โครงการพัฒนาศูนย์ข้อมูลกลางกระทรวงมหาดไทยและจังหวัด

พร้อมติดตั้ง ระยะที่ 2

สัญญาเลขที่ ศสส.9/2548 ลว. 10 มีนาคม 2548

เอกสารประกอบการอบรม

การใช้งาน PL/SQL

บริษัท โปรเฟสชั่นนัล คอมพิวเตอร์ จำกัด

475 อาคารสิริภิญโญ ชั้น 17-18 ถนนศรีอยุธยา แขวงถนนพญาไท

เขตราชเทวี กทม. 10400 โทร 0-2201-3600, 0-2201-3636

Introduction to PL/SQL

PL ย่อมาจาก Procedural Language เป็นภาษาที่ Oracle พัฒนาขึ้น เพื่อให้ผู้ใช้สามารถพัฒนาโปรแกรม ในลักษณะ procedure ได้ โดยในขณะเดียวกันยังคงสามารถใช้คำสั่ง SQL ได้เช่นเดิม ลักษณะคำสั่งภาษา SQL จะเป็นการสั่งทีละคำสั่งเดียว แล้วให้ผลลัพธ์ทันที เช่น

```
SELECT * FROM emp;
```

หรือ

```
UPDATE emp SET salary = salary * 1.1;
```

ส่วนลักษณะคำสั่งภาษา PL/SQL จะเป็นการทำงานทีละ procedure เช่น

```
DECLARE V_deptno NUMBER;
BEGIN
    SELECT deptno
    INTO v_deptno
    FROM dept
    WHERE deptname = 'Accounting';
    UPDATE emp
    SET deptno = V_deptno
    WHERE empno = 10;
END;
```

ข้อดีของภาษา PL/SQL

1. Control flow การทำงานในโปรแกรมได้ด้วยคำสั่งต่าง ๆ เช่น IF statement, Loop ต่าง ๆ
2. การเข้าถึงข้อมูล สามารถทำได้ง่ายด้วยคำสั่ง SQL ธรรมดา
3. Portability คือ เขียนโปรแกรมครั้งเดียว สามารถ port ข้าม platform ได้ถ้าต้องการย้ายเครื่อง ไม่จำเป็นต้องเขียนใหม่ สามารถเอา source code เก่ามาใช้ได้เลย
4. Tools ต่างๆ ของ oracle ใช้ภาษา PL/SQL ในการเขียนโปรแกรม ทำให้ผู้พัฒนาไม่ต้องเรียนรู้ หลายภาษา เพียงแค่เรียน PL/SQL อย่างเดียว ก็สามารถพัฒนา applications ด้วย oracle tools ได้เลย (แต่ต้อง เรียนรู้ features ของ tools นั้น ๆ เพิ่มเติม)
5. ใช้ตัวแปรได้
6. Handle exception ได้ (exception = error ที่เกิดระหว่างการทำงานในโปรแกรม) เช่น การหารด้วย 0

โครงสร้างโปรแกรมภาษา PL/SQL

การเขียนโปรแกรมภาษา PL/SQL เราจะเขียนเป็น block ซึ่งแต่ละ block มีโครงสร้างดังนี้

```
[ DECLARE
การประกาศตัวแปร ]
BEGIN
        [ EXCEPTION
        Exception Handling
]
        END;
```

1. Declaration Section เริ่มด้วย DECLARE
2. Executable Code เริ่มด้วย BEGIN จบด้วย END;
3. Exception Handler เริ่มด้วย EXCEPTION และจะถูกสร้างไว้ก่อน END;

หลักการเขียน PL/SQL Blocks

1. การประกาศตัวแปร และการ Handle Exception เป็น Optional ถ้าไม่ใช้ ไม่ต้องมีก็ได้
2. แต่ละคำสั่งจะปิดด้วย ; เสมอ
3. สามารถเขียน PL/SQL Block ซ้อนกันได้

```
DECLARE
BEGIN
        DECLARE BEGIN
        EXCEPTION       <= Sub Block
        END
EXCEPTION
END
```

4. ตัวแปรที่ประกาศภายใน Block จะใช้งานได้เฉพาะใน Block นั้นเท่านั้น ถ้าออกนอก Block แล้ว จะไม่รู้จัก
5. การ Comment ทำได้ 2 วิธี คือ
    5.1 ใช้ -- นำหน้าข้อความที่ต้องการ comment เป็นการ comment ตั้งแต่จุดนั้นจนจบบรรทัดนั้น
    5.2 ใช้ /* เปิด และ */ ปิดข้อความที่ต้องการ comment (สามารถใช้ comment ได้หลายบรรทัด)
6. การ Assign ค่าให้ตัวแปร ใช้เครื่องหมาย :=
7. การใช้เครื่องหมาย มีดังนี้

เปรียบเทียบค่า ได้แก่=, >, <, <=, >=, <>, !

Logical Operator ได้แก่ AND, OR, NOT

การคำนวณ ได้แก่ +, -, *, /, ** (ยกกำลัง)

Concatenation Operator สำหรับตัวอักษร ได้แก่ ||

การประกาศตัวแปรในภาษา PL/SQL

SYNTAX :

variable_name [CONSTANT] datatype [NOT NULL][{DEFAULT|:=}initial_value];

โดย :

variable_nameคือชื่อตัวแปร
CONSTANTเป็น keyword ว่าตัวแปรนี้ เป็น constant variable (ค่าคงที่) ไม่สามารถเปลี่ยนค่าได้
datatypeคือประเภทของตัวแปร เช่น

NUMBER[(p[,s])] number (p = precision, s = scale)
CHAR[(n)] fixed length character (default = 1 char)
VARCHAR2(n) variable length character n คือ maximum length
BOOLEAN logical มี 3 ค่า คือ (True, False, Null)

NOT NULLเป็นการตั้งกฎไว้ว่า ตัวแปรนี้ต้องมีค่าเสมอ (ห้ามเป็น NULL)
{DEFAULT|:=}initial_valueเป็นการกำหนดค่าเริ่มต้นให้ตัวแปร
หมายเหตุ ถ้าระบุ NOT NULL หรือ CONSTANT keyword ในการประกาศตัวแปรแสดงว่า ต้องมีการกำหนด Initial value ให้ตัวแปรนั้นด้วย

ตัวอย่างการ Declare ตัวแปร

1. V_amount NUMBER (12, 3); จะได้ตัวแปรชื่อ V_amount เก็บตัวเลขความยาวสูงสุด 12 หลัก แบ่งเป็นหน้าจุด 9 หลัก หลังจุด 3 หลัก (ค่าสูงสุด ที่เก็บได้คือ 999,999,999.999)

2. V_Vat NUMBER (5, 2) := 10; ได้ตัวแปรชื่อ V_vat เก็บตัวเลขหน้าจุดได้ 3 หลัก หลังจุด 2 หลัก และมีค่าเริ่มต้นเป็น 10 ทันที

3. V_valid BOOLEAN NOT NULL := TRUE ;

Introduction to PL/SQL

คำสั่งในภาษา PL/SQL

1. ใช้คำสั่งภาษา SQL ได้ดังนี้

Data Retrieval	SELECT
Data Manipulation Language	INSERT UPDATE DELETE
Transaction Control	COMMIT ROLLBACK SAVEPOINT

2. การ Assign ค่าตัวแปร

ชื่อตัวแปร := expression ;

3. การทำงานตามเงื่อนไขด้วย IF statement

```
IF condition THEN
    statements ;
[ELSIF condition THEN
    statements; ]
[ELSE
    statements;]
END IF;
```

โดย : condition คือตัวแปร Boolean หรือ expression ที่ได้ผลลัพธ์เป็นค่า Boolean

หมายเหตุ

* ELSIF เขียนติดกัน และ 1 IF จะมีกี่ ELSIF ก็ได้
* END IF เขียนเป็น 2 คำ
* IF statement มีได้อย่างมาก 1 ELSE เท่านั้น

ตัวอย่าง

```
IF V_date_shipped - V_date_ordered < 5 THEN
        V_ship_flag := ‘Acceptable’;
        ELSE V_ship_flag := ‘Unacceptable’;
END IF;
IF V_last_name = ‘Dumas’ THEN
        V_job := ‘Sales Representative’;
        V_region_id := 35;
END IF;
```

4. การทำงานแบบ loop เป็นการทำงานคำสั่งชุดเดิม หลาย ๆ รอบ loop ใน PL/SQL มี 3 แบบ

4.1 Basic loop

```
LOOP
        statement 1;
        statement 2;
        ...
        EXIT [WHEN condition];
END LOOP;
```

loop ประเภทนี้เป็นการวน loop ไปเรื่อย ๆ ไม่มีกำหนด (คือทำตั้งแต่ loop จนถึง END LOOP เสร็จแล้ววนกลับขึ้นไปทำใหม่ตั้งแต่ LOOP ไปเรื่อยๆ) จึงต้องมีการเช็คเงื่อนไขในการหยุดวน LOOP

วิธีเช็คเงื่อนไข ทำได้ 2 แบบ

แบบที่ 1

```
IF condition THEN
        EXIT;
END IF;
```

แบบที่ 2

```
EXIT WHEN condition;
```

หรืออาจสั่ง EXIT; โดยไม่มีเงื่อนไขเลยก็ได้

ตัวอย่าง

```
...
V_ord_id NUMBER := 100;
V_counter NUMBER (2) := 1;
BEGIN
LOOP
INSERT INTO ord_lines (ord_id, item_id)
VALUES (V_ord_id, V_counter);
V_counter := V_counter +1;
EXIT WHEN V_counter > 10;
END;
```

4.2 FOR loop

เป็นการวน loop ที่ทราบจำนวนครั้งในการทำงานที่แน่นอน

```
FOR index IN [REVERSE]
lower_bound..upper_bound
LOOP
        statement 1;
        statement 2;
END LOOP;
```

[REVERSE] ใช้สำหรับวน loop แบบย้อนหลัง (จาก upper_bound ลดลงทีละ 1 จนถึง lower_bound)

หมายเหตุ ไม่ต้องประกาศตัวแปร index ให้ตั้งชื่อได้เลย และโปรแกรมจะรู้จักตัวแปรที่เป็น index ภายใน loop เท่านั้น

ตัวอย่าง

```
...
V_ord_id NUMBER := 100;
BEGIN
        FOR i IN 1..10
        LOOP
        INSERT INTO ord_lines (ord_id, item_id)
        VALUES (V_ord-id, i);
        END LOOP;
...
```

สังเกตได้ว่า ตัวแปร i ซึ่งทำหน้าที่เป็น index ในการวน loop เราไม่ต้องประกาศ

4.3 WHILE loop

เป็นการวน loop ตามเงื่อนไข โปรแกรมจะทำการเช็คเงื่อนไขก่อน ตราบใดที่เงื่อนไขได้ผลลัพธ์เป็น TRUE จะทำงานตาม loop ดังนั้น loop ชนิดนี้อาจ ไม่ถูกทำงานเลยก็ได้ ถ้าการเช็คเงื่อนไขในครั้งแรกเป็น FALSE

```
WHILE condition
LOOP
        statement 1;
        statement 2;
END LOOP;
```

ตัวอย่าง

```
...
V_ord_id NUMBER := 100;
V_counter NUMBER(2) := 1;
BEGIN
        ...
        WHILE V_counter <= 10
        LOOP
                INSERT INTO ord_lines (ord_id, item_id)
                VALUES (V_ord_id, V_counter);
                V_counter := V_counter + 1;
        END LOOP;
END;
```

Introduction to PL/SQL

การนำ PL/SQL ไปใช้งาน

1. เขียนในรูปแบบของ Anonymous Block

```
[DECLARE variable_declaration ;]
BEGIN
        executable - code ;
END;
```

2. ใช้พัฒนาเป็น SubProgram สำหรับเรียกใช้ได้มี 3 ลักษณะ

2.1 PROCEDURE เป็นโปรแกรมย่อยที่ทำงานอะไรบางอย่าง โดยสามารถรับ parameter มาทำงานได้

2.2 FUNCTION เป็นโปรแกรมย่อยที่นิยมใช้ เพื่อหาค่าอะไรบางอย่าง แล้วคืน กลับมาเป็นชื่อของตัว function เอง

2.3 PACKAGE เป็นการรวบรวม PROCEDURE หรือ FUNCTION หลาย ๆ ตัว ถ้าไว้ด้วยกัน เพื่อง่ายต่อการควบคุมในแง่ privilege และเป็นหมวดหมู่ดีขึ้น

ในที่นี้ จะพูดถึงเฉพาะ PROCEDURE และ FUNCTION เท่านั้น

โครงสร้างการเขียน PROCEDURE และ FUNCTION

```
PROCEDURE name [ ( parameter,….)]
IS
        pl/sql block;
FUNCTION name [( parameter…)]
RETURN datatype
IS
        pl/sql block;
```

หมายเหตุ PL/SQL block ให้เริ่มด้วยการประกาศตัวแปร (ถ้ามี) โดยไม่ต้องมี keyword DECLARE หรือถ้าไม่ใช้ตัวแปรสามารถเริ่มด้วย BEGIN ได้เลย

วิธีการประกาศ parameters ใน Subprograms

parameter.name [IN | OUT | IN OUT]

datatype [{ := | DEFAULT} expr];

parameter.nameชื่อ parameter

Mode IN = รับค่าเข้า
OUT = ส่งค่ากลับ
IN OUT = รับค่าเข้าและส่งค่ากลับ

datatype ชนิดของข้อมูล ไม่ต้องระบุความยาว

[{ := | DEFAULT} expr]ใช้ initial ค่า กรณีเป็น parameter ใน mode IN และเวลาเรียกไม่ระบุค่า parameter เข้ามา

ตัวอย่าง

```
PROCEDURE change_salary
(p_emp_id IN NUMBER,
p_new_salary IN NUMBER)
IS
        /*variables declaration
        (don't include DECLARE)*/
        BEGIN
        UPDATE emp
        SET salary = p_new_salary
        WHERE id = p_emp_id;
        COMMIT;
        END;
FUNCTION tax
(p_value IN NUMBER)
RETURN NUMBER
IS
BEGIN
        RETURN (p_value * .1);
END;
```

เปรียบเทียบ Anonymous Block, Procedure และ Function

Anonymous Block	Procedure	Function

[DECLARE
การประกาศตัวแปร]
BEGIN
คำสั่งต่าง ๆ
END;
	PROCEDURE ชื่อ
	(parameter(s))
	IS
	การประกาศตัวแปร
	BEGIN
	คำสั่งต่าง ๆ
	END;
		FUNCTION ชื่อ
		(parameter(s))
		IS
		การประกาศตัวแปร
		BEGIN
		คำสั่งต่าง ๆ
		END;

วิธีการประกาศ parameters ใน Subprograms

parameter.name [IN | OUT | IN OUT]

datatype [{ := | DEFAULT} expr];

parameter.nameชื่อ parameter

Mode IN = รับค่าเข้า
OUT = ส่งค่ากลับ
IN OUT = รับค่าเข้าและส่งค่ากลับ

datatype ชนิดของข้อมูล ไม่ต้องระบุความยาว

[{ := | DEFAULT} expr]ใช้ initial ค่า กรณีเป็น parameter ใน mode IN และเวลาเรียกไม่ระบุค่า parameter เข้ามา

ตัวอย่าง

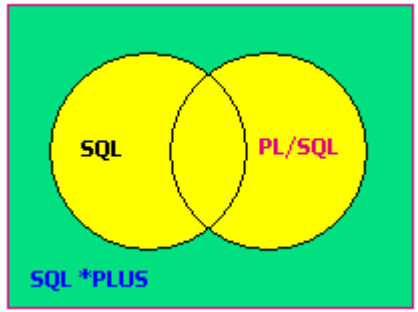
```
PROCEDURE change_salary
(p_emp_id IN NUMBER,
p_new_salary IN NUMBER)
IS
        /*variables declaration
        (don't include DECLARE)*/
        BEGIN
        UPDATE emp
        SET salary = p_new_salary
        WHERE id = p_emp_id;
        COMMIT;
        END;
FUNCTION tax
(p_value IN NUMBER)
RETURN NUMBER
IS
BEGIN
        RETURN (p_value * .1);
END;
```

Introduction to PL/SQL

SQL *Plus กับ PL/SQL

SQL *Plus เป็น tool ตัวหนึ่งของ Oracle มีลักษณะเป็น command line การใช้งานจะต้อง connect กับ Oracle Database ก่อน

Commands in SQL *Plus



SQL*Plus รองรับคำสั่ง 3 ประเภท ได้แก่

1. คำสั่งภาษา SQL
2. คำสั่งที่ทำงานตาม PL/SQL Block
3. คำสั่งของ SQL *Plus เอง ซึ่งจะเกี่ยวกับเรื่องของ environment ต่างๆ เช่น กำหนดขนาดจำนวนตัวอักษรต่อ 1 บรรทัด

โดยปกติเราจะใช้ SQL *Plus ในช่วงการ develop เพื่อสร้าง object ต่างๆ หรือเพื่อตรวจเช็คข้อมูล

ตัวอย่างคำสั่งของ SQL *Plus

- Set pagesize 25 คือ กำหนดให้ 1 หน้า screen มี 25 บรรทัด
- Set linesize 100 คือ กำหนดให้ 1 บรรทัดมี 100 chars
- Spool a.txt คือ เก็บสิ่งที่ display บนหน้าจอ ลง file ชื่อ a.txt ตั้งแต่จุดนี้ไปจนกระทั่งปิด spool
- Spool off คือ ปิด spool

เราสามารถใช้ SQL*Plus เป็น tools ในการเขียน Stored Procedure โดยภาษา PL/SQL ได้เช่นกัน