Oracle Academic Initiative

Oracle9i Introduction to SQL

Oleh:

Tessy Badriyah, SKom.MT

Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Surabaya

BAB 9 : Manipulasi Data

9.1. Sasaran

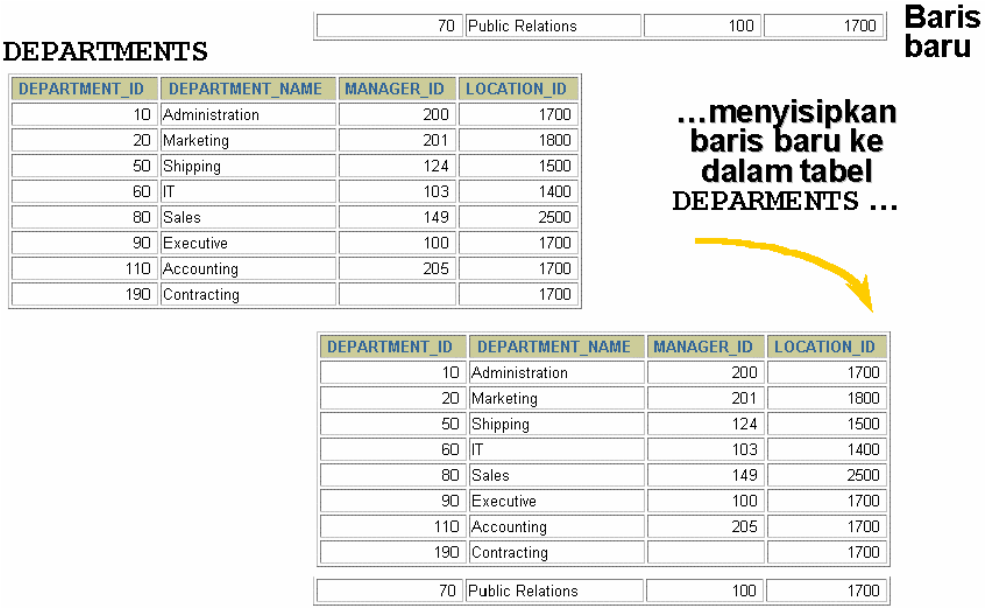
- Memahami Statement DML (Data Manipulation Language)
- Menyisipkan baris ke dalam table
- Merubah baris dalam table
- Menghapus baris dari table
- Mengontrol transaksi

9.2. 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.



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

9.3. Menambahkan Baris Baru ke dalam Tabel

Menambahkan baris baru ke dalam table menggunakan perintah INSERT.

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

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

9.4. Menambahkan Baris dengan Nilai NULL

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

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

Atau secara eksplisit memasukkan nilai NULL ke suatu kolom :

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

9.5. Menambahkan Nilai Khusus

Fungsi SYSDATE menyimpan waktu dan tanggal saat ini.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES                (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 100);
1 row created.
```

9.6. Menambahkan Nilai Tanggal Spesifik

Untuk menambahkan data yang memiliki tipe data tanggal dengan format yang spesifik :

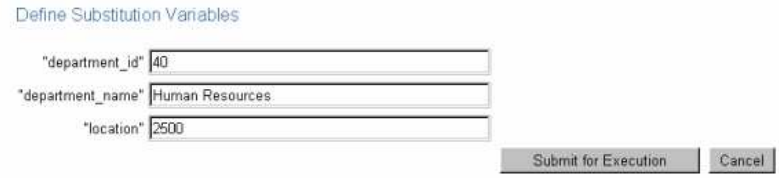
```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

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	

9.7. Pembuatan Script

Dapat juga digunakan variabel substitusi dengan & ke dalam INSERT :

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




```
1 row created.
```

9.8. Mengkopi Baris dari Tabel yang lain

Perintah INSERT juga bisa digunakan untuk mengkopi baris data yang berasal dari table yang lain. Berikut ini membuat tabel sales_reps dari tabel employees :

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

4 rows created.
```

9.9. Merubah data dalam Tabel

Ilustrasi perubahan data dalam tabel :

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_I
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Ubah/Update baris dalam tabel EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

9.10. Sintak Statement 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];
```

9.11. Mengupdate Baris dalam Tabel

Berikut contoh perintah UPDATE untuk mengubah satu baris saja :

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

Jika klausa WHERE dihilangkan, maka perintah UPDATE akan mengubah nilai kolom dari semua data atau record yang ada pada tabel :

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

9.12. Mengupdate dua kolom dengan subquery

Query berikut ini akan mengupdate job dan salary yang dimiliki oleh employee 114 supaya sama dengan job dan salary yang dimiliki oleh employee 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.
```

9.13. Mengupdate Berdasarkan Baris pada Tabel yang lain

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

9.14. Mengupdate Baris : terdapat kesalahan Integrity Constraint

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

```
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
```

Nomer departemen 55 tidak ada

9.15. Menghapus Baris dari Tabel

Ilustrasi untuk menghapus 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

Delete a row from the DEPARTMENTS table.

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

9.16. Statement DELETE

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

```
DELETE [FROM]   table
[WHERE          condition];
```

9.17. Menghapus Baris dari Tabel

Jika klausa WHERE disertakan pada perintah DELETE maka baris data tertentu yang akan dihapus hanya yang memenuhi kriteria pada WHERE :

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

Tapi jika klausa WHERE dihilangkan, maka semua baris data dalam tabel akan dihapus :

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

9.18. Menghapus Berdasarkan Baris pada Tabel yang lain

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

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

9.19. Menghapus Baris : terdapat kesalahan Integrity Constraint

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

```
DELETE FROM departments
WHERE       department_id = 60;
```

```
DELETE FROM departments
            *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

Kita tidak bisa menghapus baris yang berisi primary key yang digunakan sebagai foreign key di tabel yang lain

9.20. Menggunakan Subquery dalam Statement INSERT

Subquery dapat digunakan dalam statement INSERT :

```
INSERT INTO
        (SELECT employee_id, last_name,
                email, hire_date, job_id,
salary,
                department_id
         FROM   employees
         WHERE  department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000, 50);

1 row created.
```

9.21. Menggunakan Keyword WITH CHECK OPTION pada DML Statement

Subquery digunakan untuk mengidentifikasi tabel dan kolom dari statement DML. Dengan keyword WITH CHECK OPTION akan mencegah terjadinya perubahan baris yang tidak berada dalam subquery.

```
INSERT INTO  (SELECT employee_id, last_name, email,
                hire_date, job_id, salary
             FROM   employees
             WHERE  department_id = 50 WITH CHECK OPTION)
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);
INSERT INTO
         *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

9.22. Pendahuluan : Explicit Default Feature

Dengan menggunakan keyword DEFAULT secara eksplisit kita bisa menginisialisasi nilai dari suatu kolom. Perintah tersebut dapat digunakan dalam INSERT dan UPDATE Statement.

9.23. Penggunaan Explicit Default Feature

DEFAULT dengan INSERT :

```
INSERT INTO departments
  (department_id, department_name, manager_id)
VALUES (300, 'Engineering', DEFAULT);
```

DEFAULT dengan UPDATE :

```
UPDATE departments
SET manager_id = DEFAULT WHERE department_id = 10;
```

9.24. Statement MERGE

Statement MERGE digunakan untuk menggabungkan satu tabel dengan tabel yang lainnya. Perintah ini pada dasarnya digunakan untuk melakukan sinkronisasi data diantara dua tabel.
Sintak dari Statement MERGE :

```
MERGE INTO table_name table_alias
  USING (table|view|sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
    col1 = col_val1,
    col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

Contoh penggunaan Statement MERGE berikut ini untuk INSERT dan UPDATE baris data dalam tabel COPY_EMP yang sesuai dengan data yang ada pada tabel EMPLOYEES.

```
MERGE INTO copy_emp  c
   USING employees e
   ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
  UPDATE SET
      c.first_name     = e.first_name,
      c.last_name      = e.last_name,
      ...
      c.department_id  = e.department_id
WHEN NOT MATCHED THEN
  INSERT VALUES(e.employee_id, e.first_name, e.last_name,
          e.email, e.phone_number, e.hire_date, e.job_id,
          e.salary, e.commission_pct, e.manager_id,
          e.department_id);
```

9.25. Transaksi Database

Transaksi database berisi salah satu dari hal berikut :

- Statement DML untuk melakukan manipulasi terhadap data yang telah ada
- Statement DDL
- Statement DCL

Pada saat perintah DML selesai dijalankan, diikuti dengan salah satu event berikut ini :

- Diberikan perintah COMMIT atau ROLLBACK
- Menjalankan perintah DDL atau DCL (akan dilakukan COMMIT secara otomatis)
- User keluar dari iSQL*PLUS
- System crash

9.26. Keuntungan dari Statement COMMIT dan ROLLBACK

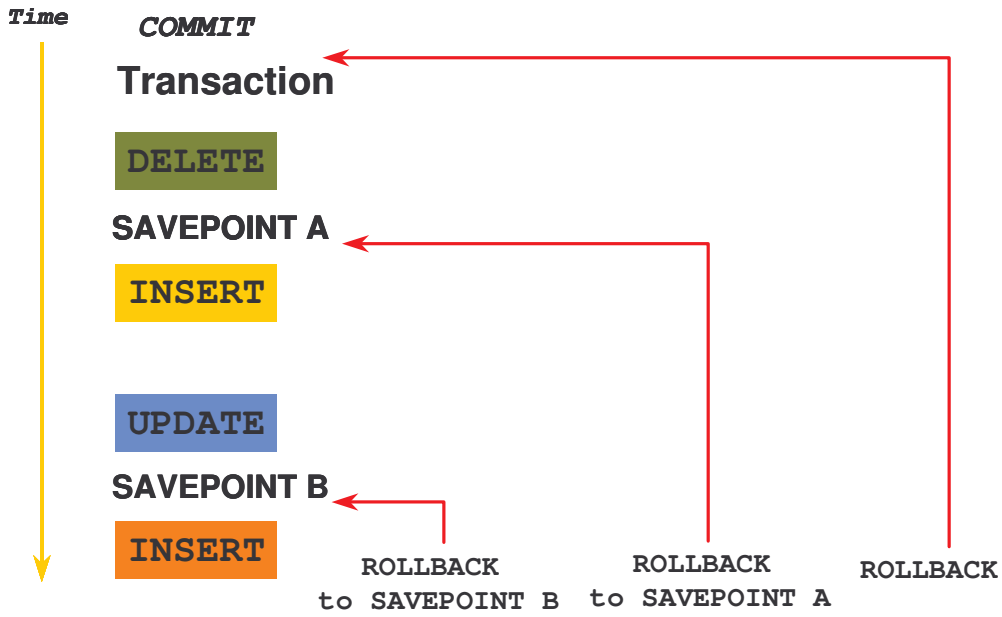
Ada 2 statement DCL yang penting yaitu COMMIT dan ROLLBACK, selain dari itu ada SAVEPOINT. Perintah COMMIT menandai perubahan secara permanen pada data. Sedangkan ROLLBACK mengembalikan keadaan sesuai dengan titik (keadaan) yang ditandai dengan SAVEPOINT, atau jika ROLLBACK tidak diberi parameter maka keadaan akan dikembalikan pada titik perubahan yang terakhir.

Dengan menggunakan COMMIT dan ROLLBACK maka :

- Dapat dipastikan konsistensi data
- Dapat ditampilkan perubahan data sebelum membuat perubahan tersebut menjadi permanen.
- Dapat dilakukan pengelompokan secara logika operasi-operasi yang berelasi

9.27. Pengontrolan Transaksi

Berikut ini contoh pengontrolan transaksi dengan perintah COMMIT, ROLLBACK dan SAVEPOINT



9.28. Melakukan ROLLBACK ke SAVEPOINT

Perintah ROLLBACK dapat melakukan *rolling back* menuju ke suatu titik yang ditandai dengan SAVEPOINT.

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

9.29. 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.*

9.30. Status Data sebelum COMMIT atau ROLLBACK

Status data sebelum COMMIT atau ROLLBACK :

- Current user dapat menampilkan hasil dari operasi DML melalui statement SELECT
- Sedangkan user yang lain tidak dapat menampilkan hasil dari operasi DML
- Baris yang berpengaruh akibat dari perintah DML akan di-*locked,* sehingga user lain tidak dapat melakukan perubahan terhadap baris tersebut.

9.31. Status Data setelah COMMIT

Status data setelah diberi COMMIT :

- Data berubah secara permanen
- Keadaan awal sebelum data permanen akan hilang
- Semua user dapat melihat hasilnya
- Penguncian (*lock)* pada baris data akan dilepas, sehingga baris data tersebut *available* untuk semua user
- Semua savepoints dihapus

9.32. Meng-COMMIT-kan Data

Misal dibuat perubahan data sebagai berikut :

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 row deleted.

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 row inserted.
```

Kemudian perubahan dibuat permanen dengan perintah COMMIT :

```
COMMIT;
Commit complete.
```

9.33. Status Data setelah ROLLBACK

Status data setelah ROLLBACK :

- Perubahan data tidak dilakukan
- Status sebelum data diubah akan di*restore*
- Penguncian terhadap baris data akan dilepas

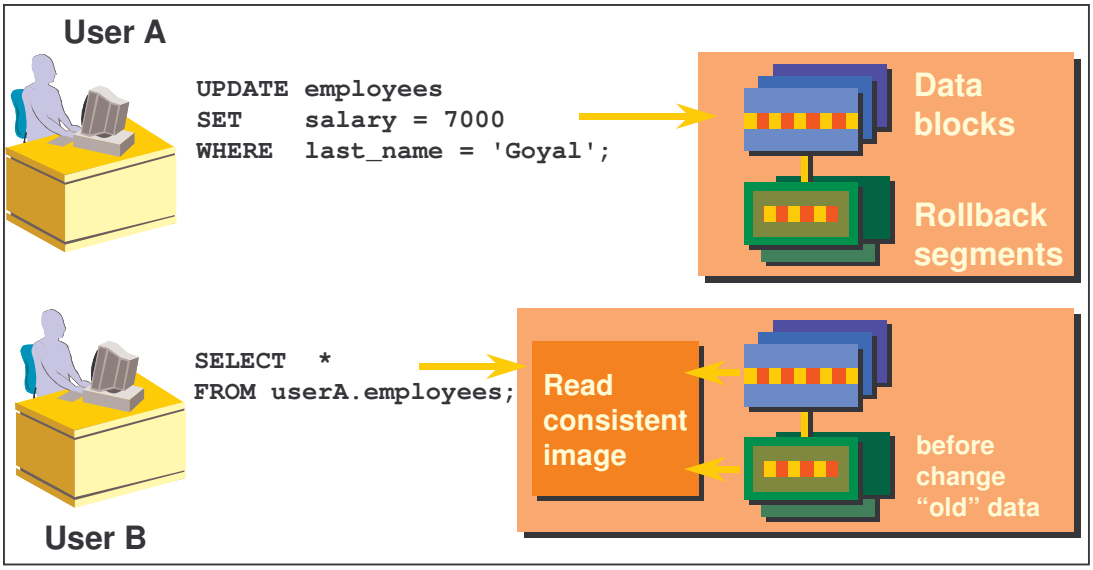
Contoh perintah ROLLBACK :

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

9.34. Konsistensi Pembacaan (Read Consistency)

Konsistensi pembacaan menjamin tampilan yang konsisten untuk semua data setiap saat. Perubahan yang dibuat oleh seorang user tidak akan konflik dengan perubahan yang dibuat oleh user yang lain.

9.35. Implementasi Konsistensi Pembacaan



9.36. Penguncian (Locking)

Pada database Oracle, penguncian atau *locking* berarti :

- Mencegah terjadinya interaksi yang destruktif diantara transaksi yang bersamaan
- Tidak memerlukan aksi dari user
- Ditangani selama terjadinya transaksi
- Ada dua tipe yaitu : explicit locking dan implicit locking.

9.37. Implicit Locking

Ada dua model lock :

- Exclusive : mengunci sama sekali
- Share : masih mengijinkan user lain untuk mengakses

High level dari data concurrency :

- DML : Table share, row exclusive
- Queries : tidak memerlukan lock
- DDL : melindungi dari definisi object.

Perintah Lock ditangani sampai diberikan perintah commit atau rollback.

9.38. Latihan

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.