

1.1 PL/SQL

linguaggio per DBMS

PL/SQL (*Procedural Language/Structured Query Language*) è il linguaggio di programmazione del DBMS ORACLE. PL/SQL ha le funzionalità di un linguaggio di tipo generale, ispirato al PASCAL. Ha funzionalità definite nella libreria Package Standard che comprende librerie per l'accesso a file di testo, alla comunicazione fra processi, al trattamento dinamico dei comandi SQL. E' possibile creare librerie dell'utilizzatore.

I tipi di dato di PL/SQL rispecchiano i tipi di dato previsti dal DBMS ORACLE con la possibilità di dichiarare, da questi, propri tipi di dato. Fra le istruzioni di PL/SQL sono accettati comandi SQL.

PL/SQL è utilizzato per scrivere subroutine richiamate da eventi sul DataBase (*trigger*), funzioni e procedure.

Dalla versione 8 di Oracle 8 la programmazione ad Oggetti. Le classi sono tipi particolari chiamate *Abstract Data Type* (ADT).

I programmi PL/SQL sono divisi in blocchi composti di tre parti:

- Dichiarativa opzionale, contiene la dichiarazione delle variabili, costanti e funzioni
- Eseguitibile obbligatoria, contiene le istruzioni
- Eccezioni opzionale

La parte dichiarativa inizia con la parola chiave **DECLARE**

La parte eseguibile inizia con la parola chiave **BEGIN** e termina con la parola chiave **END**;

La gestione delle eccezioni inizia con la parola chiave **EXCEPTION** e termina con la parola chiave **END**;

Il "canovaccio" di un programma PLS/QL ha la struttura:

```
DECLARE
-- qui si dichiarano oggetti, variabili e costanti utilizzate nel programma
BEGIN
-- qui si scrive il programma PLSQL
END;
EXCEPTION
-- qui si scrive l'eventuale programma di gestione di errori ed eccezioni
END;
```

I dati appartenenti ad una singola riga del DBMS sono accessibili tramite il comando **SELECT ... INTO**, analogo al comando **SELECT** di SQL, l'opzione **INTO** indica le variabili che ricevono i valori del DataBase, ad esempio:

```
SELECT name, job, sal INTO my_name, my_job, my_sal FROM emp WHERE empno =
my_empno;
```

L'accesso a più righe di un Data Base è gestito tramite una struttura di dati detta *cursore*, che è associata ad un comando SQL **SELECT**; il *cursore* è utilizzato dalle istruzioni **OPEN nomecursore... FETCH nomecursore INTO ...**, simile nell'utilizzo alle istruzioni di lettura file, e **FOR ... LOOP**.

Qui di seguito un esempio di programma PL/SQL che utilizza un cursore per accedere a dei dati in una tabella; i dati contengono informazioni di tipo gerarchico, come si può vedere dal risultato del comando **SELECT**¹:

```
Select Level livello ,substr(concat(LPAD(' ',2*(LEVEL-1)), Nome),1,16) Nome,
idfile IDEN, idfilepadre IDENPADRE from albero
START WITH IdfilePadre = 0 CONNECT BY PRIOR Idfile = idfilepadre;
LIVELLO NOME                                IDEN      IDENPADRE
-----
-----
```

¹ Le clausole **START WITH** e **CONNECT BY** non fanno parte dello standard SQL

1	Parko	1	0
2	Krono	2	1
3	Cerro	4	2
2	Viale	3	1
3	Luino	5	3
3	Pigato	6	3
1	Crotalo	7	0
2	Tramvia	8	7

La subroutine seguente legge i dati della tabella *albero* e visualizza le sequenze gerarchiche:

```

Parko Krono Cerro
Parko Viale Luino
Parko Viale Pigato
Crotalo Tramvia
DECLARE
  livello INTEGER(2);
  num      INTEGER(2) := 0; -- dichiarazione con assegnazione
  nome     VARCHAR2(12);
  CURSOR albgen IS SELECT level, Nome FROM albero START WITH IdfilePadre = 0
                    CONNECT BY PRIOR Idfile = idfilepadre;
  TYPE TableNames IS TABLE OF Varchar2(12)
    INDEX BY BINARY_INTEGER; -- TABLE come matrici
  tabNomi TableNames;
  PROCEDURE stampa (tabNomi TableNames, livello Integer) IS
    SeqNomi varchar2(120);
  BEGIN
    FOR counter IN 1..livello LOOP
      SeqNomi := CONCAT(SeqNomi, CONCAT(' ', Tabnomi(counter)));
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(SeqNomi);
  END stampa;
BEGIN
  OPEN albgen;
  LOOP
    FETCH albgen INTO livello, Nome;
    EXIT WHEN albgen%NOTFOUND;
    IF livello > num THEN
      num := num + 1;
    ELSE
      stampa(tabnomi, num);
      num := livello;
    END IF;
    tabnomi(num) := nome;
  END LOOP;
  Stampa(tabnomi, num);
  CLOSE albgen;
END;
```

1.1.1 Varianti

Linguaggi simili sono utilizzati in PostgreSQL (PL/pgSQL) e MySQL.