Pertemuan 1 : Berinteraksi dengan Oracle Server

Tujuan pembelajaran

- Memahami penulisan SELECT di PL/SQL
- Memahami penulisan statement DML di PL/SQL
- Memahami control transaksi di PL/SQL

SQL statement di PL/SQL

- Perintah SELECT digunakan untuk mendapatkan baris data dalam tabel yang ada di database
- Perubahan data dapat menggunakan DML statement
- Kontrol transaksi dapat dilakukan dengan perintah COMMIT, ROLLBACK dan SAVEPOINT
- Menentukan hasil operasi DML dengan atribut implicit cursor.

SELECT statement

- Sintaks

```
SELECT  select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
[WHERE  condition];
```

- Contoh:

```
DECLARE
  v_deptno           NUMBER(4);
  v_location_id      NUMBER(4);
BEGIN
  SELECT      department_id, location_id
  INTO        v_deptno, v_location_id
  FROM        departments
  WHERE       department_name = 'Sales';
  ...
END;
/
```

```
DECLARE
  v_hire_date    employees.hire_date%TYPE;
  v_salary       employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      v_hire_date, v_salary
  FROM      employees
  WHERE     employee_id = 100;
  ...
END;
/
```

Contoh SELECT statement

- Apa outputnya?

```
SET SERVEROUTPUT ON
DECLARE
  v_sum_sal   NUMBER(10,2);
  v_deptno     NUMBER NOT NULL := 60;
BEGIN
  SELECT      SUM(salary)  -- group function
  INTO        v_sum_sal
  FROM        employees
  WHERE       department_id = v_deptno;
  DBMS_OUTPUT.PUT_LINE ('The sum salary is ' ||
                        TO_CHAR(v_sum_sal));
END;
/
```

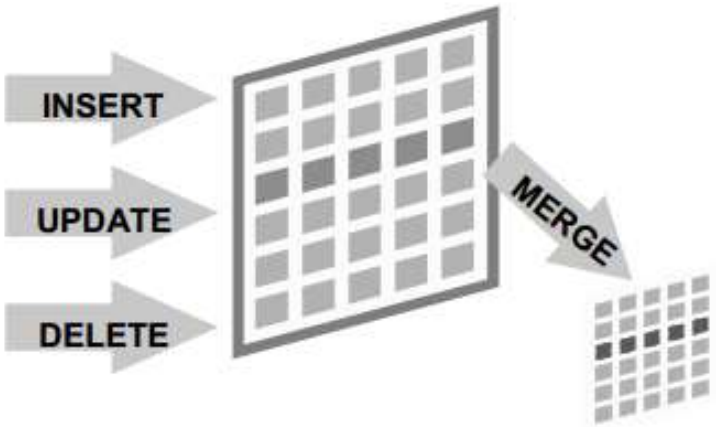
Contoh SELECT statement

- Jika nama variable sama dengan nama kolom dari tabel. Error atau tidak?

```
DECLARE
  hire_date      employees.hire_date%TYPE;
  sysdate        hire_date%TYPE;
  employee_id    employees.employee_id%TYPE := 176;
BEGIN
  SELECT      hire_date, sysdate
  INTO        hire_date, sysdate
  FROM        employees
  WHERE       employee_id = employee_id;
END;
/
```

DML statement dengan PL/SQL

- Merubah data di database dengan DML statement



Menyisipkan data baru

```
BEGIN
  INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
  VALUES
   (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES',
    sysdate, 'AD_ASST', 4000);
END;
/
```

Merubah data

```
DECLARE
  v_sal_increase   employees.salary%TYPE := 800;
BEGIN
  UPDATE      employees
  SET         salary = salary + v_sal_increase
  WHERE       job_id = 'ST_CLERK';

END;
/
```

Menghapus data

```
DECLARE
  v_deptno    employees.department_id%TYPE := 10;
BEGIN
  DELETE FROM    employees
  WHERE          department_id = v_deptno;
END;
/
```

MERGING rows

- Statement MERGE menyisipkan data baru dan merubah data pada suatu tabel berdasarkan data dari tabel lain

```
DECLARE
      v_empno EMPLOYEES.EMPLOYEE_ID%TYPE := 100;
BEGIN
MERGE INTO copy_emp c
     USING employees e
     ON (c.employee_id = v_empno)
   WHEN MATCHED THEN
     UPDATE SET
       c.first_name     = e.first_name,
       c.last_name      = e.last_name,
       c.email          = e.email,
       c.phone_number   = e.phone_number,
       c.hire_date      = e.hire_date,
       c.job_id         = e.job_id,
       c.salary         = e.salary,
       c.commission_pct = e.commission_pct,
       c.manager_id     = e.manager_id,
       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);
END;
/
```

Cara pemberian nama yang disarankan

Identifier	Naming Convention	Example
Variable	v_name	v_sal
Constant	c_name	c_company_name
Cursor	name_cursor	emp_cursor
Exception	e_name	e_too_many
Table type	name_table_type	amount_table_type
Table	name_table	countries
Record type	name_record_type	emp_record_type
Record	name_record	customer_record
*i*SQL*Plus substitution variable (also referred to as substitution parameter)	p_name	p_sal
*i*SQL*Plus host or bind variable	g_name	g_year_sal

SQL Cursor

- Oracle server menggunakan SQL cursor untuk melakukan parsing dan menjalankan SQL statement
- Ada dua tipe cursor:
  - Implicit
  - Explicit

Atribut dari SQL Cursor

- Dengan menggunakan atribut dari SQL cursor, kita bisa tahu output dari SQL statement
  - SQL%ROWCOUNT
    - Jumlah baris yang dihasilkan
  - SQL%FOUND
    - Jika dihasilkan satu atau lebih baris data
  - SQL%NOTFOUND
    - Jika tidak ditemukan data
  - SQL%ISOPEN
    - Selalu bernilai FALSE karena PL/SQL akan langsung menutup implicit cursor begitu mereka selesai dijalankan.

Contoh penggunaan SQL%ROWCOUNT

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
  v_employee_id employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM  employees
  WHERE        employee_id = v_employee_id;
  :rows_deleted := (SQL%ROWCOUNT ||
                        ' row deleted.');
END;
/
PRINT rows_deleted
```

Kontrol Transaksi

- Dengan menggunakan COMMIT, ROLLBACK dan SAVEPOINT kita bisa melakukan kontrol transaksi.
- Perintah tersebut diletakkan di dalam PL/SQL block.

Ringkasan

- Telah dipelajari SQL statement di dalam PL/SQL block dengan menggunakan perintah:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - MERGE
- Penggunaan kontrol transaksi COMMIT, ROLLBACK dan SAVEPOINT dalam PL/SQL block