www.itsci.mju.ac.th/sayan

LEC 06: PL/SQL

SAYAN UNANKARD
1/2558

ABOUT PL/SQL

PL/SQL:

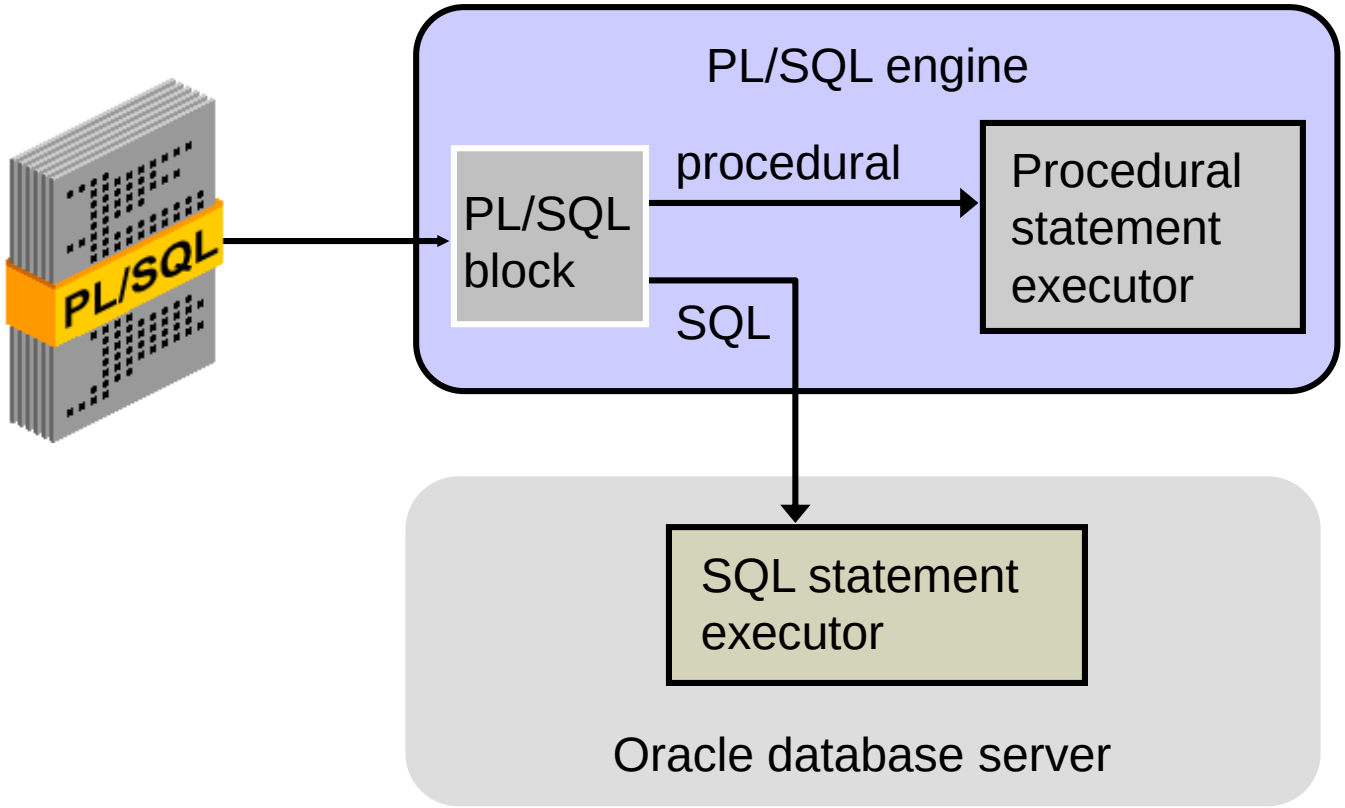
- ย่อมาจาก “Procedural Language extension to SQL”
- เป็นมาตรฐานของบริษัทออราเคิล ในการเข้าถึงข้อมูลด้วยภาษาเชิงโปรแกรม
- Seamlessly integrates procedural constructs with SQL

ABOUT PL/SQL

PL/SQL:

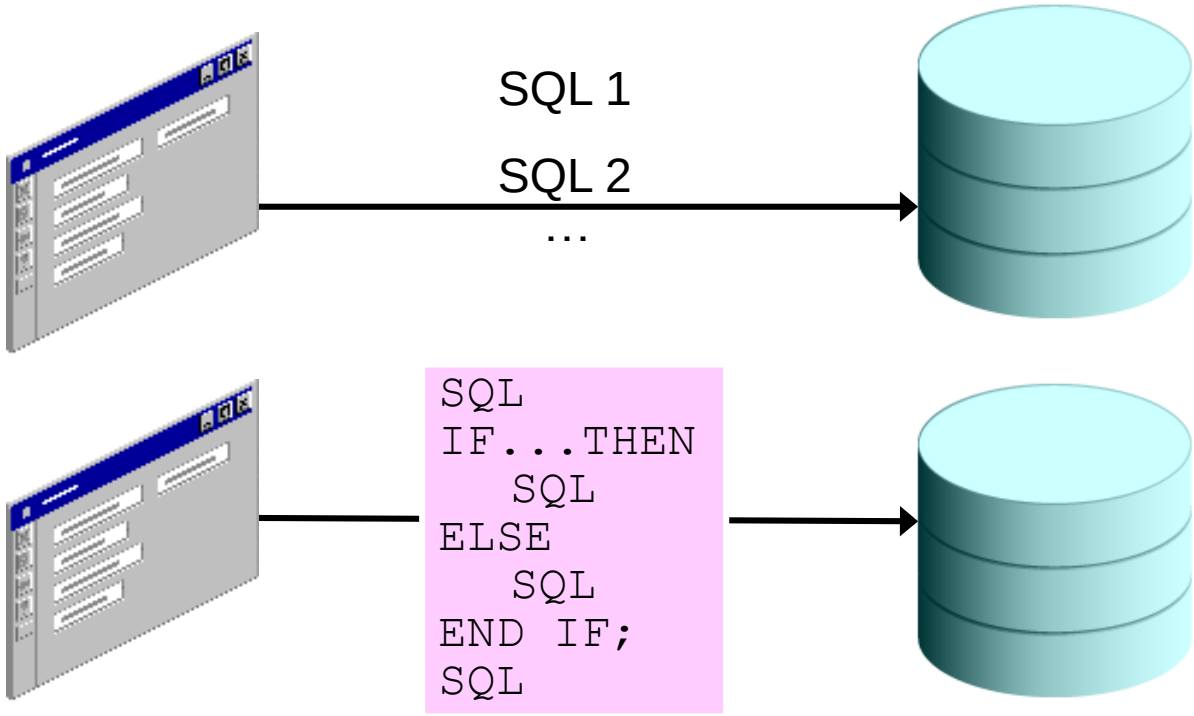
- เป็นการพัฒนาโปรแกรมในฝั่งของฐานข้อมูล
- ทำให้โค้ดแก้ไขได้ง่ายขึ้น เนื่องจากอยู่ในโครงสร้างที่เป็นมาตรฐาน
- ประกอบด้วยโครงสร้าง
  - ตัวแปร ค่าคงที่ และ ชนิดข้อมูล
  - โครงสร้างควบคุม เช่น เงื่อนไข และ การวนซ้ำ
  - ส่วนของโปรแกรมที่สามารถนำมาใช้ใหม่ได้

PL/SQL ENVIRONMENT



BENEFITS OF PL/SQL

- รวมโครงสร้างของโปรแกรมเข้ากับคำสั่ง SQL
- ปรับปรุงประสิทธิภาพของโปรแกรม



BENEFITS OF PL/SQL

ประโยชน์ของ PL/SQL

- เป็นการพัฒนาโปรแกรมเป็นโมดูล
- สามารถใช้งานรวมกับเครื่องมือของ Oracle ได้
- เคลื่อนย้าย และ แก้ไขง่าย
- มีการจัดการข้อผิดพลาดของโปรแกรม

PL/SQL BLOCK STRUCTURE

โครงสร้างของชุดคำสั่ง PL/SQL

- DECLARE (optional)
  - Variables, cursors, user-defined exceptions
- BEGIN (mandatory)
  - SQL statements
  - PL/SQL statements
- EXCEPTION (optional)
  - Actions to perform when errors occur
- END; (mandatory)

BLOCK TYPES

Anonymous

```
[DECLARE]


BEGIN
  --statements

[EXCEPTION]

END;
```

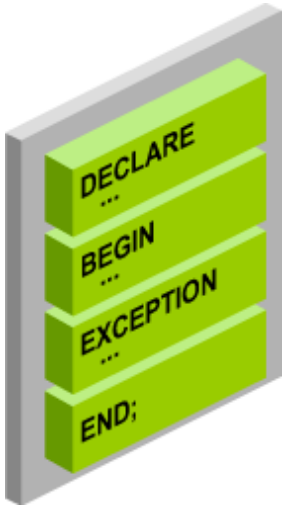
Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```

PROGRAM CONSTRUCTS

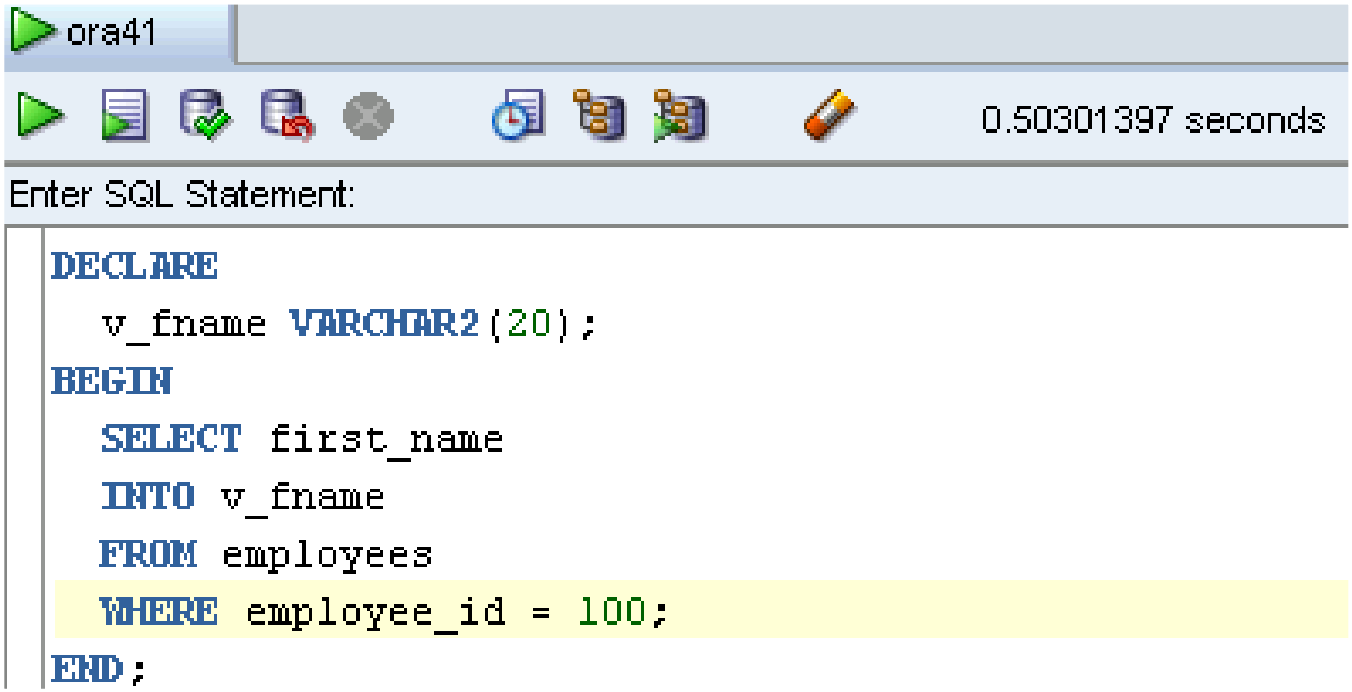


Tools Constructs
Anonymous blocks
Application procedures or functions
Application packages
Application triggers
Object types

Database Server Constructs
Anonymous blocks
Stored procedures or functions
Stored packages
Database triggers
Object types

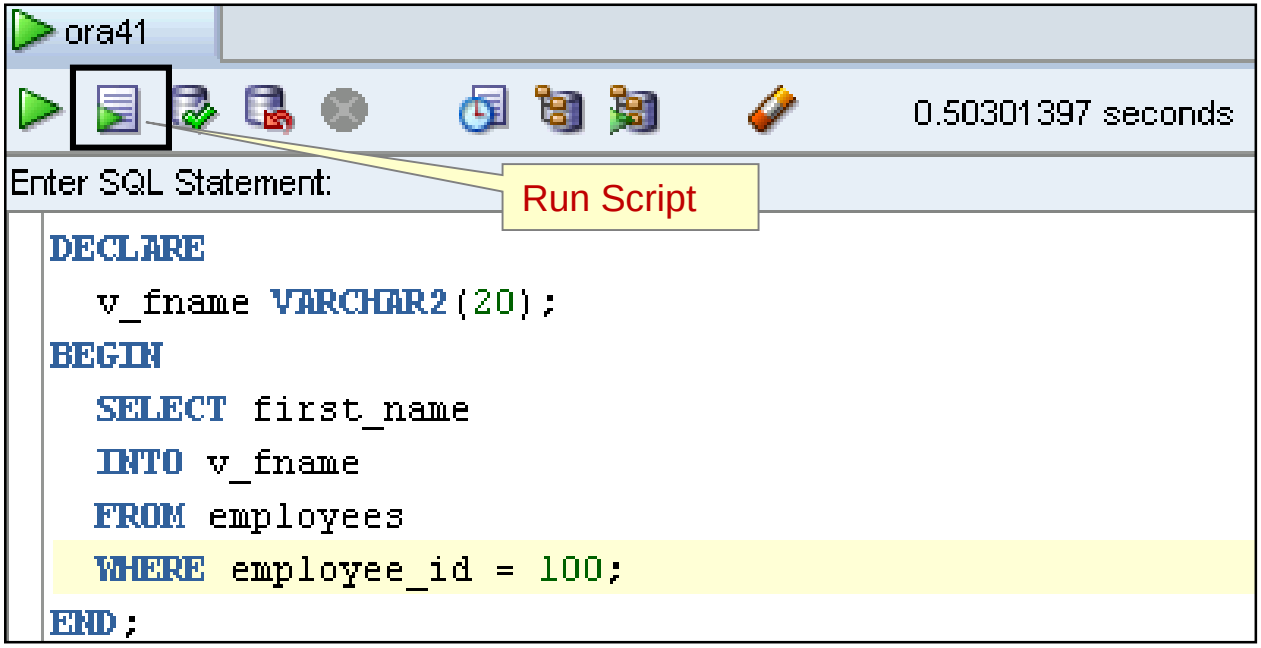
CREATE AN ANONYMOUS BLOCK

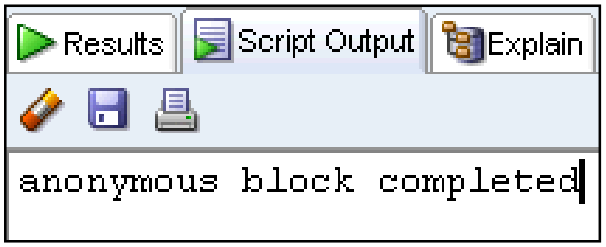
เขียนชุดคำสั่ง anonymous block ใน SQL Developer workspace:



EXECUTE AN ANONYMOUS BLOCK

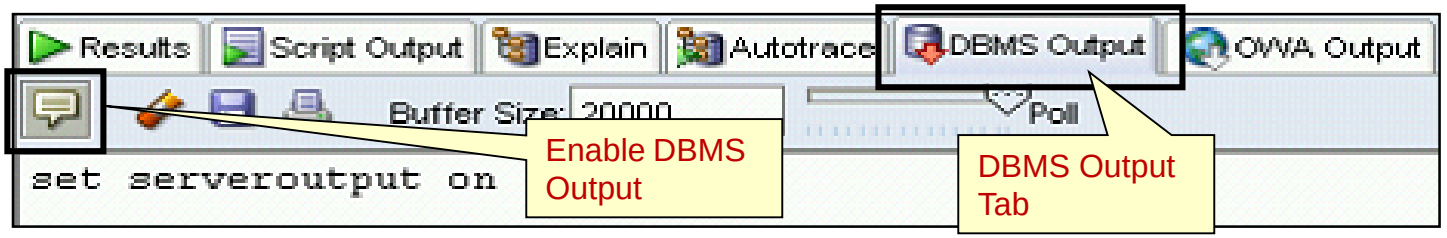
คลิกที่ปุ่ม Run Script เพื่อประมวลผลชุดคำสั่ง anonymous block





TEST THE OUTPUT OF A PL/SQL BLOCK

การกำหนดให้แสดงผลลัพธ์ใน SQL Developer โดยคลิกปุ่ม Enable DBMS Output บน DBMS Output tab

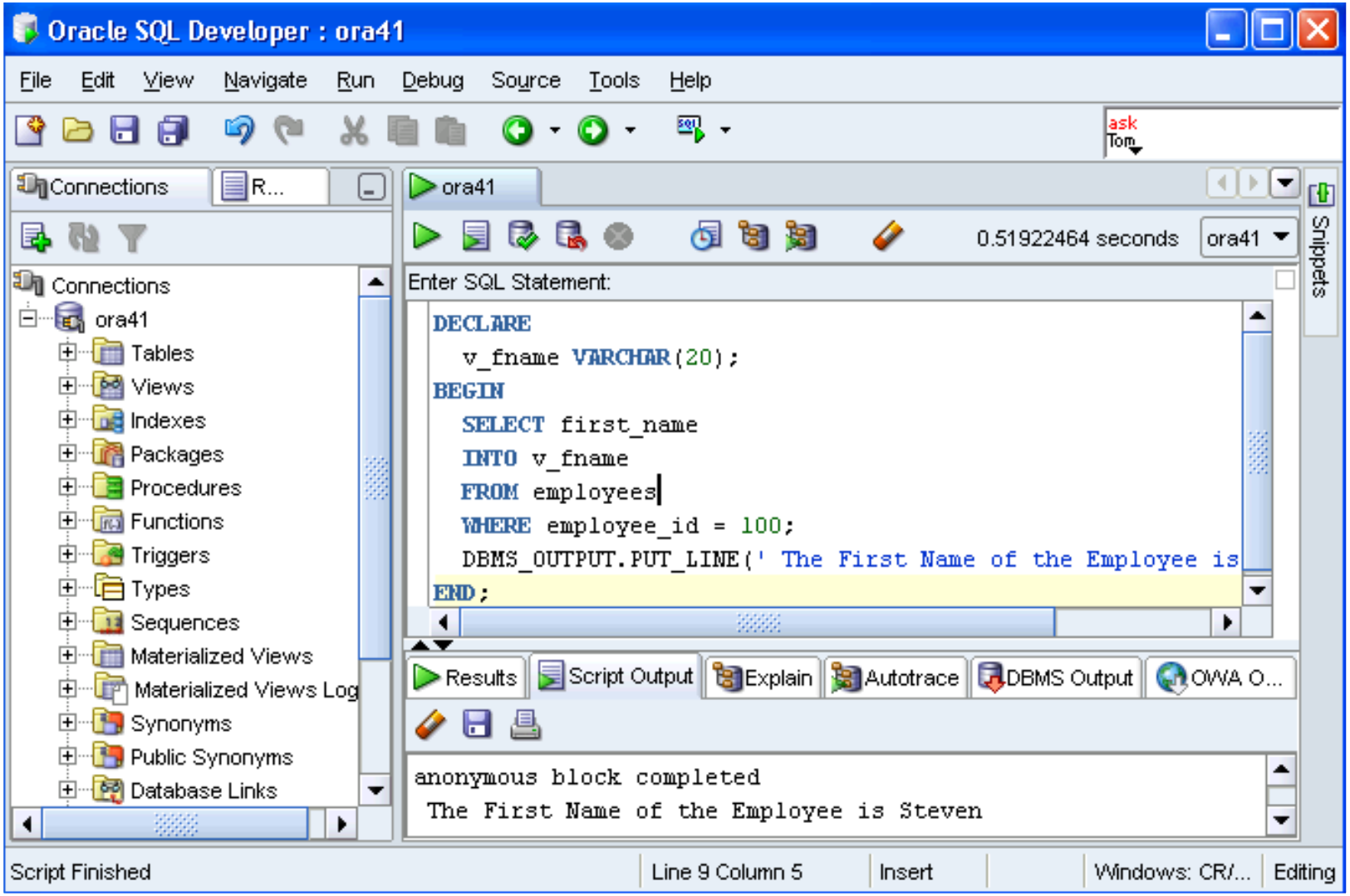


ใช้คำสั่งในการแสดงผลใน Oracle package ดังนี้

- DBMS_OUTPUT.PUT_LINE

```
DBMS_OUTPUT.PUT_LINE(' The First Name of the Employee is ' || v_fname);
...
```

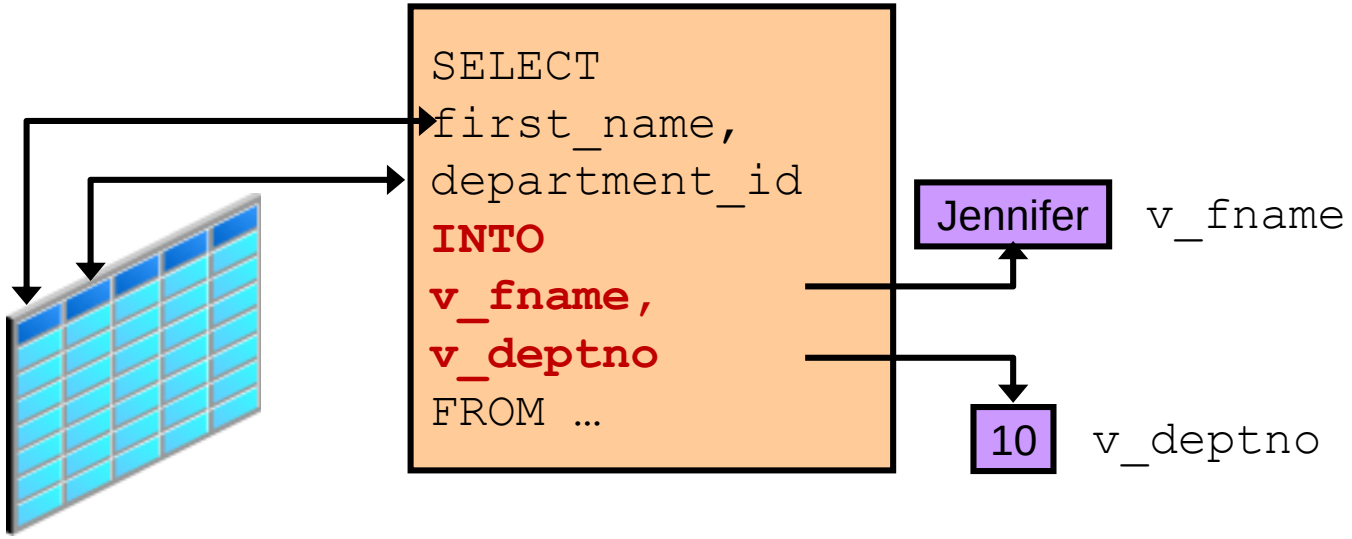
TEST THE OUTPUT OF A PL/SQL BLOCK



USE OF VARIABLES

ตัวแปร (Variables) ใช้สำหรับ

- จัดเก็บข้อมูลชั่วคราว
- จัดการหรือประมวลผลค่าข้อมูลที่จัดเก็บไว้
- นำกลับมาใช้ใหม่ได้



REQUIREMENTS FOR VARIABLE NAMES

การกำหนดชื่อตัวแปร

- ขึ้นต้นด้วยตัวอักษร
- ประกอบด้วยตัวอักษรและตัวเลข
- ประกอบด้วยอักษรพิเศษ เช่น $ _ และ #
- จะต้องมีความยาวไม่เกิน 30 ตัวอักษร
- ไม่สามารถใช้คำสงวนได้ เช่น select, insert

HANDLING VARIABLES IN PL/SQL

การจัดการตัวแปร

- ประกาศตัวแปรและกำหนดค่าเริ่มต้นในส่วนของ declarative section
- ใช้งานและกำหนดค่าให้ใหม่ในส่วนของ executable section
- ใช้สำหรับส่งพารามิเตอร์ไปยังโปรแกรมย่อย
- ใช้สำหรับส่งค่าผลลัพธ์ออกมาจากโปรแกรมย่อย

OPERATORS IN PL/SQL

- ตรรกะ (Logical)
- การคำนวณ (Arithmetic)
- การเชื่อมต่อ (Concatenation)
- การใช้วงเล็บ

  (Parentheses to control order of operations
- เลขชี้กำลัง

  (Exponential operator (**))

Same as in SQL

OPERATORS IN PL/SQL: EXAMPLES

- การเพิ่มค่าสำหรับนับรอบของ loop

```
loop_count := loop_count + 1;
```

- การกำหนดค่าตัวแปร boolean

```
good_sal := sal BETWEEN 50000 AND 150000;
```

- การตรวจสอบค่าว่าง

```
valid  := (empno IS NOT NULL);
```

DECLARING AND INITIALIZING PL/SQL VARIABLES

Syntax:

```
identifier [CONSTANT] datatype [NOT NULL]
     [:= | DEFAULT expr];
```

Examples:

```
DECLARE
  v_hiredate      DATE;
  v_deptno        NUMBER(2) NOT NULL := 10;
  v_location      VARCHAR2(13) := 'Atlanta';
  c_comm          CONSTANT NUMBER := 1400;
```

DECLARING AND INITIALIZING PL/SQL VARIABLES

1

```
DECLARE
   v_myName VARCHAR2(20);
BEGIN
 DBMS_OUTPUT.PUT_LINE('My name is: '|| v_myName);
 v_myName := 'John';
 DBMS_OUTPUT.PUT_LINE('My name is: '|| v_myName);
END;
/
```

2

```
DECLARE
 v_myName VARCHAR2(20):= 'John';
BEGIN
 v_myName := 'Steven';
 DBMS_OUTPUT.PUT_LINE('My name is: '|| v_myName);
END;
/
```

DELIMITERS IN STRING LITERALS

```
DECLARE
    v_event VARCHAR2(15);
BEGIN
   v_event  := q'!Father's day!';
  DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
   '|| v_event );
   v_event  := q'[Mother's day]';
  DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
   '|| v_event );
END;
/
```

```
anonymous block completed
3rd Sunday in June is : Father's day
2nd Sunday in May is : Mother's day
```

TYPES OF VARIABLES

TRUE

25-JAN-01

BLOB

Snow White
Long, long ago,
in a land far, far away,
there lived a princess called
Snow White. . .

256120.08

BFILE

VARCHAR2 or CLOB

Atlanta

GUIDELINES FOR DECLARING AND INITIALIZING PL/SQL VARIABLES

ข้อแนะนำในการใช้ตัวแปร

- ทำตามกฎการตั้งชื่อ
- ตั้งชื่อตัวแปรให้สื่อความหมาย
- การกำหนดค่าให้กับตัวแปร จะต้องไม่เป็นค่าว่าง และ สามารถกำหนดเป็นค่าคงที
- การกำหนดค่าเริ่มต้นโดยใช้สัญลักษณ์ assignment operator (:=) หรือ DEFAULT keyword

```
v_myName VARCHAR2(20):='John';
```

```
v_myName VARCHAR2(20) DEFAULT 'John';
```

- ประกาศตัวแปร 1 ตัวแปรต่อหนึ่งบรรทัดเพื่อให้อ่านได้ง่าย

GUIDELINES FOR DECLARING PL/SQL VARIABLES

- หลีกเลี่ยงการตั้งชื่อตัวแปรให้ตรงกับชื่อคอลัมน์

```
DECLARE
   employee_id  NUMBER(6);
BEGIN
   SELECT     employee_id
   INTO       employee_id
   FROM       employees
   WHERE      last_name = 'Kochhar';
END;
/
```

- ใช้ เงื่อนไขของ NOT NULL เมื่อมีการใช้ค่าตัวแปร

COMMENTING CODE

- กรณีที่เป็นบรรทัดเดี่ยวให้ใช้เครื่องหมาย hyphens (--)
- กรณีที่หลายบรรทัดให้ใช้ /* สำหรับเปิด และ */ สำหรับปิด

Example:

```
DECLARE
...
v_annual_sal NUMBER (9,2);
BEGIN
/* Compute the annual salary based on the
   monthly salary input from the user */
v_annual_sal := monthly_sal * 12;
--The following line displays the annual salary
DBMS_OUTPUT.PUT_LINE(v_annual_sal);
END;
/
```

SCALAR DATA TYPES

- เก็บค่าเดียว
- ไม่มีส่วนประกอบอื่น ๆ ภายใน

TRUE

25-JAN-01

The soul of the lazy man desires, and he has nothing; but the soul of the diligent shall be made rich.

256120.08

Atlanta

DECLARING SCALAR VARIABLES

Examples:

```
DECLARE
  v_emp_job           VARCHAR2(9);
  v_count_loop        BINARY_INTEGER := 0;
  v_dept_total_sal    NUMBER(9,2) := 0;
  v_orderdate         DATE := SYSDATE + 7;
  c_tax_rate          NUMBER(3,2) := 8.25;
  v_valid             BOOLEAN NOT NULL := TRUE;
  ...
```

%TYPE ATTRIBUTE

ใช้สำหรับประกาศตัวแปรต่อไปนี้

- คอลัมน์ในฐานข้อมูล
- การตัวแปรอื่น ๆ ที่ประกาศไว้

มีคำนำหน้าเป็น

- ชื่อตารางและคอลัมน์ของฐานข้อมูล

ชื่อของตัวแปรอื่น ๆ ที่ประกาศไว้

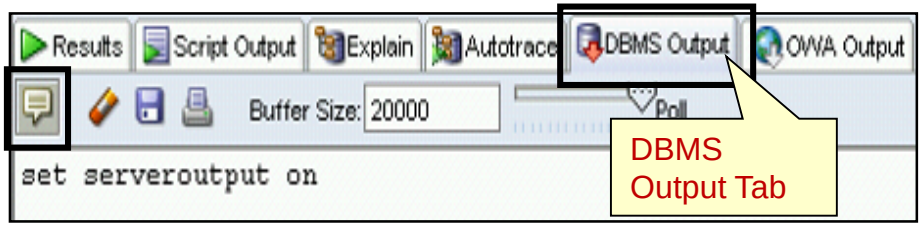
```
Declare
 varname employee.name%TYPE;
BEGIN
 select name into varname from employee where id = 1
END;
```

DECLARING BOOLEAN VARIABLES

- ตัวแปรชนิด บูลีน สามารถกำหนดค่าได้เฉพาะ TRUE, FALSE, และNULL
- เงื่อนไขสามารถใช้คำสั่ง AND,OR และ NOT
- การคำนวณทางคณิตศาสตร์ ตัวอักษร และ วันที่ สามารถใช้ในการคืนค่าเป็นค่าบูลีนได้
- **Examples**
  - emp_sal1 := 50000;
  - emp_sal2 := 60000;

Declare and initialize a Boolean variable:

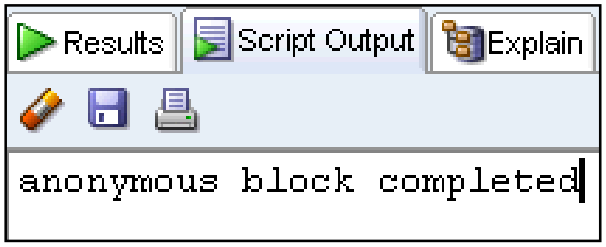
- DECLARE
  flag BOOLEAN := FALSE;
  BEGIN
  flag := TRUE;
  END;



BIND VARIABLES

การส่งผ่านตัวแปร (Bind variables)

- เรียกว่า host variables
- ใช้ VARIABLE keyword ในการประกาศตัวแปร
- ใช้ในคำสั่ง SQL และ PL/SQL blocks
- สามารถเรียกใช้ได้ก็ต่อเมื่อ PL/SQL block ถูกประมวลผล
- อ้างอิงโดยใช้เครื่องหมาย colon (:)



PRINTING BIND VARIABLES

Example:

```
VARIABLE b_emp_salary NUMBER
BEGIN
   SELECT salary  INTO :b_emp_salary
   FROM  employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name FROM employees
WHERE salary=:b_emp_salary;
```

PRINTING BIND VARIABLES

Example:

```
VARIABLE b_emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
  v_empno NUMBER(6):=&empno;
BEGIN
   SELECT salary  INTO :b_emp_salary
   FROM  employees WHERE employee_id = v_empno;
END;
```

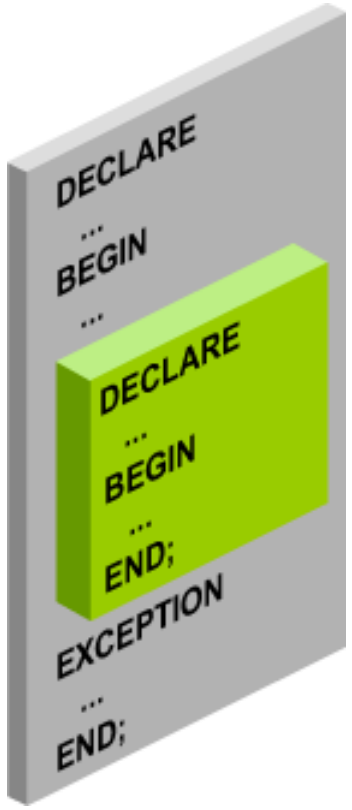
Output:

```
7000
```

NESTED BLOCKS

สามารถเขียนชุดคำสั่ง PL/SQL ซ้อนกันได้

- อยู่ภายใต้คำสั่ง (BEGIN ... END) ของชุดคำสั่งแรก



NESTED BLOCKS

Example:

```
DECLARE
 v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
  DECLARE
   v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';
  BEGIN
   DBMS_OUTPUT.PUT_LINE(v_inner_variable);
   DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
 DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

VARIABLE SCOPE AND VISIBILITY

```
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
  END;
 DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
/
```

1

2

QUALIFY AN IDENTIFIER

```
BEGIN <<outer>>
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '
                        ||outer.v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
  END;
END;
END outer;
```

SQL STATEMENTS IN PL/SQL

- ใช้สำหรับการอ่านค่าจากฐานข้อมูลโดยใช้คำสั่ง SELECT
- เปลี่ยนแปลงข้อมูลในฐานข้อมูลโดยใช้ชุดคำสั่ง DML
- ควบคุมการทำงานของทรานแซคชัน โดยใช้คำสั่ง COMMIT, ROLLBACK หรือ SAVEPOINT

SELECT STATEMENTS IN PL/SQL

รูปแบบการอ่านข้อมูลจากฐานข้อมูลโดยใช้คำสั่ง SELECT

Syntax:

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

SELECT STATEMENTS IN PL/SQL

- ใช้คำสั่ง INTO ในการอ่านค่าใส่ตัวแปร
- คิวรีข้อมูลจะต้องคืนค่าแถวเดียว

Example:

```
DECLARE
 v_fname VARCHAR2(25);
BEGIN
 SELECT first_name INTO v_fname
 FROM employees WHERE employee_id=200;
 DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
END;
/
```

RETRIEVING DATA IN PL/SQL

อ่านค่าคอลัมน์ hire_date และ salary สำหรับข้อมูลตาราง employee

Example:

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

RETRIEVING DATA IN PL/SQL

คืนค่าผลรวมของเงินเดือนสำหรับพนักงานทุกคนในแผนกที่ 60

Example:

```
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 of salary is ' ||
                        v_sum_sal);
END;
```

NAMING CONVENTIONS

```
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;
/
```

```
Error report:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
01422. 00000 -  "exact fetch returns more than requested number of rows"
*Cause:    The number specified in exact fetch is less than the rows returned.
*Action:   Rewrite the query or change number of rows requested
```

NAMING CONVENTIONS

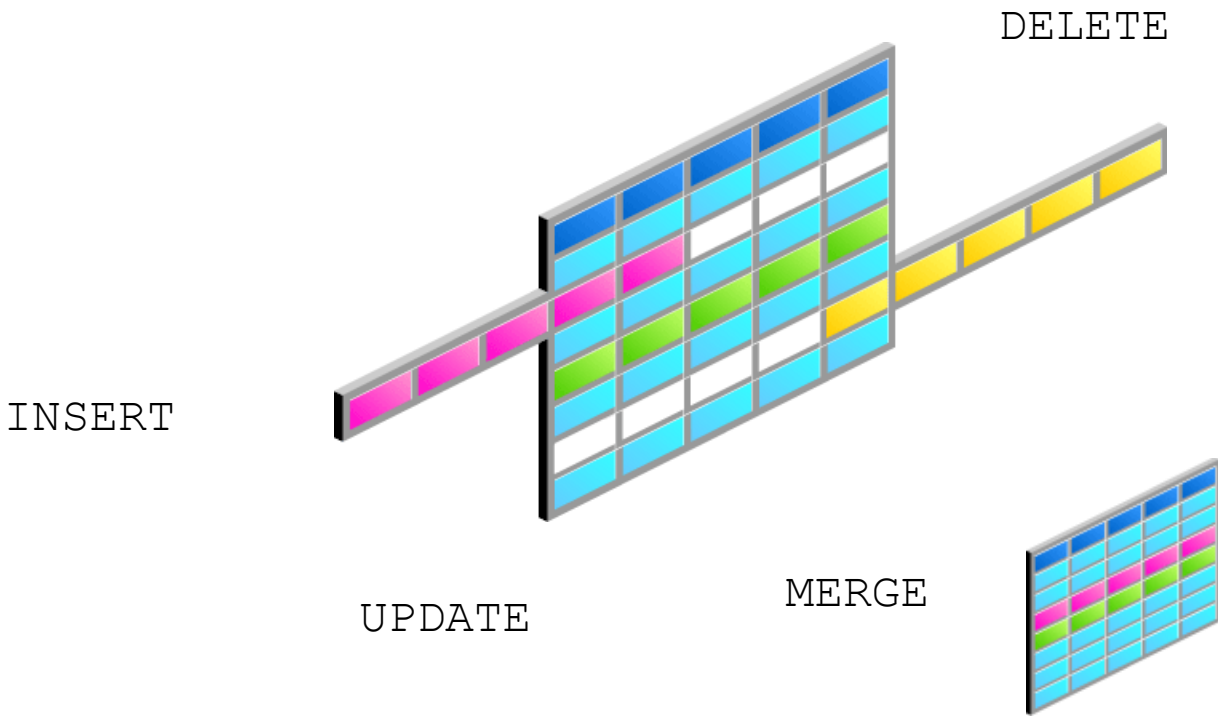
การตั้งชื่อตัวแปรในส่วนของคำสั่ง INTO

- ไม่ใช้ชื่อซ้ำกับคำสงวน
- ไม่ใช้ชื่อคอลัมน์ในฐานข้อมูล
- สามารถแสดงข้อผิดพลาดที่เกิดขึ้นได้ในคำสั่ง PL/SQL
- ชื่อของตัวแปร และ พารามิเตอร์ มีลำดับความสำคัญเหนือกว่าชื่อของตารางฐานข้อมูล
- ชื่อของคอลัมน์ในฐานข้อมูล มีลำดับความสำคัญเหนือกว่าชื่อของตัวแปร

USING PL/SQL TO MANIPULATE DATA

การเปลี่ยนแปลงข้อมูลของตารางในฐานข้อมูลโดยใช้คำสั่ง DML

- INSERT
- UPDATE
- DELETE



INSERTING DATA

เพิ่มข้อมูลลงฐานข้อมูลในตาราง EMPLOYEES

Example:

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

UPDATING DATA

แก้ไขข้อมูลเงินเดือน โดยเพิ่มตามจำนวนที่กำหนด

Example:

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

DELETING DATA

ลบข้อมูลที่มีรหัสแผนกเป็น 10 จากตาราง employees

Example:

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

SQL FUNCTIONS IN PL/SQL

- ฟังก์ชันที่สามารถใช้ได้ในส่วนของคำสั่ง PL/SQL
  - Single-row functions
    - Date: Add_Month(hiredate,3)
    - Number: Trunc(10.01,0), Round(19.27,1) //19.3
    - Character: Lower (name), Length(name)
- ฟังก์ชันที่ไม่สามารถใช้ได้ในส่วนของคำสั่ง PL/SQL
  - DECODE
  - Group functions

    (available only in SQL statements in a PL/SQL block)

SQL FUNCTIONS IN PL/SQL: EXAMPLES

- ตัวอย่างการใช้ฟังก์ชัน หาความยาวของข้อความ

```
v_desc_size INTEGER(5);
v_prod_description VARCHAR2(70):='You can use this
product with your radios for higher frequency';

-- get the length of the string in prod description
v_desc_size:= LENGTH(v_prod_description);
```

- ตัวอย่างการใช้ฟังก์ชัน จำนวนเดือนที่พนักงานทำงานมาแล้ว

```
v_tenure INTEGER(5);

v_tenure := MONTHS_BETWEEN (CURRENT_DATE, v_hiredate);
```

SQL CURSOR

- เคอร์เซอร์ คือตัวชี้ข้อมูลในส่วนของหน่วยความจำที่จัดสรรโดย Oracle server
- ถูกใช้ในการจัดการผลลัพธ์ที่ได้จากคำสั่ง SELECT
- ประกอบด้วย 2 ประเภทของเคอร์เซอร์
  - Implicit : สร้างและจัดการภายในโดย Oracle server เพื่อประมวลผลคำสั่ง SQL
  - Explicit : ถูกสร้างและจัดการโดยนักพัฒนาระบบ

SQL CURSOR ATTRIBUTES FOR IMPLICIT CURSORS

การใช้แอททริบิวส์ของ SQL cursor สามารถทดสอบผลลัพธ์ของคำสั่ง SQL โดยคำสั่งต่อไปนี้

SQL%FOUND	คืนค่า boolean หากผลลัพธ์ที่ได้ มี ข้อมูลจะคืนค่า True
SQL%NOTFOUND	คืนค่า boolean หากผลลัพธ์ที่ได้ ไม่มี ข้อมูลจะคืนค่า True
SQL%ROWCOUNT	คืนค่าจำนวนแถวของผลลัพธ์ที่ได้

SQL CURSOR ATTRIBUTES FOR IMPLICIT CURSORS

ตัวอย่างการลบข้อมูลที่กำหนดรหัสพนักงาน และแสดงผลจำนวนแถวที่ลบได้

Example:

```
DECLARE
  v_rows_deleted VARCHAR2(30)
  v_empno employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM  employees
  WHERE employee_id = v_empno;
  v_rows_deleted := (SQL%ROWCOUNT || ' row deleted.');
  DBMS_OUTPUT.PUT_LINE (v_rows_deleted);
END;
```