UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiect nr. 154/323 cod SMIS – 4428 cofinanțat de prin Fondul European de Dezvoltare Regională “Investiții pentru viitorul dumneavoastră”.

Programul Operaţional Sectorial Creşterea Competitivităţii Economice - POS CCE

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Baze de date

25. Proceduri stocate. Instrucțiuni

Introducere

Procedura este un subprogram care execută un set de instrucțiuni, dar nu returnează direct o valoare către programul apelant. Rezultatele obținute prin procesarea datelor pot fi totuşi folosite în programul apelant, în funcție de modul de declarare a parametrilor. Parametrii sunt denumite variabilele care sunt disponibile atât pentru programul principal(apelant), cât şi pentru procedură(subprogram) şi care determină funcționalitatea şi rezultatele. Parametrii unei proceduri sunt opționali şi sunt declarați în momentul creării procedurii. Procedurile declarate în cadrul unui bloc PL/SQL au ciclul de viață atât cât există blocul(nestocate). După închiderea blocului, procedura se pierde, deoarece este stocată în memorie(nu şi pe serverul bazei de date). Pentru a fi recreată, se compilează din nou blocul şi se execută. O procedură stocată pe serverul bazei de date poate fi folosită oricând, chiar şi după închiderea sesiunii, deoarece rămâne creată în dicționar, ca orice obiect al bazei de date.

Sintaxa unei proceduri definită într-un bloc PL/SQL

- ***PROCEDURE procedure_name*** -- declararea procedurii
  [parameter_name [IN \|OUT \|IN OUT] parameter_type, …]
  IS / AS
  [procedure_declaration_section]
  BEGIN -- blocul procedurii
  procedure_executable_section
  [EXCEPTION
  procedure_exception_section]
  END [procedure_name]; -- sfârşitul declarării procedurii

  …..

  BEGIN -- blocul apelant

  DECLARE

  block_variables;

  BEGIN

  block_ executable_section

  procedure_name[parameters]; -- apelul procedurii

  [EXCEPTION

  block exceptions]

  END;

  END; -- sfârşitul blocului apelant

Cuvinte cheie

- ***procedure_name*** – este numele procedurii;
- ***parameter_name*** – reprezintă numele unui parametru formal din lista de parametri;
- ***parameter_type*** – reprezintă tipul unui parametru formal;
- ***procedure_declaration_section*** – reprezintă secțiunea de declarare de variabile pentru procedură;
- ***procedure_executable_section*** – reprezintă secțiunea executabilă a procedurii ;
- ***procedure_exception_section*** – reprezintă secțiunea de tratare a excepțiilor din cadrul procedurii ;
- ***block_variables*** - reprezintă secțiunea de declarare de variabile pentru blocul apelant ;
- ***block exceptions*** – reprezintă secțiunea de tratare a excepțiilor în cadrul blocului apelant;
- ***IN |OUT |IN OUT*** – specifică dacă parametrul poate fi referit sau modificat în interiorul sau exteriorul procedurii.

Exemplu:

- **Exemplul următor conține o procedură nestocată, care întoarce salariul maxim pentru un departament şi o funcție transmise ca parametri.**

```
DECLARE
PROCEDURE SALARIU (dep IN NUMBER, job IN VARCHAR2,
salariu IN OUT NUMBER) IS
        sal_max NUMBER;
BEGIN
     select max(salariu) into sal_max from angajati
     where id_dep = dep and functie=job group by id_dep;
     salariu:=sal_max;
EXCEPTION  when
    no_data_found  then
    dbms_output.put_line ('Nu a fost gasita nicio inregistrare');
END;
```

Sintaxa unei proceduri PL/SQL stocată pe server

```
CREATE [OR REPLACE] PROCEDURE procedure_name
        [ parameter_name [IN |OUT |IN OUT] parameter_type, … ]

[AUTHID {DEFINER | CURRENT_USER}]

[PRAGMA AUTONOMOUS_TRANSACTION]
IS /AS
        [declaration_section]


BEGIN
         executable_section
[EXCEPTION
         exception_section]
END [procedure_name];
```

Cuvinte cheie

- ***procedure_name*** – este numele procedurii;
  parameter_name – reprezintă numele unui parametru formal din lista de parametri;
- ***parameter_type*** – reprezintă tipul unui parametru formal;
- ***declaration_section*** – reprezintă secțiunea de declarare de variabile ;
- ***executable_section*** – reprezintă secțiunea executabilă a procedurii;
- ***exception_section*** – reprezintă secțiunea de tratare a excepțiilor ;
- ***[AUTHID {DEFINER | CURRENT_USER}]*** - specifică dacă o procedură stocată se execută cu drepturile celui care a creat-o (valoare implicită), sau ale utilizatorului curent ;
- ***[PRAGMA AUTONOMOUS_TRANSACTION]*** - specifică că execuția procedurii suspendă tranzacția curentă care se reia după terminarea execuției procedurii, adică într-o tranzacție imbricăm altă tranzacție, cu propriile sale *COMMIT* sau *ROLLBACK*;
- ***IN |OUT |IN OUT*** – specifică dacă parametrul poate fi referit sau modificat în interiorul sau exteriorul procedurii.

Exemplu:

În exemplul următor, se creează o procedură stocată VENIT, care calculează veniturile angajaților cu o vechime de peste 20 ani, pe fiecare departament.

```
CREATE or REPLACE PROCEDURE VENIT(dep IN NUMBER, data_referinta IN DATE DEFAULT
sysdate, venit IN OUT NUMBER) IS
depart VARCHAR2(30);
nr_dep number(2);
data_in date;
salar number;
BEGIN
select den_dep into depart from departamente where id_dep=dep;
select sum(salariu+nvl(comision,0)) into venit from angajati
      where id_dep = dep and add_months(data_ang,240) < data_referinta;
      dbms_output.put_line (rpad(depart,20) ||' '||rpad(data_in, 20)|| ' '||lpad(venit, 10) );
EXCEPTION when
          no_data_found then
          dbms_output.put_line('Nu a fost gasita nicio inregistrare');
END;
```

Programul apelant

```
DECLARE
        dep number(2);
        data_in date:=sysdate;
        salariu number:=0;
        total number:=0;
BEGIN
dbms_output.put_line ('= VENITURI PE DEPARTAMENTE  =');
FOR dep IN (select distinct id_dep from departamente  order by id_dep)
        LOOP
                venit(dep.id_dep, data_in,salariu);
                total:=total+nvl(salariu,0);
        END LOOP;
dbms_output.put_line ('Suma totala = '||total);
END;
```

Parametrii unei proceduri

- Sintaxa unui parametru formal este:

  parameter_name IN|OUT|IN OUT [NOCOPY]

  parameter_type [:= |DEFAULT |expression]

  unde:

  parameter_name - este numele parametrului;
  parameter_type - este tipul parametrului;

  DEFAULT - este valoarea implicită;

  expression - este o expresie atribuită parametrului;

  NOCOPY – specifică că parametrul se transmite prin referință (adresa), nu prin valoare şi este valabil pentru *OUT* şi *IN OUT* care se transmit implicit prin valoare.

- Parametrii unei proceduri sunt opționali şi sunt declarați în momentul creării procedurii. Denumirea trebuie să fie diferită de denumirea procedurii, să înceapă cu o literă, să nu conțină spații şi să aibă lungimea maxima de 30 caractere.

Parametrii unei proceduri - continuare

- Parametrii unei proceduri pot fi transmişi în două moduri, prin referință sau prin valoare.
- Există trei tipuri de parametri: *IN*, *OUT* şi *IN OUT* şi au următoarele semnificații:
- ***IN*** – parametrul poate fi referit în interiorul procedurii, dar nu poate fi modificat;
- ***OUT*** - parametrul nu poate fi referit în interiorul procedurii, dar poate fi modificat şi poate fi referit în afara procedurii(în programul apelant);
- ***IN OUT*** - parametrul poate fi referit în interiorul procedurii, poate fi modificat şi poate fi referit în afara procedurii(în programul apelant).
- Parametrul *IN* este tipul implicit. Un parametru *OUT* este inițializat ca *NULL,* iar procedura atribuie parametrului o valoare care poate fi referită în afara ei.

Pachete PL/SQL

- Un pachet(*package*) este o bibliotecă de obiecte stocate pe server, de tipul procedurilor, funcțiilor, cursoarelor, tipurilor de date, excepțiilor, variabilelor şi constantelor. Toate obiectele declarate în pachete sunt globale şi pot fi apelate din orice program PL/SQL, asemănător clasicelor variabile globale din alte limbaje de programare. Un pachet este compus din două secțiuni distincte, o secțiune de creare(*create package*), care conține specificațiile publice de declarare a conținutului(structura obiectelor) şi o secțiune care cuprinde corpul pachetului(*package body*), în care sunt descrise efectiv obiectele. Trebuie specificat faptul că un pachet este creat, la rândul lui, tot ca un obiect în dicționarul bazei de date.

  Principalele avantaje oferite de pachete sunt:

- modularitatea aplicațiilor;
- posibilitatea declarării de obiecte globale;
- îmbunătățirea performanțelor sistemului de gestiune ;
- Uşurința în proiectarea aplicațiilor;
- adăugarea de funcționalități noi.

Specificațiile unui pachet

- Specificațiile reprezintă partea publică a unui pachet şi au următoarea sintaxă:

  CREATE [OR REPLACE] PACKAGE package_name IS|AS
  global type and variable declarations
  subprogram specifications
  END [package_name]

unde:

- ***package _name*** – este numele pachetului;
- ***global type and variable declarations*** – reprezintă declarațiile globale ale cursoarelor, excepțiilor, constantelor, variabilelor şi tipurilor de date şi descrierea acestora din punct de vedere al structurii;
- ***subprogram specifications -*** reprezintă numele procedurilor şi funcțiilor cu parametrii formali aferenți.

Exemplu:

```
CREATE OR REPLACE PACKAGE salariu AS
data_calcul  date;
TYPE state_salarii IS TABLE OF state%ROWTYPE ;
v_state  state_salarii;
TYPE rec_personal IS RECORD (ecuson personal.id_ang%TYPE,
studii personal.studii%TYPE, salariu personal.salariu%TYPE, spor
personal.spor%TYPE, data_angajare personal.data_ang%TYPE) ;
PROCEDURE calcul_salariu(an IN number, luna IN number, id_ang
IN number, dep IN number, venit IN OUT number, retineri IN OUT number);
FUNCTION impozit(id_ang IN number, venit IN number, taxe
OUT number) RETURN number;
END salariu;
```

Corpul unui pachet

Corpul pachetului (*package body*) conține descrierea efectivă a procedurilor şi funcțiilor definite în specificații, având următoarea sintaxă:

CREATE [OR REPLACE] PACKAGE BODY package_name IS | AS
local type and variable declarations
subprogram bodies

[BEGIN

initialization statements

END]
END [package_name];

unde:

- ***package_name*** – este numele pachetului;
- ***local type and variable declarations*** – reprezintă declarațiile locale ale cursoarelor, excepțiilor, constantelor, variabilelor şi tipurilor de date;
- ***subprogram bodies*** - reprezintă codul sursă al procedurilor şi funcțiilor definite în specificații;
- ***initialization statements*** – reprezintă codul de inițializare.

Exemplu:

```
CREATE OR REPLACE PACKAGE BODY salariu AS
PROCEDURE calcul_salariu(an IN number, luna IN number, id_ang
      IN number, dep IN number, venit IN OUT number, retineri IN OUT number) IS
nr_ang number(3);
BEGIN
…….
END calcul_salariu;
FUNCTION impozit (id_ang IN number, venit IN number, taxe  OUT number) RETURN
number IS
cota_impozit number(5,2);
imp number;
BEGIN
……..
RETURN imp;
END impozit;
END salariu;
```

Restricții în definirea pachetelor

- În pachete, nu se permite declararea a două proceduri sau funcții cu acelaşi nume, dacă parametrii acestora diferă numai prin nume sau mod (*IN, OUT, IN OUT*). Trebuie ca cel puțin un parametru să fie de un alt tip, iar tipul nu trebuie să fie din aceeaşi familie (de exemplu, tipul *CHAR* este din aceeaşi familie cu *VARCHAR2)*. Aceeaşi situație este şi în cazul rezultatului returnat de o funcție. Erorile nu apar în momentul creării, ci în momentul executării, ceea ce creează o anumită confuzie.
- O variabilă, declarată globală în specificații, nu poate fi apelată tot în această secțiune, direct sau indirect, prin intermediul subprogramelor.

Informații despre pachete din dicționarul bazei de date

- Deoarece pachetele sunt create ca oricare alt obiect, din dicționarul bazei de date putem să aflăm informații despre ele făcând interogări pe view-urile sistemului de gestiune Oracle.
- De exemplu, dacă vrem să vedem toate pachetele create(care au specificații) de userul curent, data când au fost create, data ultimei modificări şi starea lor, putem executa următoarea cerere SQL:
- **SQL> SELECT object_name,created,last_ddl_time, status**

  FROM user_objects

  WHERE object_type = 'PACKAGE';
- Iar pentru listarea codului sursă a unui pachet, executăm cererea:
- **SQL>SELECT text FROM user_source**

  WHERE name = 'APEL' AND type='PACKAGE BODY'

  ORDER BY LINE;