# Subprograme în PL/SQL

Subprogramele reprezintă structuri esențiale ale limbajului PL/SQL, permițând împărțirea logicii programului în componente mai mici, clar definite și reutilizabile. Ele pot fi proceduri sau funcții, fiecare având un rol particular în rezolvarea unor operații de procesare a datelor sau de implementare a unor algoritmi. Organizarea codului sub formă de subprograme contribuie la lizibilitate, mentenanță și modularitate în aplicațiile dezvoltate peste Oracle Database.

Un subprogram poate primi parametri de intrare sau ieșire, poate apela la rândul său alte subprograme, poate fi definit în interiorul unui bloc PL/SQL sau publicat ca obiect permanent în baza de date. PL/SQL pune la dispoziție mecanisme de compilare, gestionare și ștergere a subprogramelor, precum și posibilitatea de a analiza metadatele asociate acestora prin intermediul tabelelor de sistem precum USER_OBJECTS.

## Procedurile

Procedurile sunt subprograme care execută o acțiune, dar nu returnează o valoare prin instrucțiunea RETURN. Ele sunt utilizate frecvent pentru operații ce modifică starea bazei de date, pentru validări, procesări sau generarea unor rapoarte prin intermediul pachetelor și job-urilor. O procedură poate:

- primi parametri de diverse tipuri (IN, OUT, IN OUT)
- utiliza instrucțiuni SQL și PL/SQL
- apela alte proceduri sau funcții
- declanșa excepții și poate avea o secțiune EXCEPTION dedicată.

```
CREATE OR REPLACE PROCEDURE nume_proc
(parametru1 TIP1, parametru2 TIP2) IS
BEGIN
 corp_procedura
END nume_procedura;
```

# Funcțiile

Funcțiile sunt subprograme care returnează o valoare prin utilizarea cuvântului cheie RETURN. De regulă sunt folosite pentru operații de calcul, procesare sau determinări logice. Spre deosebire de proceduri, funcțiile pot fi utilizate direct în instrucțiuni SQL, cu condiția să nu modifice date (adică să nu conțină INSERT/UPDATE/DELETE).

```
CREATE OR REPLACE FUNCTION nume_functie
(parametru1 TIP1) RETURN TIPRETURN IS
  declaratii
BEGIN
  corp_functie;
END nume_functie;
```

# Parametri IN, OUT și IN OUT

Subprogramele PL/SQL permit trei moduri de transmitere a parametrilor:

## Parametru IN

- transmis din program către subprogram;
- nu poate fi modificat în interiorul subprogramului;
- este comportamentul default dacă nu este specificat alt mod.

## Parametru OUT

- folosit pentru a returna valori din subprogram;
- nu își păstrează valoarea inițială la intrarea în subprogram.

## Parametru IN OUT

- permite atât citirea valorii inițiale, cât și modificarea acesteia.

## Declararea subprogramelor

Subprogramele pot fi declarate:

1. În blocuri anonime

Sunt vizibile doar în interiorul blocului:

```
DECLARE
  PROCEDURE salut IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Salut!');
  END;
BEGIN
  salut;
END;
```

2. Ca obiecte permanente în baza de date (CREATE OR REPLACE)

Sunt accesibile oricărui utilizator cu privilegii.

3. În pachete (PACKAGE)

Vom discuta la un tutoriat ulterior.

## Compilarea subprogramelor

Oracle compilează automat codul PL/SQL la momentul execuției. Totuși, când dorim compilarea manuală (de exemplu după o modificare), folosim:

```
ALTER PROCEDURE nume_procedura COMPILE;
ALTER FUNCTION nume_functie COMPILE;
```

Dacă apar erori, acestea pot fi vizualizate cu:

```
SHOW ERRORS PROCEDURE nume_procedura;
SHOW ERRORS FUNCTION nume_functie;
```

Subprogramele pot fi eliminate definitiv cu instrucțiunile:

```
DROP PROCEDURE nume_procedura;
DROP FUNCTION nume_functie;
```

## Recursivitatea în subprograme

PL/SQL permite apelul recursiv, adică un subprogram se poate apela pe el însuși. Recursivitatea este utilă în algoritmi precum factorialul, parcurgerea arborescentă a datelor, generarea de structuri ierarhice etc.

Atunci când o procedură sau funcție trebuie să apeleze o altă procedură care nu a fost încă definită, se folosește forward declaration. Este utilă în declarațiile locale.

```
DECLARE
  PROCEDURE b;  -- Forward declaration

  PROCEDURE a IS
  BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Procedura A');
    b;
  END;

  PROCEDURE b IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Procedura B');
  END;
BEGIN
  a;
END;
```

## Tabela USER_OBJECTS

Oracle păstrează metadate despre obiectele fiecărui utilizator în tabela USER_OBJECTS. Aceasta este extrem de utilă când dorim să verificăm existența, starea sau tipul obiectelor PL/SQL.

Principalele coloane relevante sunt:

OBJECT_NAME – numele obiectului (procedură, funcție, pachet etc.).

OBJECT_TYPE – tipul obiectului, ex.: PROCEDURE, FUNCTION, PACKAGE.

STATUS – VALID / INVALID; un obiect INVALID necesită recompilare.

CREATED – data creării.

LAST_DDL_TIME – ultima dată când obiectul a fost modificat.

Exemplu de interogare:

```
SELECT object_name, object_type, status, last_ddl_time
FROM user_objects
WHERE object_type IN ('PROCEDURE', 'FUNCTION')
ORDER BY last_ddl_time DESC;
```

Această tabelă ajută programatorul să urmărească subprogramele existente, versiunea acestora și eventualele probleme de compilare.

## Exerciții

a) Să se scrie o procedură care primește numele unui angajat ca parametru și furnizează numele celor mai bine-plătiți 2 colegi din departamentul său
b) Să se scrie o cerere SQL care să ne dea informații despre angajații care au ca id numere prime. Se pot folosi funcții auxiliare
c) Să se creeze o procedură care furnizează numărul de angajați care lucrează într-o anumită țară al cărei nume este dat ca parametru
d) Să se creeze o funcție care primește codul unei regiuni și returnează numărul de tări din acea regiune în care nu lucrează niciun angajat.

a) Write a procedure that receives the name of an employee as a parameter and returns the names of the two best-payed colleagues in his department.
b) Write a SQL query that returns information about employees with prime id numbers. You can write auxilliary functions.
c) Write a procedure that returns the number of employees that work in a country given as a parameter.
d) Write a function that receives a region code as a parameter and returns the number of countries in that region that have 0 employees working within them.