

Keuntungan-Keuntungan dari Pernyataan COMMIT dan ROLLBACK

Dengan pernyataan COMMIT dan ROLLBACK, Anda memiliki kontrol perubahan-perubahan terhadap data permanen.



Pernyataan-Pernyataan Kontrol Transaksi Eksplisit

Anda dapat mengontrol transaksi-transaksi logika dengan menggunakan pernyataan-pernyataan `COMMIT`, `SAVEPOINT`, dan `ROLLBACK`.

Pernyataan	Keterangan
`COMMIT`	Mengakhiri transaksi saat ini dengan membuat semua data sementara dirubah manjadi permanen
`SAVEPOINT` *name*	Menandai suatu *savepoint* sampai transaksi saat ini.
`ROOLBACK`	`ROLLBACK` mengakhiri transaksi saat ini dengan membuang semua perubahan data sementara
`ROLLBACK TO SAVEPOINT` *name*	`ROLLBACK TO SAVEPOINT` kembali (*roolls back*) dari transaksi saat ini sampai *savepoint* tertentu, dengan cara demikian membuang beberapa perubahan dan atau *savepoint* yang pernah dibuat sesudah *savepoint* sampai dimana Anda me-*roll back*. Jika Anda menhilangkan klausa `TO SAVEPOINT`, pernyataan `ROLLBACK` me-*roll back* didalam transaksi. Karena *savepoint* adalah logika, maka tidak ada cara untuk mendaftar *savepoint* yang Anda sudah buat.

Catatan : `SAVEPOINT` bukan standar SQL ANSI.

Me-Roll Back Perubahan-Perubahan ke suatu Tanda

- **Buat suatu tanda dalam suatu transaksi saat ini dengan menggunakan pernyataan `SAVEPOINT`.**
- ***Roll back* ke tanda tersebut dengan menggunakan pernyataan `ROLLBACK TO SAVEPOINT`.**

```
UPDATE...
SAVEPOINT update_done
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```


Me-*Roll Back* Perubahan-Perubahan ke suatu Tanda

Anda dapat membuat suatu tanda dalam transaksi saat ini dengan menggunakan pernyataan `SAVEPOINT`, yang membagi transaksi ke dalam bagian-bagian kecil. Anda kemudian dapat membuang perubahan-perubahan sementara sampai tanda tersebut dengan menggunakan pernyataan `ROLLBACK TO SAVEPOINT`.

Jika Anda membuat suatu *savepoint* kedua dengan nama yang sama dengan savepoint sebelumnya, savepoint sebelumnya dihapus.

Memproses Transaksi Implisit

- **Suatu *commit* otomatis terjadi di bawah keadaan-keadaan berikut :**
  - **Dikeluarkannya pernyataan DDL**
  - **Dikeluarkannya pernyataan DCL**
  - **Keluar secara normal dari *i*SQL*PLUS, tanpa secara ekspilist mengeluarkan pernyataan-pernyataan `COMMIT` atau `ROLLBACK`**
- **Suatu *rollback* otomatis terjadi dibawah suatu penghentian abnormal dari *i*SQL*PLUS atau suatu kegagalan sistem.**

Memproses Transaksi Implisit

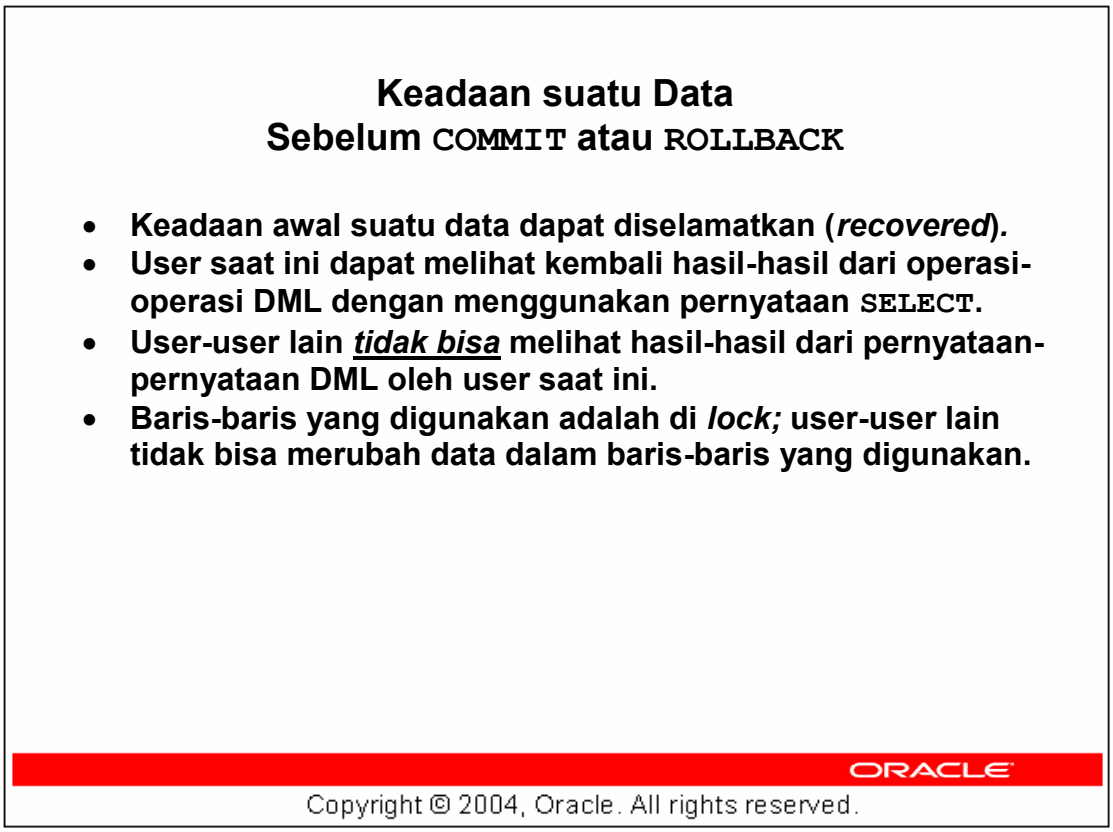
Status	Kejadian
Automatic commit	Pernyataan DDL atau pernyataan DCL dikeluarkan. *i*SQL*Plus keluar secara normal, tanpa secara eksplisit mengeluarkan perintah-perintah `COMMIT` atau `ROLLBACK`.
Automatic rollback	Penghentian abnormal dari *i*SQL*Plus atau sistem gagal.

Catatan : Suatu perintah ketiga ada dalam *i*SQL*PLUS. Perintah `AUTOCOMMIT` dapat dibuat on atau off. Jika diset *on*, setiap pernyataan DML individual di-*commit* segera setelah dieksekusi. Anda tidak dapat me- *roll back* perubahan-perubahan. Jika diset *off,* pernyataan `COMMIT` masih bisa dikeluarkan secara eksplisit. Juga, pernyataan `COMMIT` dikeluarkan saat suatu pernyataan DML dikeluarkan atau saat Anda keluar dari *i*SQL*PLUS.

Kegagalan Sistem (*System Failures*)

Saat suatu transaksi disela oleh gagalnya suatu sistem, semua transaksi secara otomatis di-*roll back*. Ini mencegah kesalahan dari akibat perubahan-perubahan yang tidak diinginkan terhadap data dan mengembalikan tabel-tabel ke keadaan saat terakhir kali *commit*. Dalam cara ini, server Oracle melindungi integritas tabel-tabel.

Dari *i*SQL*Plus, keluar normal dari sesi diselesaikan dengan meng-klik tombol keluar. Dengan SQL*Plus**,** keluar secara normal dilakukan dengan mengetik perintah `EXIT` di *prompt*. Menutup jendela (*window*) ditafsirkan sebagai suatu keluar yang abnormal.

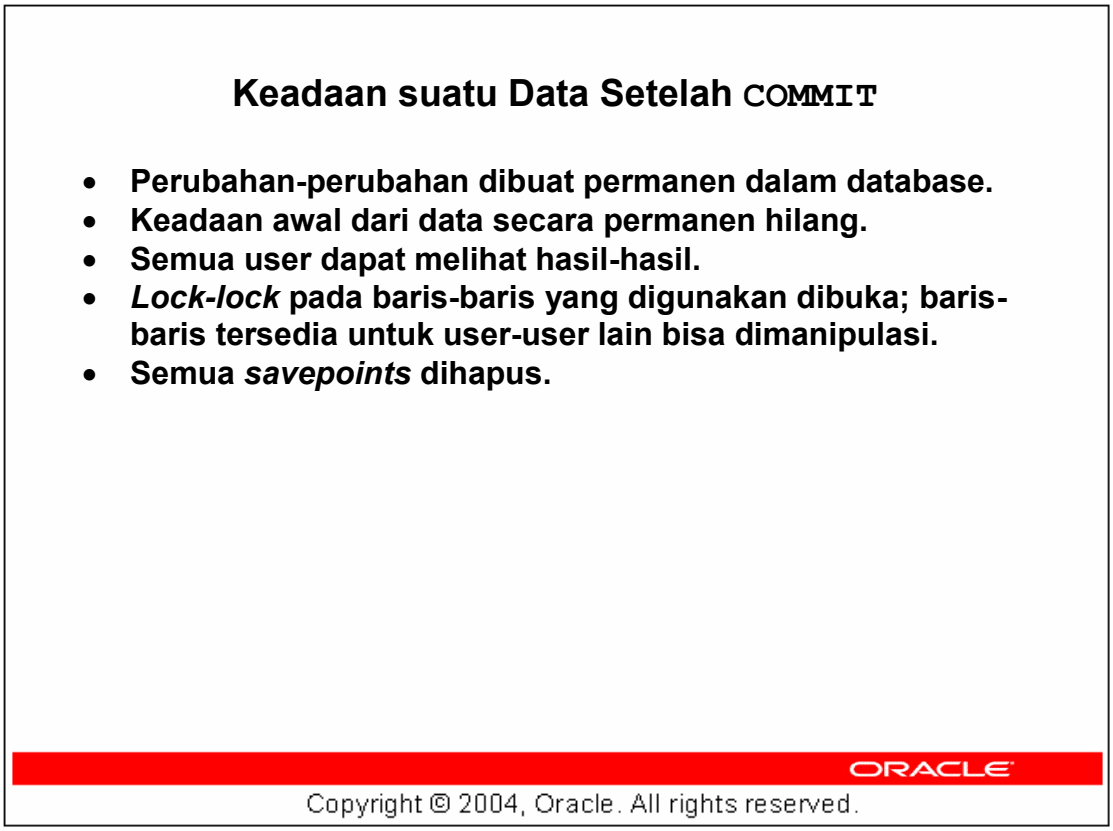


Meng-Commit Perubahan-Perubahan

Setiap perubahan data yang dilakukan selama transaksi adalah sementara sampai dilakukan *commit*.

Keadaan dari data sebelum pernyataan `COMMIT` atau `ROLLBACK` dijalankan dapat dijelaskan sebagai berikut :

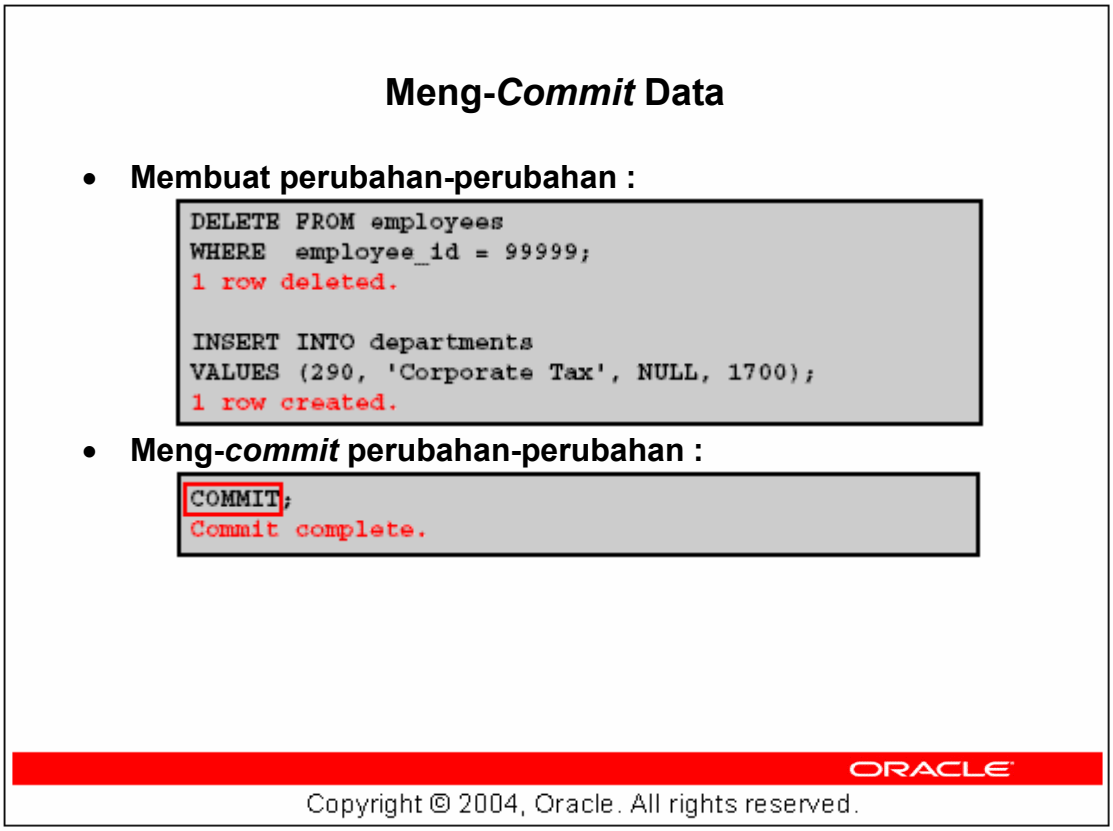
- Operasi-operasi manipulasi data secara primer mempengaruhi *buffer* database ; karena itu keadaan semula dari data dapat dikembalikan.
- User yang sedang bekerja dapat melihat hasil-hasil dari operasi manipulasi data dengan meng-query tabel.
- User lain tidak dapat melihat hasil-hasil dari operasi manipulasi data yang dibuat oleh user yang sedang bekerja. Oracle server melakukan pembacaan konsisten untuk memastikan bahwa setiap user melihat data yang terakhir di-commit.
- Baris yang sedang digunakan akan dikunci (*locked*); sehingga user lain tidak dapat merubah data pada baris tersebut.



Meng-Commit Perubahan-Perubahan (lanjutan)

Perintah `COMMIT` membuat semua perubahan sementara menjadi permanen. Hal-hal berikut ini terjadi setelah perintah `COMMIT` dilakukan :

- Perubahan data akan disimpan dalam database.
- Keadaan data semula tidak ada dengan query-query SQL normal.
- Semua user dapat melihat hasil-hasil dari transaksi.
- Baris yang sedang digunakan tidak lagi terkunci; data pada baris-baris tersebut sekarang dapat dirubah oleh user yang lain.
- Semua *savepoints* dihapus.



Meng-*Commit* Perubahan-Perubahan (lanjutan)

Contoh pada slide menampilkan sebuah baris yang dihapus dari tabel `EMPLOYEES` dan menambahkan sebuah baris baru pada tabel `DEPARTMENTS`. Selanjutnya membuat perubahan tersebut menjadi permanen dengan menjalankan pernyataan `COMMIT`.

Contoh

Hapus departemen 290 dan 300 dalam tabel DEPARTMENTS, dan *update* sebuah baris dalam tabel `COPY_EMP`. Buat perubahan data menjadi permanen.

```
DELETE FROM departments
WHERE department_id IN (290, 300);
1 row deleted.

UPDATE     employees
  SET      department_id = 80
  WHERE    employee_id = 206;
1 row updated.

COMMIT;
 Commit Complete.
```

Keadaan Data Setelah ROLLBACK

Membuang semua perubahan-perubahan sementara dengan menggunakan pernyataan ROLLBACK :

- **Perubahan-perubahan data dibatalkan.**
- **Keadaan data awal disimpan kembali.**
- ***Lock-lock* pada baris-baris yang digunakan dilepaskan.**

```
DELETE FROM copy_emp;
22 rows deleted.
ROLLBACK ;
Rollback complete.
```

ORACLE

Me-*Roll Back* Perubahan-Perubahan

Membuang semua perubahan sementara dengan menggunakan pernyataan ROLLBACK, akan menghasilkan sebagai berikut :

- Perubahan-perubahan data dibatalkan.
- Keadaan data awal disimpan kembali.
- Lock-lock pada baris-baris yang digunakan dilepaskan.

Keadaan Data Setelah ROLLBACK

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE  id = 100;
1 row deleted.

SELECT * FROM   test WHERE  id = 100;
No rows selected.

COMMIT;
Commit complete.
```

ORACLE

Contoh

Ketika mencoba untuk menghapus sebuah *record* dari tabel `TEST`, Anda dapat secara tidak sengaja menghapus isi tabel. Anda dapat melakukan koreksi terhadap kesalahan ini, mengeluarkan kembali perintah yang sesuai, dan membuat permanen perubahan data.

Statement-Level Rollback

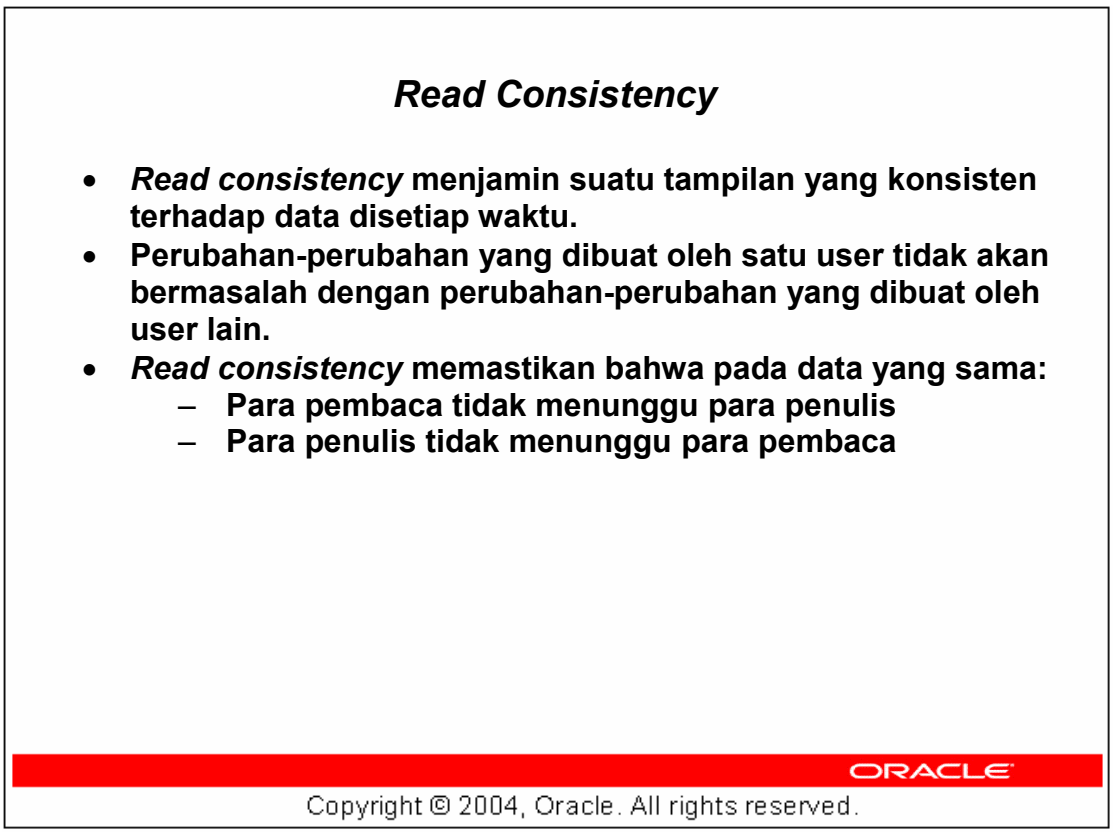
- **Jika suatu pernyataan DML gagal saat dijalankan, hanya pernyataan itu yang di-*roll back*.**
- **Server Oracle menerapkan suatu *savepoint* implisit.**
- **Semua perubahan lain disimpan.**
- **User harus membatalkan transaksi-transaksi secara eksplisit dengan mengeksekusi suatu pernyataan `COMMIT` atau `ROLLBACK`.**

Statement-Level RollBack

Sebagian dari sebuah transaksi dapat diabaikan dengan suatu *rollback* implisit jika eksekusi suatu pernyataan terdeteksi *error*. Jika suatu pernyataan DML tunggal gagal dieksekusi pada waktu transaksi, efeknya adalah akan dilakukan pembatalan oleh *statement-level rollback*, tetapi perubahan yang dibuat oleh pernyataan-pernyataan DML sebelumnya tidak akan ikut diubah. Pernyataan-pernyataan bisa di-*commit* atau di-*roll back* secara eksplisit oleh user.

Server Oracle mengeluarkan suatu *commit* implisit sebelum dan sesudah beberapa pernyataan DML. Jadi, meskipun pernyataan DDL Anda tidak sukses dieksekusi, Anda tidak dapat melakukan me-*roll back* pernyataan sebelumnya karena server mengeluarkan suatu *commit*.

Hentikan transaksi-transaksi Anda secara eksplisit dengan mengeksekusi pernyataan `COMMIT` atau `ROLLBACK`.



Read Consistency (Konsistensi Baca)

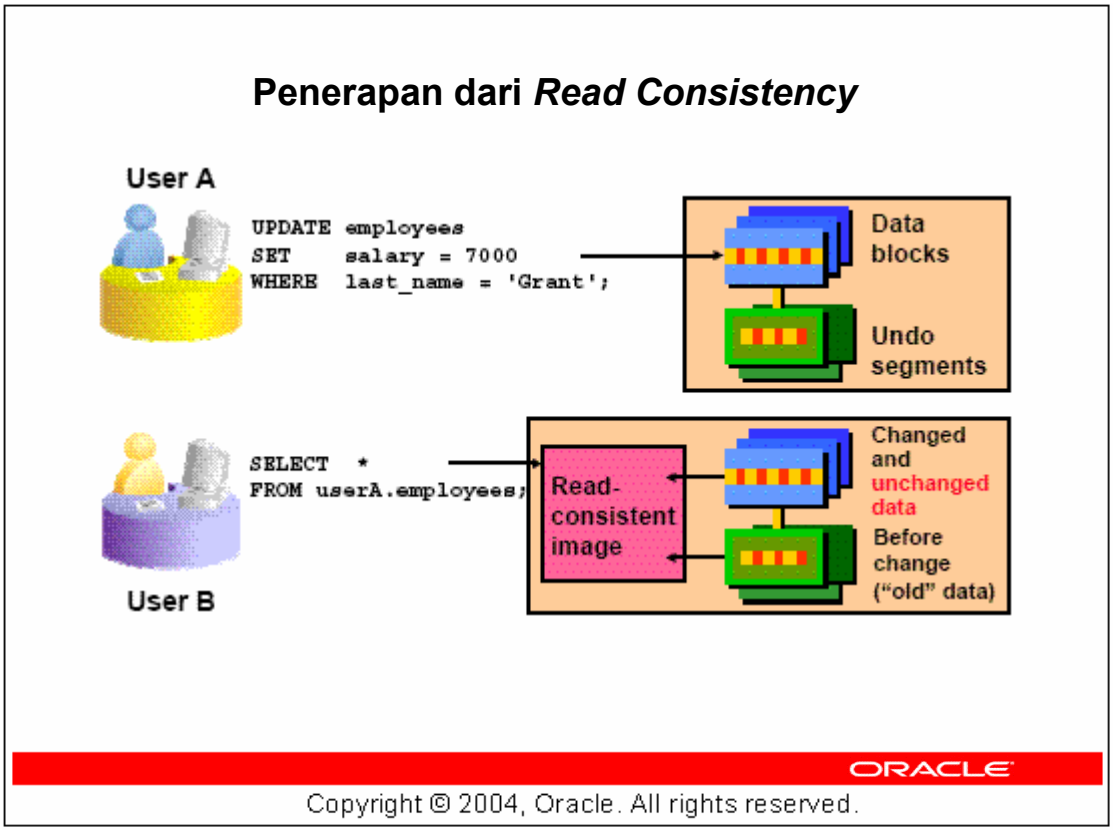
User-user Database mengakses database dengan dua cara :

- Membaca operasi-operasi (pernyataan `SELECT`)
- Menulis operasi-operasi (pernyataan-pernyataan `INSERT`, `UPDATE`, `DELETE`)

Anda perlu read consistency (konsistensi baca) sehingga hal-hal berikut ini terjadi :

- Pembaca dan penulis database percaya pada suatu tampilan yang konsisten dari data.
- Para pembaca tidak melihat data yang sedang dalam proses perubahan.
- Para penulis percaya bahwa perubahan-perubahan pada database dilakukan dengan cara yang konsisten.
- Perubahan-perubahan yang dilakukan oleh satu penulis tidak mengganggu atau bermasalah dengan perubahan-perubahan yang dibuat oleh penulis yang lain

Tujuan dari *read consistency* adalah untuk memastikan bahwa setiap user melihat data seperti apa adanya saat terakhir *commit*, sebelum sebuah operasi DML dijalankan



Penerapan dari *Read Consistency* (Konsistensi Baca)

Read consistency adalah suatu penerapan/implementasi yang otomatis. *Read consistency* menjaga sebagian salinan (*copy*) dari database dalam *undo segments*. Gambar *read-consistent* diatas adalah dikonstruksi dari data yang di-*commit* dari tabel dan data lama yang sedang dirubah serta belum di-*commit* dari *undo segment.*

Ketika operasi *insert*, *update* atau *delete* dilakukan terhadap database, server Oracle mengambil suatu salinan dari data sebelum data tersebut diubah dan ditulisnya pada suatu ***undo segment***.

Semua pembaca, kecuali yang melakukan perubahan, tetap melihat database seperti apa adanya sebelum perubahan dijalankan; mereka melihat "kilasan" data *undo segment* mereka sendiri.

Sebelum perubahan-perubahan di-*commit* ke database, hanya user yang melakukan perubahan yang dapat melihat database dengan perubahan. Setiap user lain melihat kilasan dalam *undo segment*. Hal ini menjamin bahwa para pembaca konsisten membaca data yang pada saat ini sedang akan dirubah.

Ketika suatu pernyataan DML di-*commit*, perubahan yang terjadi pada database akan dapat dilihat oleh semua user yang mengeluarkan suatu pernyataan *select* setelah perintah *commit* selesai dilakukan. Tempat yang digunakan oleh data lama dalam *file undo segment* akan dikosongkan untuk dapat digunakan kembali.

Jika transaksi di-roll back, perubahan yang tidak dilakukan:

- Data tetap Asli, bentuk lama dari data dalam *undo segment* ditulis kembali ke tabel.
- Semua user melihat database seperti apa adanya sebelum transaksi dilakukan.

Ringkasan

Dalam pelajaran ini, Anda sudah mempelajari bagaimana untuk menggunakan pernyataan-pernyataan berikut :

Fungsi	Keterangan
INSERT	Menambah suatu baris baru pada tabel
UPDATE	Merubah baris-baris yang ada dalam tabel
DELETE	Menghapus baris-baris yang ada dari suatu tabel
COMMIT	Membuat perubahan sementara menjadi permanen
SAVEPOINT	Digunakan untuk *roll back* ke tanda *savepoint*
ROLLBACK	Membatalkan semua perubahan-perubahan data sementara

ORACLE

Ringkasan

Pada pelajaran ini, Anda sudah mempelajari bagaimana untuk merubah data pada database Oracle dengan menggunakan pernyataan-pernyataan `INSERT`, `UPDATE` dan `DELETE`, begitu juga bagaimana untuk mengontrol perubahan-perubahan data dengan menggunakan pernyataan-pernyataan `COMMIT`, `SAVEPOINT`, dan `ROLLBACK`.

Server Oracle manjamin suatu tampilan data yang konsisten setiap waktu.