PL/SQL VS. TRANSACT SQL 筆記

筆著針對 PL/SQL 與 TRANSACT SQL(MS SQL)的比較，是以一個 PL/SQL 初學者的觀點比較，也許有需多真正的涵義尚不明瞭，尚請各位先進包涵與指正。其用意乃作為有一方之基礎，又想學習另一方之參考。

1. 什麼是 PL/SQL 語言

PL/SQL 是 ORACLE 延伸 SQL-92 後使用於 ORACLE DATABASE 的 SQL。增加了流程控制的語法。

2. PL/SQL 與 TRANSACT SQL 比較

區塊表示 BEGIN … END;/

ORACLE	MS SQL

● 必須置於 BEGIN ... END;/ 區塊間
● 每一指令必須以分號；結尾
SET SERVEROUTPUT ON
DECLARE
AA VARCHAR2(100);
BEGIN
AA := 'LKK';
DBMS_OUTPUT.PUT_LINE(AA);
END;
/
	無須區分(無須特別表示)
	DECLARE
	@AA AS VARCHAR(100)
	SET @AA = 'LKK'
	SELECT @AA

DECLARE

ORACLE	MS SQL
DECLARE mNo NUMBER; mName VARCHAR2(100);	必須以以關鍵字**@**開頭 DECLARE **@AA** AS VARCHAR(100) ,@BB INT

OPERATOR

ORACLE	MS SQL

資串結合使用關鍵字**‖**
DECLARE
mName VARCHAR2(100);
begin
DBMS_OUTPUT.PUT_LINE('值 ' ‖ mName);
end;
/
	資串結合使用關鍵字**+**
	DECLARE @STR VARCHAR(10)
	SET @STR = 'MS SQL'
	SELECT 'VALUE=' + @STR

● 指定變數值使用**:=**
DECLARE
AA VARCHAR2(100);
BEGIN
AA **:=** 'LKK';
END;
/
	● 必須以關鍵字 **SET**
	DECLARE **@AA** AS VARCHAR(100)
	SET @AA = 'LKK'
	SELECT @AA

CONDITIONAL CONTROL

IF-THEN

ORACLE	MS SQL

IF condition **THEN**
sequence_of_statements;
END IF;
 	
IF condition **THEN**
sequence_of_statements1;
ELSE
sequence_of_statements2;
END IF;
	IF *Boolean_expression*
	{ *sql_statement* | *statement_block* }
	END
	
	IF *Boolean_expression*
	{ *sql_statement* | *statement_block* }
	ELSE
	{ *sql_statement* | *statement_block* }
	END

IF x > y **THEN**
high := x;
END IF;
 	
IF trans_type = ’ CR’ **THEN**
UPDATE accounts SET balance = balance + credit
WHERE ...
ELSE
IF new_balance >= minimum_balance **THEN**
UPDATE accounts SET balance = balance - debit
WHERE ...
ELSE
RAISE insufficient_funds;
END IF;
END IF;
	IF @y>1
	PRINT @X
	
	IF (@y>1) AND (@Y>@X)
	BEGIN
	PRINT @X
	PRINT @Y
	END
	ELSE
	PRINT @X+@Y

IF-THEN-ELSIF

ORACLE	MS SQL

IF condition1 **THEN**
sequence_of_statements1;
ELSIF condition2 **THEN**
sequence_of_statements2;
ELSE
sequence_of_statements3;
END IF;
	● 無，使用 IF-ELSE 延伸

LOOP

ORACLE	MS SQL

LOOP sequence_of_statements; **END LOOP;** 必須配合 EXIT、EXIT WHEN…等離開回圈	無

WHILE-LOOP

ORACLE	MS SQL
WHILE condition **LOOP** sequence_of_statements; **END LOOP;** If the condition yields TRUE, the sequence of statements is executed. If the condition yields FALSE or NULL, next statement.	**WHILE** *Boolean_expression* { *sql_statement* | *statement_block* } [BREAK] { *sql_statement* | *statement_block* } [CONTINUE]
WHILE myTable_CURS%FOUND **LOOP** DBMS_OUTPUT.PUT_LINE('No = '|| TO_CHAR(myTable_REC.MYNO)); FETCH myTable_CURS INTO myTable_REC; **END LOOP;**	**WHILE** (SELECT AVG(price) FROM titles) < $30 **BEGIN** UPDATE titles SET price = price * 2 SELECT MAX(price) FROM titles **END**

FOR-LOOP

ORACLE	MS SQL
FOR counter **IN** [**REVERSE**] lower_bound..higher_bound **LOOP** sequence_of_statements; **END LOOP;** **FOR** i IN 1..3 **LOOP** -- assign the values 1,2,3 to i sequence_of_statements; -- executes three times **END LOOP;**	無

GOTO

ORACLE	MS SQL

GOTO <<LABEL>>
 	
BEGIN
...
<<update_row>>
BEGIN
...
END;
...
GOTO *update_row;*
	GOTO label
	
	DECLARE @tablename sysname
	SET @tablename = N'authors'
	table_loop:
	IF (@@FETCH_STATUS <> -2)
	BEGIN
	...
	END
	...

. . . END;	IF (@@FETCH_STATUS <> -1) GOTO *table_loop*

CONTINUE、BREAK、EXIT

ORACLE	MS SQL
● EXIT The EXIT statement forces a loop to complete Unconditionally. When an EXIT statement is encountered, the loop completes immediately and control passes to the next statement. ● EXIT-WHEN The EXIT-WHEN statement allows a loop to complete conditionally. When the EXIT statement is encountered, the condition in the WHEN clause is evaluated. If the condition evaluates to TRUE, the loop completes and control passes to the next statement after the loop.	● BREAK 退出最內層的 WHILE 迴圈。任何出現在標示迴圈結束之 END 關鍵字後的陳述式，會因此被執行。 ● CONTINUE 重新開始 WHILE 迴圈，忽略掉在 CONTINUE 後的任何陳述式。

RETURN

ORACLE	MS SQL
無	RETURN [integer_expression] ● 指定傳回的整數值。預存程序可傳回整數值給呼叫程序或應用程式。 ● 無條件退出查詢或程序。RETURN 是立即而完整的，而且可在任何時刻用於退出程序、批次，或陳述式封鎖。其中附隨有 RETURN 的陳述式不會被執行。

PROCEDURE

ORACLE	MS SQL
CREATE [OR REPLACE] PROCEDURE [schema.]procedure [(argument [IN | OUT | IN OUT] datatype [, argument [IN | OUT | IN OUT] datatype] ...)] {IS | AS} pl/sql_subprogram_body	CREATE PROC [EDURE] procedure_name [; number] [{ @parameter data_type } [VARYING] [= default] [OUTPUT]] [,...n] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION }] [FOR REPLICATION] AS sql_statement [...n]

● PROCEDURE 內容必須以 BEGIN…END;/框範
● 參數必須以小掛號區別
 	
CREATE OR REPLACE PROCEDURE
INTO_MYTABLE(MNO IN NUMBER,MNAME IN VARCHAR2)
AS
BEGIN
INSERT INTO MYTABLE(MYNO,MYNAME) VALUES (MNO,MNAME);
END;
/
 	
註：ORACLE 中的 USER_OBJECTS 紀錄所有物件資訊，類式 MS SQL 的 SYSOBJECTS
	● 由 SYSOBJECTS 判斷 PROCDURE 是否存在
	● 參數不需小掛號
	
	IF EXISTS(SELECT * FROM SYSOBJECTS
	WHERE name = 'MY_PROC' AND type = 'P')
	DROP PROC MY_PROC
	GO
	CREATE PROCEDURE
	MY_PROC @MNO INT,@MNAME VARCHAR(20)
	AS
	SELECT CONVERT(CHAR(1),@MNO)+' - '+@MNAME
	
	EXEC MY_PROC 2,'PROC TEST'

FUNCTION

ORACLE	MS SQL
CREATE [OR REPLACE] FUNCTION [schema.]function [(argument [IN] datatype [, argument [IN] datatype] ...)] **RETURN** datatype {IS | AS} pl/sql_subprogram_body **RETURE** VALUE	CREATE FUNCTION [owner_name.] function_name ([{ @parameter_name scalar_parameter_data_type [= default] } [,...n]]) **RETURNS** scalar_return_data_type [WITH < function_option> [[,] ...n]] [AS] BEGIN function_body **RETURN** scalar_expression END

參數指選告型態，不指定大小
 	
CREATE OR REPLACE FUNCTION MY_FUNC
(MM VARCHAR2)
RETURN VARCHAR2
AS
BEGIN
RETURN 'FUNC -> ' || MM;
END;
/
	呼叫時必須包含 owner name
	
	CREATE FUNCTION MY_FUNC
	(@MM CHAR(10))
	RETURNS CHAR(20)
	AS
	BEGIN
	RETURN 'FUNC -> ' + @MM
	END
	
	SELECT **dbo**.MY_FUNC('傳入')

PACKAGE(PL/SQL)

ORACLE

- PACKAGE 是 PROCEDURE、FUNCTION 的集合。
- PACKAGE 包含兩部份，一個為規格(宣告與變數)、另一為實作。

```
--建立 PACKAGE 規格
SET SERVEROUTPUT ON
CREATE OR REPLACE PACKAGE PACK1 IS
PROCEDURE MY_FIRST_RPO(A number);
VAR_1 INTEGER;
VAR_2 NUMBER;
VAR_3 VARCHAR2(30);
END PACK1;
/

--建立 PACKAGE 主體
CREATE OR REPLACE PACKAGE BODY PACK1 IS
PROCEDURE MY_FIRST_RPO(A NUMBER) IS
BEGIN
      VAR_1:= A;
END;

FUNCTION FUN1 RETURN VARCHAR2 IS
BEGIN
      VAR_2 := '歡迎光臨';
      RETURN VAR_2 ||' '|| TO_CHAR(VAR_1);
END;

BEGIN
      DBMS_OUTPUT.PUT_LINE('FIRST PACKAGE PROGRAM');
END PACK1;
/
```

EXCEPTION(PL/SQL)

ORACLE

```
...
      EXCEPTION
            WHEN exception_name1 THEN    -- handler
                  sequence_of_statements1
            WHEN exception_name2 THEN    -- another handler
```

```
        sequence_of_statements2
    ...
    WHEN OTHERS THEN                -- optional handler
        sequence_of_statements3
END;
```

```
--SYSTEM DEFINED
SET SERVEROUTPUT ON
DECLARE
    A INTEGER;
BEGIN
    A := 'XXXYYY';
    DBMS_OUTPUT.PUT_LINE('正常運作');
EXCEPTION
    WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('例外處理');
END;
/

--USER DEFINED
SET SERVEROUTPUT ON
DECLARE
    CHECK_CODE VARCHAR2(1);
    WRONG_CODE   EXCEPTION;
BEGIN
    CHECK_CODE := '&B';
    IF CHECK_CODE NOT IN('D','H','Y') THEN
        RAISE   WRONG_CODE;
    END IF;
    DBMS_OUTPUT.PUT_LINE('  輸入正確  ');
EXCEPTION
    WHEN   WRONG_CODE   THEN
    DBMS_OUTPUT.PUT_LINE('  觸發自訂例外處理  ');
    DBMS_OUTPUT.PUT_LINE('MYERR_CODE =' || SQLCODE || ' ' || 'MYERR_DESC = ' ||   SQLERRM );
END;
/
```

%TYPE、%ROWTYPE ATTRIBUTE(PL/SQL)

ORACLE
The %TYPE attribute is particularly useful when declaring variables that refer to database columns. You can reference a table and column, or you can reference a schema, table, and column, as the following example shows:

```
    my_dname    scott.dept.dname%TYPE;

--%TYPE
SET SERVEROUTPUT ON
DECLARE
   mNo myTable.myNo%TYPE;    --myTable=table name, myNo=field name
   mName myTable.myNAME%TYPE;
BEGIN
   SELECT myNO,myNAME INTO mNO,mName
   FROM MYTABLE
   WHERE MYNO=99;
   DBMS_OUTPUT.PUT_LINE('值 ' || mName);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE(' EXCEPTION (NO FOUND)');
END;
/

--%ROWTYPE
SET SERVEROUTPUT ON
DECLARE
   mDATA myTable%ROWTYPE;
BEGIN
   SELECT myNO,myNAME INTO mDATA.myNO,mDATA.myNAME
   FROM MYTABLE
   WHERE MYNO=1;
   DBMS_OUTPUT.PUT_LINE('ROWTYPE 值 ' || mDATA.MYNAME);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE(' EXCEPTION (NO FOUND)');
END;
/
```

SELECT … INTO

ORACLE	MS SQL
指定(ASSIGN)變數值，例如： SET SERVEROUTPUT ON DECLARE mNo NUMBER; mName VARCHAR2(100);	**建立新的資料表**並使用 SELECT 的結果集來填入該資料表。新資料表的結構由選取清單中的運算式屬性所定義，例如： **SELECT** Shippers.*, Link.Address, Link.City,

begin
SELECT myNO,myNAME INTO mNO,mName
FROM MYTABLE
WHERE MYNO=1;
DBMS_OUTPUT.PUT_LINE('值 ' ‖ mName);
end;
/
 	
註：SELECT … INTO 若沒有任何資料或超出一列都會產生錯誤。
	Link.Region, Link.PostalCode
	INTO NewShippers
	FROM Shippers
	JOIN LinkServer.DB.dbo.Shippers AS Link
	ON (Shippers.ShipperID = Link.ShipperID)

CURSOR

ORACLE	MS SQL

SET SERVEROUTPUT ON
DECLARE
CURSOR myTable_CURS IS SELECT * FROM myTable;
myTable_REC myTable%ROWTYPE;
BEGIN
OPEN myTable_CURS;
FETCH myTable_CURS INTO myTable_REC;
WHILE myTable_CURS%FOUND LOOP
DBMS_OUTPUT.PUT_LINE('No = '‖
TO_CHAR(myTable_REC.MYNO) ‖
' , PHONE = '‖ myTable_REC.MYNAME);
FETCH myTable_CURS INTO myTable_REC;
END LOOP;
CLOSE myTable_CURS;
END;
/
	DECLARE
	@MYNO INT, @MYNAME CHAR(100)
	DECLARE
	myTable_CURS CURSOR FOR
	SELECT * FROM myTable
	OPEN myTable_CURS
	FETCH NEXT FROM myTable_CURS
	INTO @MYNO, @MYNAME
	WHILE @@FETCH_STATUS = 0
	BEGIN
	PRINT CONVERT(CHAR(5),@MYNO)+' ,
	'+@MYNAME
	FETCH NEXT FROM myTable_CURS
	INTO @MYNO, @MYNAME
	END
	CLOSE myTable_CURS
	DEALLOCATE myTable_CURS

註：無須 DEALLOCATE ● %NOTFOUND evaluates to TRUE if an INSERT, UPDATE, or DELETE affected no rows or a SELECT INTO returned no rows. Otherwise, %NOTFOUND evaluates ● %FOUND is the logical opposite of %NOTFOUND. After an explicit cursor is open but before the first fetch, %FOUND evaluates to NULL. Thereafter, it evaluates to TRUE if the last fetch returned a row or to FALSE if no row was returned%ROWCOUNT returns the number of rows affected by an INSERT, UPDATE, or DELETE or returned by a SELECT INTO. %ROWCOUNT returns a zero if an INSERT, UPDATE, or DELETE affected no rows or a SELECT INTO returned no rows ● %ISOPEN evaluates to TRUE if its cursor is open; otherwise, %ISOPEN evaluates to FALSE	註： ● DECLARE CURSOR 定義了 Transact-SQL 伺服器資料指標的屬性。 ● OPEN 陳述式擴展了結果集， ● FETCH …INTO 會從結果集中傳回一個資料列。 ● CLOSE 陳述式釋放了與資料指標關聯的目前結果集。 ● DEALLOCATE 陳述式則釋放資料指標所使用的資源。 ● @@FETCH_STATUS 傳回最後一個資料指標 FETCH 陳述式的狀態，此狀態由該連線目前所開啟的任何資料指標所發出。(RESULT=0 表 FETCH 陳述式順利執行；= -1 表 FETCH 陳述式失敗，或資料列超出結果集；= -2 表擷取的資料列已遺漏。

SEQUENCT

ORACLE	MS SQL
● 用來產生欄位唯一值 CREATE SEQUENCE MYTABLE_ID MINVALUE 1 MAXVALUE 99999 INCREMENT BY 1 START WITH 1 NOCACHE NOORDER NOCYCLE; select MYTABLE_ID.NEXTVAL from DUAL; //取得唯一值	● 指定欄位 IDENTITY 屬性 表示新資料行為識別資料行。當新的資料列新增至資料表時， MicrosoftR SQL Server 將為資料行提供唯一且累加的值。識別資料行通常用於結合 PRIMARY KEY 條件約束，作為資料表唯一性的資料列識別。IDENTITY 屬性可以指定為 tinyint、smallint、int、decimal(p,0) 、或 numeric(p,0) 資料行。每一個資料表只能建立一個識別資料行。識別資料行無法使用界限預設值及 DEFAULT 條件約束。您必須同時指定初始值及遞增值，或都不指定。如果不指定，預設為 (1,1)。 CREATE TABLE jobs (job_id smallint IDENTITY(1,1) PRIMARY KEY CLUSTERED,)

TRANSACTION PROCESSING

ORACLE	MS SQL

Begin when the first executable SQL command is executed.	**BEGIN TRAN** [**SACTION**] [*transaction_name* | *@tran_name_variable* [WITH MARK ['*description*']]]
COMMIT;	**COMMIT** [**TRAN** [**SACTION**] [transaction_name | **@**tran_name_variable]]
ROLLBACK [TO label_name]	**ROLLBACK** [**TRAN** [**SACTION**] [transaction_name | @tran_name_variable | savepoint_name | @savepoint_variable]]
SAVEPOINT label_name;	**SAVE TRAN** [**SACTION**] { savepoint_name | @savepoint_variable } 分散式交易中皆不支援 SAVE TRANSACTION。
INSERT INTO MYTABLE VALUES (1,'--1'); **ROLLBACK**; INSERT INTO mytable VALUES (2,'--2'); **SAVEPOINT P1;** UPDATE mytable set myname='--2 modify' where myno=2; **SAVEPOINT P2;** ROLLBACK TO p1; INSERT INTO mytable VALUES (3,'--3'); **COMMIT;**	**BEGIN TRANSACTION** UPDATE titles SET advance = advance * 1.25 WHERE ytd_sales > 8000 **COMMIT** **BEGIN TRAN** ta INSERT INTO MYTABLE VALUES(3,'3-') INSERT INTO MYTABLE VALUES(4,'4-') **SAVE TRANSACTION** ta_p INSERT INTO MYTABLE VALUES(5,'5-') **ROLLBACK TRAN** ta_p **COMMIT**

3. 備註

什麼是 SQL？

SQL 是一專門用來處理關聯式資料庫的標準程式語言。它誕生於 1970 年代後半，促使 SQL 問世的功臣是位於加州聖荷西的 IBM 實驗室(IBM Laboratory)。SQL 為 Structured Query Languoge(結構化查詢語言)的縮寫。它讀成「S-Q-L」或「see-kwul」。

SQL 的標準化作業，是由 ANSI(美國國家標準學會)與 ISO(國際標準組織)這 2 個標準化組織所推動的。它最初的標準化規格，是在 1986 年由 ANSI 所制定，並緊接著在 1992 年時 ANSI 與 ISO 分別制南的新的規格，這項規格一般為 SQL-92 規格，通稱 SQL2。在目前，ANSI 與 SIO 也持續地在研討新一代的規格，這項規格一般稱為 SQL-99 規格，通稱 SQL3。

現在的 SQL 規格便是上述的 SQL-92(SQL2)。儘管 SQL 的標準化不斷地進行著，但在另一方面，資料庫的軟體商卻也續地在擴充自已獨特的功能。事實上目前大多數的資料庫軟體商，都會在其 RDBMS 產品的 SQL 中，加上它們獨特的功能。因此各個產品中實際使用的是不具相容性的擴充後

的 SQL。包括了 Oracle 所使用的 PL/SQL、以及 SQLServer 中所使用的 Transact-SQL。而這些 SQL 中由軟體商獨自擴充的部份都是不具有相容性的。

SQL 是一種關聯式資料庫的專屬語言，換句話說它只能用來處理資料庫。SQL 被稱為「非程序語言」。程序語言(procedural language)會將多項命令彙整起來，然後按照處理的流程依序撰寫。非程序語言(nonprocedural language)則是單獨地撰寫一個命令。一個命令便擁有一個完整的意義，可用來處理一項工作。

SQL 的功能分類

- 資料定義語言(DDL)。SQL 中屬於資料定義語言的命令，共有下列這幾項：
  - CREATE：建立資料庫或資料表
  - ALTER：變更資料庫或資料表的結構
  - DROP：刪除資料庫或資料表
- 資料操作語言(DML)
  - SELECT：搜尋資料
  - INSERT：新增資料
  - UPDATE：更新資料
  - DELETE：刪除資料
- 資料控制語言(DCL)
  - GRANT：授予使用者操作資料的權限
  - REVOKE：撤回使用者操作資料的權限
  - COMMIT：確定資料的變更
  - ROLLBACK 取消資料的變更

Transact-SQL

Transact-SQL 是 Microsoft 延伸 SQL-92 後使用於 MS SQL Server 的 SQL。增加了流程控制的語法(if、while)、區域變數及其他功能，讓我們能夠編寫更複雜的資料查詢，以及建立可以放在伺服器上的程式物件(預存程序、觸發機制)。Transact-SQL 的敘述，至少要包含一個命令(一個代表動作的動詞)：select、update、insert、delete。

(go 是假指令，告訴 ISQL 把這些指令整批交付給伺服器處理)

REFERENCE

- ORACLE 8i 系統建置與管理手冊 … 知城數位科技 … 王成春、蕭雅云著
- 資料庫管理系統課程 …東南技術學院資訊管理科 … http://study.tnit.edu.tw/teacher/ccyen/db/
- PL/SQL User’ s Guide and Reference, Release 2 (9.2)
- Oracle9i SQL Reference, Release 2 (9.2)
- MICROSOFT MSDN

五大資料庫管理系統之比較總表

http://www.dbmaker.com.tw/reference/issue/tdbcomp_t.html