PERTEMUAN 9
MANIPULASI DATA

Tujuan Pembelajaran :

- Memahami Statement DML (Data Manipulation Language)
- Menyisipkan baris ke dalam table
- Merubah baris dalam table
- Menghapus baris dari table
- Mengontrol transaksi

TEORI DAN PERCOBAAN

9.1. Data Manipulation Language

Data Manipulation Language (DML) adalah suatu statement yang dijalankan pada saat kita memerlukan :

- penambahan baris baru pada table
- memodifikasi baris yang ada pada table
- menghapus baris yang ada pada table

DML Statement identik dengan operasi INSERT, MODIFY dan DELETE.

Istilah Transaksi mengandung pengertian kumpulan Statement DML yang membentuk suatu fungsi tertentu.

9.2. Menambahkan Baris Baru ke dalam Tabel (INSERT)

Menambahkan baris baru ke dalam table menggunakan perintah INSERT.

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

Percobaan 1 :

Menyisipkan baris ke dalam table DEPT.

```
SQL> INSERT INTO dept(deptno,dname,loc)
  2  VALUES (50,'DEVELOPMENT','DETROIT');

1 row created.
```

9.3. INSERT dengan nilai NULL

Kolom yang tidak disebutkan dalam perintah INSERT INTO secara otomatis akan diisi dengan nilai NULL.

Percobaan 2 :

Seperti pada contoh berikut kolom *loc* akan berisi nilai NULL

```
SQL> INSERT INTO dept(deptno,dname)
  2  VALUES (60,'MIS');

1 row created.
```

Percobaan 3:

Kalau tidak disebutkan kolom apa saja yang harus diisi, maka nilai pada VALUES harus mencantumkan semua kolom yang ada pada table sesuai dengan urutannya.

```
SQL> INSERT INTO dept
  2  VALUES (70,'FINANCE', NULL);

1 row created.
```

9.4. INSERT menggunakan Fungsi dan Nilai terformat

Suatu fungsi bisa digunakan sebagai suatu nilai dalam perintah INSERT.

Percobaan 4:

Contoh berikut ini menyertakan fungsi *sysdate* ke dalam table EMP

```
SQL> INSERT INTO emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
  2  VALUES (7196, 'GREEN', 'SALESMAN', 7782, sysdate, 2000, NULL, 10);

1 row created.
```

Nilai yang dimasukkan ke dalam perintah INSERT bisa menggunakan nilai terformat.

Percobaan 5 :

Contoh berikut ini menggunakan nilai data tanggal yang diformat pada perintah INSERT.

```
SQL> INSERT INTO emp
  2  VALUES (2296, 'AROMANO','SALESMAN',7782,TO_DATE('FEB 3 97','MON DD YY'),1300,NULL,10);

1 row created.
```

9.5. INSERT dengan variable substitusi

Pada perintah INSERT bisa dicantumkan variable substitusi.

Percobaan 5 : Berikut ini akan dicantumkan variable substitusi dari suatu nilai kolom pada baris baru yang akan dimasukkan ke dalam table dengan perintah INSERT.

```
SQL> INSERT INTO dept(deptno,dname,loc)
  2  VALUES (&department_id, '&department_name', '&location');
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

9.6. Pembuatan Script dengan Kustomisasi Prompt

Untuk melakukan kustomisasi, dapat digunakan perintah ACCEPT dan PROMPT.

Percobaan 6 : Sama seperti percobaan 5, dengan penambahan kustomisasi prompt.

```
SQL>  @d:\tessy2003\coba6.sql
Silahkan masukkan nomer department : 90
Silahkan masukkan nama department : PAYROLL
Silahkan masukkan lokasi department : HOUSTON

1 row created.
```

Isi dari file coba6.sql :

```
ACCEPT department_id PROMPT 'Silahkan masukkan nomer department : '
ACCEPT department_name PROMPT 'Silahkan masukkan nama department : '
ACCEPT location PROMPT 'Silahkan masukkan lokasi department : '

INSERT INTO dept(deptno,dname,loc)
VALUES (&department_id,'&department_name','&location');
```

9.7. Mengkopi Baris dari Tabel lain

Perintah INSERT juga bisa digunakan untuk mengkopi baris data yang berasal dari table yang lain.

Percobaan 7 : Berikut ini akan ditambahkan baris baru ke dalam table manager yang berasal dari table pegawai yang pekerjaannya = 'MANAGER'

```
SQL> INSERT INTO managers(id,name,salary,hiredate)
  2              SELECT empno,ename,sal,hiredate
  3              FROM emp
  4              WHERE job='MANAGER';

3 rows created.
```

9.8. Perubahan Data dalam Tabel (UPDATE)

Untuk memodifikasi baris data yang ada pada table digunakan perintah UPDATE.

Sintak dari perintah UPDATE :

UPDATE ***table***
SET ***column = value [, column = value, …]***
[WHERE ***condition];***

Percobaan 7 : Ubah nomer department menjadi = 20, untuk pegawai yang memiliki nomer department = 7782;

```
SQL> UPDATE emp
  2  SET deptno=20
  3  WHERE empno=7782;

1 row updated.
```

Semua baris pada table akan dimodifikasi jika klausa WHERE tidak disertakan.

```
SQL>  UPDATE emp
  2   SET deptno=20;

14 rows updated.
```

Hasilnya, jika diperiksa :

```
SQL> SELECT ename, deptno FROM emp;

ENAME          DEPTNO
---------- ----------
ALLEN              20
WARD               20
JONES              20
MARTIN             20
BLAKE              20
CLARK              20
SCOTT              20

.....
```

9.9. UPDATE dengan multiple column subquery

Perintah UPDATE bisa menggunakan multiple column subquery.

Percobaan 8 : Ubah nilai dari kolom pekerjaan dan nomer department dari pegawai dengan nomer pegawai '7698' sehingga nilainya sama dengan jenis pekerjaan dan nomer department yang dimiliki oleh pegawai bernomer '7499'

```
SQL> UPDATE EMP
  2  SET (job,deptno) = (SELECT job,deptno
  3                      FROM EMP
  4                      WHERE empno=7499)
  5  WHERE empno=7698;

1 row updated.
```

9.10. UPDATE berdasarkan table yang lain

UPDATE berdasarkan table yang lain artinya perubahan pada sebuah table dimana kondisi perubahannya ditentukan berdasarkan nilai yang terdapat pada table yang lain.

Percobaan 9 : Ubah data nomer department yang dimiliki oleh pegawai dengan pekerjaan yang sama dengan pegawai bernomor '7788' , data nomer department tersebut harus diubah menjadi data yang sama dengan nomer department yang dimiliki oleh pegawai bernomor '7788';

```
SQL> UPDATE emp
  2  SET deptno = (SELECT deptno
  3                FROM emp
  4                WHERE empno=7788)
  5  WHERE job = (SELECT job
  6               FROM emp
  7               WHERE empno=7788)
  8  ;

2 rows updated.
```

9.11. Kesalahan pada UPDATE

Salah satu kesalahan pada perintah UPDATE, misal jika kita berusaha untuk merubah data sedangkan data tersebut terikat pada integrity constraint (merupakan suatu key).

Percobaan 10 : Ubah data pegawai dengan nomer department 10 menjadi data dengan nomer department 55.

```
SQL> update emp
  2  set deptno=55
  3  where deptno=10;
update emp
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.EMP_FOREIGN_KEY) violated - parent key
not found
```

terjadi kesalahan dikarenakan nomer department '55' tidak terdapat dalam table pegawai. Query berikut ini baru bisa dijalankan tanpa ada kesalahan :

```
SQL> UPDATE EMP
  2  SET deptno=20
  3  WHERE deptno=10;

3 rows updated.
```

9.12. Menghapus baris dari table (DELETE)

Baris data yang ada pada table dapat dihapus dengan menggunakan perintah DELETE.

Percobaan 11 : Menghapus data pegawai yang mempunyai nama department = 'DEVELOPMENT'

```
SQL> DELETE FROM dept
  2  WHERE dname='DEVELOPMENT';
```

Percobaan 12 : Hapus data semua pegawai yang tanggal mulai bekerjanya setelah tanggal 1 Januari 1997

```
SQL> DELETE FROM EMP
  2  WHERE hiredate > TO_DATE('01-01-97','DD-MM-YY');

0 rows deleted.
```

Jika klausa WHERE ditiadakan, maka semua baris dalam table akan dihapus.

Percobaan 13 : Menghapus semua baris yang ada pada table pegawai.

```
SQL> delete from emp;

14 rows deleted.
```

9.13. DELETE berdasarkan table yang lain

Subquery dapat digunakan dalam statement DELETE untuk menghapus baris pada suatu table berdasarkan data yang ada di table yang lain.

Percobaan 14 : Menghapus data pada table pegawai, untuk pegawai yang bekerja pada department 'SALES'

```
SQL> DELETE FROM EMP
  2  WHERE deptno = (SELECT deptno
  3                  FROM dept
  4                  WHERE dname='SALES');

6 rows deleted.
```

9.14. Kesalahan pada DELETE

Jika baris data yang dihapus pada table berkaitan dengan integrity constraint, maka akan terjadi kesalahan.

Percobaan 15 : Misal akan dihapus data pada table department yang memiliki nomer department '10'. Jika nomer department 10 ini mempunyai data yang berkaitan dengan data yang ada di table pegawai (ada pegawai yang bekerja di department 10), maka akan muncul pesan kesalahan, seperti pada query berikut :

```
SQL> DELETE FROM dept
  2  WHERE deptno=10;
DELETE FROM dept
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.EMP_FOREIGN_KEY) violated - child record
found
```

9.15. Transaksi Database

Jika DML berkaitan dengan manipulasi data pada table, maka DDL (Data Definition Language) berkaitan dengan pendefinisian table, sedangkan DCL (Data Control Language) berkaitan kontrol transaksi. DDL dan DCL secara otomatis akan di-commit (dilakukan perubahan secara permanen) pada akhir dari transaksi.

9.16. Statement COMMIT dan ROLLBACK

Ada 2 statement DCL yang penting yaitu COMMIT dan ROLLBACK, selain dari itu ada SAVEPOINT. Perintah COMMIT menandai perubahan secara permanen

pada data. Sedangkan ROLLBACK mengembalikan keadaan sesuai dengan titik (keadaan) yang ditandai dengan SAVEPOINT, atau jika ROLLBACK tidak diberi parameter maka keadaan akan dikembalikan pada titik perubahan yang terakhir.

9.17. Pemrosesan Transaksi secara Implisit

Transaksi akan diproses secara implicit atau dilakukan operasi COMMIT secara otomatis, untuk keadaan berikut :

- Setelah Statement DDL diberikan
- Setelah Statement DCL diberikan
- Proses exit secara normal dari SQL*PLUS.

Sedangkan perintah ROLLBACK secara otomatis akan dijalankan jika terjadi kondisi yang abnormal atau terjadi *system failure.*

LATIHAN SOAL

1. Buat table MY_EMPLOYEE yang mempunyai struktur sebagai berikut :

```
Name                                      Null?    Type
----------------------------------------- -------- ------------
ID                                        NOT NULL NUMBER(4)
LAST_NAME                                          VARCHAR2(25)
FIRST_NAME                                         VARCHAR2(25)
USERID                                             VARCHAR2(8)
SALARY                                             NUMBER(9,2)
```

2. Tambahkan baris data berikut ke dalam table MY_EMPLOYEE, sehingga jika ditampilkan akan tampak listing data table sebagai berikut :

```
   ID LAST_NAME  FIRST_NAME                USERID       SALARY
----- ---------- ------------------------- -------- ----------
    1 Patel      Ralph                     rpatel          795
    2 Dancs      Betty                     bdancs          860
    3 Biri       Ben                       bbiri          1100
    4 Newman     Chad                      cnewman         750
    5 Ropeburn   Audry                     aropebur       1550
```

3. Buat penambahan baris data ke dalam table menjadi permanen dengan menggunakan perintah COMMIT;

Sebelum di-COMMIT, untuk membuktikan bahwa penambahan data belum permanen, buka SQL*PLUS lagi tanpa menutup SQL*PLUS yang masih dibuka, kemudian dari SQL*PLUS yang baru beri perintah :
SELECT * FROM MY_EMPLOYEE;
Maka akan terlihat bahwa table masih kosong. Tabel baru berisi jika perintah COMMIT sudah diberikan atau kita keluar secara normal dari SQL*PLUS tempat baris data ditambahkan.

4. Ubah nama akhir dari pegawai bernomer 3 menjadi 'Drexler'
5. Ubah gaji menjadi 1000 untuk semua pegawai yang gajinya kurang dari 900

6. Periksa perubahan yang dibuat pada soal no 4 dan 5.

```
LAST_NAME      SALARY
---------- ----------
Patel            1000
Dancs            1000
Drexler          1100
Newman           1000
Ropeburn         1550
```

7. Delete pegawai dengan nama 'Betty Dancs', kemudian periksa hasilnya :

```
  ID LAST_NAME  FIRST_NAME                USERID       SALARY
---- ---------- ------------------------- -------- ----------
   1 Patel      Ralph                     rpatel         1000
   3 Drexler    Ben                       bbiri          1100
   4 Newman     Chad                      cnewman        1000
   5 Ropeburn   Audry                     aropebur       1550
```

8. Simpan semua perubahan (DML) dengan memberikan perintah COMMIT;
9. Beri tanda SAVEPOINT sini;
10. Setelah itu hapus semua data dalam table MY_EMPLOYEE
11. Periksa hasilnya dengan me-list semua isi tabel
12. Batalkan penghapusan dengan memberikan perintah ROLLBACK sini;
13. Periksa hasilnya dengan me-list semua isi table. Maka data pada table akan terlihat kembali.