PL/SQL

Limbajul PL/SQL sau Procedural Language for SQL este o extensie a limbajului SQL lansată de cei de la oracle pentru a aduce niște capabilități extinse în sistemele lor de baze de date. Printre acestea se numără funcțiile și procedurile stocate, blocurile de cod cu variabile, tratarea erorilor sau lucrul mai ușor cu declanșatorii(triggeri).

Structura unui program PL/SQL se bazează pe blocuri de cod. Acestea pot fi anonime sau pot fi integrate în cazul unei alte structuri, precum o funcție sau o procedură. Un bloc este delimitat de cuvintele cheie obligatorii BEGIN și END, dar poate conține și alte cuvinte cheie precum DECLARE care anunță începerea unei secțiuni de declarare a variabilelor sau EXCEPTION care precede secțiunea de tratarea a excepțiilor.

```
DECLARE
declararea variabilelor
BEGIN
codul propriu-zis
EXCEPT
excepții
END;
```

Instrucțiuni

- Instrucțiunile CONTINUE (WHEN exp) și NULL care trec la instrucțiunea următoare
- Instrucțiunea GOTO etichetă
- Instrucțiunea EXIT WHEN exp
- Instrucțiunea IF: IF expresie_booleană THEN instrucțiune ELSE / ELSIF instrucțiune END IF;
- Instrucțiunea CASE: CASE variabila WHEN val1 WHEN val2 END CASE;
- Instrucțiunea LOOP: LOOP secvența END LOOP; Aceasta are nevoie de isntrucțiunea EXIT WHEN expresie_booleană și are și forme pentru FOR și WHILE
- WHILE condiție LOOP instrucțiuni END LOOP;
- FOR contor IN ind1..ind2 LOOP instrucțiuni END LOOP;

Ca în orice limbaj de programare avem puse la dispoziție în PL/SQL variabile care ne pot ajuta să stocăm informații de mai multe tipuri pe durata rulării programului. Unde există diferențe față de multe limbaje de programare moderne este în declararea variabilelor, aceasta putând fi realizată doar în cadrul blocului DECLARE la începutul programului. Declararea variabilelor se face enunțând obligatoriu numele, tipul de date și opțional o valoare implicită.

```
DECLARE
v_1 VARCHAR(40) := 'nume';
v_2 NUMBER(2);
v_3 INTEGER NOT NULL;
```

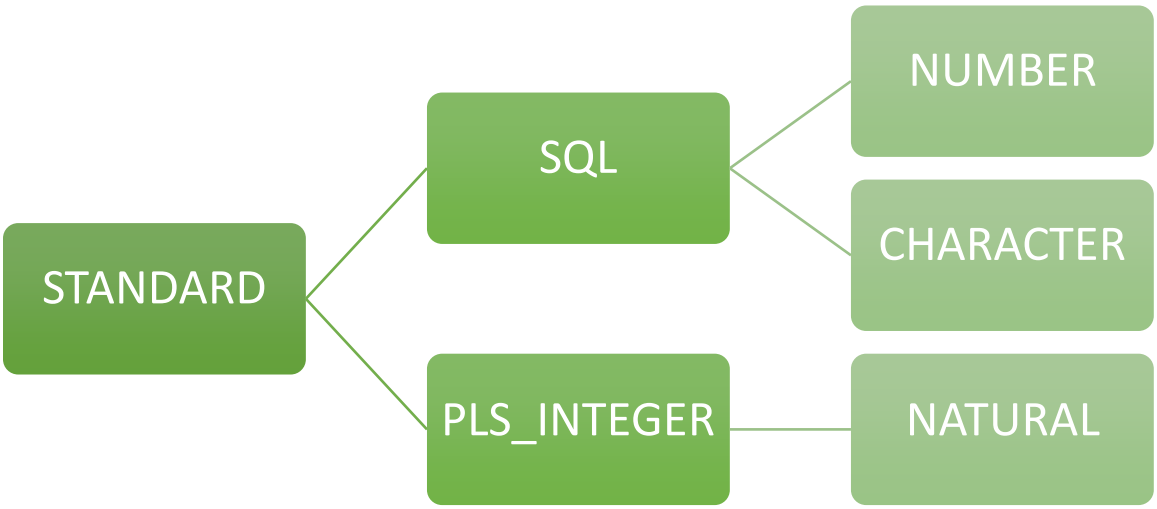
Tipurile de date din PL/SQL ne permit manipularea unei varietăți mari de câmpuri, obiecte sau entități care stochează informații sub forme diverse. Variabilele sunt de 2 feluri:

-scalare, adică rețin informații despre un singur element

-compuse, adică pot conține informații despre orice număr de variabile de un tip oarecare(acesta trebuie însă să fie constant în cadrul unui anumit tabel).

Variabile Scalare

Variabilele scalare rețin valori indivizibile intern, adică valori care nu pot fi împărțite în componente diferite. Ierarhia acestor tipuri de date este asigurată de prezența subtipurilor care implementează un mecanism de „moștenire” (o variabilă din subtip are toate proprietățile supratipului). Toate tipurile de variabile scalare sunt descendente ale supratipului STANDARD.



Tipurile de date NUMBER

Aceste tipuri de date sunt prezente și în SQL și ne permit memorarea valorilor numerice într-un mod simplu, ușor de folosit și rapid de implementat. Cele mai folosite subtipuri sunt:

- NUMBER(p,s) – stochează numere reale cu maxim p cifre în partea întreagă și s cifre în partea zecimală. Parametrul s este opțional însă pot fi folosite și valori negative sau nule pentru el ducând la rotunjirea părții întregi a numărului
- BINARY_FLOAT - stochează numere în formatul cu virgulă mobilă FLOAT
- BINARY_DOUBLE – stochează numere în formatul cu virgulă mobilă DOUBLE

Tipuri de date CHARACTER

Tipurile character permit stocarea informației sub forma șirurilor de caractere și sunt prezente și în SQL-ul obișnuit. Cele mai folosite subtipuri sunt:

- CHAR(n BYTE/CHAR) – stochează șiruri de caractere cu dimensiune fixă de n caractere sau n bytes, depinzând de ce keyword utilizăm în declarare
- VARCHAR2(n BYTE/CHAR) – stochează șiruri de caractere cu dimensiune variabilă de cel mult n caractere sau bytes
- NVARCHAR2(n BYTE/CHAR) – la fel ca VARCHAR2 însă reține caractere din setul național (din alt alfabet, simblorui, etc.)

Tipuri de date DATE

Acestea sunt utilizate pentru a reține informații despre date, ore sau intervale de timp. Cele mai folosite sunt:

- DATE – pentru date calendaristice cu ani zile și luni

- TIMESTAMP(p) – pentru date și ore cu precizia de p milisecunde. Se pot utiliza și keywordurile WITH (LOCAL) TIMEZONE care precizează fusul orar în care dorim să reținem informația
- INTERVAL – acesta reține un interval de timp cu precizii variabile (YEAR TO MONTH sau DAY TO SECOND)

Alte tipuri din SQL

Cele trei supratipuri prezentate mai sus constituie cele mai des folosite tipuri din SQL ul normal însă avem la dispoziție multe alte modalități de stocare a datelor. Pentru stocarea id-urilor intrărilor din tabele se folosește tipul de date ROWID, pentru reținerea informațiilor binare avem tipurile LOB: BLOB pentru obiecte, sau CLOB pentru caractere sau BFILE pentru adrese de fișiere.

Tipuri noi adăugate în PLSQL

Deși tipurile utilizate în SQL oferă o varietate mare de posibilități pentru programatori, o dată cu introducerea PL/SQL-ului a apărut și nevoia de alte tipuri de date mai complexe care pot facilita scrierea blocurilor de cod.

Tipul de date BOOLEAN reține o variabilă ce poate fi True, False sau null. Acest tip ne permite să lucrăm mai ușor cu instrucțiunile IF sau LOOP însă nu poate fi utilizat în cadrul comenzilor SQL.

Tipul de date referință funcționează ca referințele din alte limbaje de programare și ne indică adresa unui anumit obiect din program. Cel mai des întâlnit caz de folosință are acestui tip este REF CURSORUL (care va fi explicat într-un tutoriat ulterior).

Tipurile de date PLS_INTEGER și BINARY_INTEGER permit stocarea numerelor întregi pe 32 de biți cu semn. Variabilele de acest tip sunt mult mai rapide ca cele de tipul NUMBER și permit rularea mai eficientă a codului.

De asemenea, ne putem declara subtipuri definite de noi prin ajutorul sintaxei SUBTYPE nume_subtip IS nume_supratip (constrângeri). Putem astfel:

- seta o anumită precizie pentru niște valori numerice sau șiruri de caractere fără a o declara explicit pentru fiecare variabilă

```
DECLARE
SUBTYPE an_nastere IS NUMBER(4);
v_an1 an_nastere := 1990
v_an2 an_nastere := 1982
v_an3 an_nastere := 2001
```

- asigura că variabilele sunt nenule fără a declara explicit de fiecare dată

```
DECLARE
SUBTYPE nume IS VARCHAR2(40) NOT NULL;
v_nume nume;
v_prenume nume;
v_initiala_prenume_tata nume;
```

- redenumi tipuri pentru a facilita înțelegerea codului de către alte persoane

```
DECLARE
SUBTYPE durata_proiect_zile IS INTERVAL DAY TO SECOND;
SUBTYPE ora IS TIMESTAMP(5) WITH LOCAL TIMEZONE;
```

Pentru a putea deduce mai ușor tipul variabilelor deja existente în sistem s-a introdus și atributul %TYPE. Acesta copiază tipul variabilei transmise prin nume în variabila pe care dorim să o folosim.

```
DECLARE
v_nume ELEVI.nume%TYPE;
v_nr_tel_elev CHAR(10);
v_nr_tel_parinte v_nr_tel_elev%TYPE;
```

Exerciții

a) a) Să se afișeze pe ecran diferența între datele de angajare pentru cel mai vechi și cel mai nou angajat

b) Să se afișeze pe ecran numele departamentului unde lucrează angajatul cu numele dat de la tastatură. Dacă nu există acesta să se afișeze un mesaj pe ecran.

c) Să se citească de la tastatură id-ul unui manager. Dacă acesta este manager la Administration, Marketing sau IT se va afișa numele complet al managerului. Dacă acesta este la Human Resources se va afișa salariul. Dacă este la Accounting se va afișa data angajării.

d) Să se afișeze numărul maxim de angajați care pot fi plătiți cu 20 000 de dolari. (Adică suma salariilor acestora este mai mică sau egală cu 20 000.

e) Să se determine cel mai bine plătit angajat născut după anul 2000 și departamentul al cărui manager are cel mai mic salariu. Să se afișeze dacă angajatul determinat aparține departamentului determinat.

f) Să se determine cele 2 orașe în care lucrează angajații cu media salariului cea mai mare respectiv cea mai mică. Să se afișeze cel care are dimensiunea mai mare (numărul de caractere din nume).