



Curs 11

Funcții în PL/SQL (partea II)

Cuprins

Funcții în PL/SQL (partea II)

- 1. Recursivitate**
- 2. Utilizarea în expresii SQL a funcțiilor definite de utilizator**
- 3. Informații referitoare la subprograme**
- 4. Dependența subprogramelor**

1. Recursivitate

- *Un subprogram recursiv presupune că acesta se apelează pe el însuși.*
- În *Oracle* o problemă delicată este legată de locul unde se plasează un apel recursiv.
- De exemplu, dacă apelul este în interiorul unui cursor *FOR* sau între comenzile *OPEN* și *CLOSE*, atunci la fiecare apel este deschis alt cursor.
- În felul acesta, programul poate depăși limita pentru *OPEN_CURSORS* setată în parametrul de inițializare *Oracle*.

1. Recursivitate

Exemplu:

Să se calculeze recursiv al m -lea termen din șirul lui Fibonacci.

```
FUNCTION fibonacci(m INTEGER)  
RETURN INTEGER IS  
BEGIN  
  IF (m = 1) OR (m = 2) THEN  
    RETURN 1;  
  ELSE  
    RETURN fibonacci(m-1) + fibonacci(m-2);  
  END IF;  
END fibonacci;
```

1. Recursivitate

Declarații *forward*

- *Subprogramele sunt reciproc recursive dacă ele se apelează unul pe altul direct sau indirect.*
- Declarațiile *forward* permit definirea subprogramelor reciproc recursive.
- În *PL/SQL*, un identificador trebuie declarat înainte de a-l folosi.
- De asemenea, un subprogram trebuie declarat înainte de a-l apela.

1. Recursivitate

```
PROCEDURE alfa ( ... ) IS
BEGIN
  beta( ... );      -- apel incorect
  ...
END;
PROCEDURE beta ( ... ) IS
BEGIN
  ...
END;
```

- Procedura *beta* nu poate fi apelată deoarece nu este încă declarată.
- Problema se poate rezolva simplu, inversând ordinea celor două proceduri.
- Această soluție nu este eficientă întotdeauna.

1. Recursivitate

- *PL/SQL* permite un tip special de declarare a unui subprogram numit *forward*.
- El constă dintr-o specificare de subprogram terminată prin “;”.

```
PROCEDURE beta ( ... ); -- declarație forward
```

```
..
```

```
PROCEDURE alfa ( ... ) IS
```

```
BEGIN
```

```
    beta( ... );
```

```
    ...
```

```
END;
```

```
PROCEDURE beta ( ... ) IS
```

```
BEGIN
```

```
    ...
```

```
END;
```

1. Recursivitate

Se pot folosi declarații *forward*:

- pentru a defini subprograme într-o anumită ordine logică
- pentru a defini subprograme reciproc recursive
- pentru a grupa subprograme într-un pachet

1. Recursivitate

- Lista parametrilor formali din declarația *forward* trebuie să fie identică cu cea corespunzătoare corpului subprogramului.
- Corpul subprogramului poate apărea oriunde după declarația sa *forward*, dar trebuie să rămână în aceeași unitate de program.

Cuprins

Funcții în PL/SQL (partea II)

1. Recursivitate
2. Utilizarea în expresii SQL a funcțiilor definite de utilizator
3. Informații referitoare la subprograme
4. Dependența subprogramelor

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

- Începând cu varianta **Oracle** Release 7.1, o funcție stocată poate fi referită într-o comandă **SQL** la fel ca orice funcție standard furnizată de sistem (built-in function), dar cu anumite restricții.
- Funcțiile **PL/SQL** definite de utilizator pot fi apelate din orice expresie **SQL** în care pot fi folosite funcții **SQL** standard.

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

• Funcțiile *PL/SQL* pot să apară în:

1. lista de selecție a comenzii *SELECT*
2. condiția clauzelor *WHERE* și *HAVING*
3. clauzele *CONNECT BY*, *START WITH*, *ORDER BY* și *GROUP BY*
4. clauza *VALUES* a comenzii *INSERT*
5. clauza *SET* a comenzii *UPDATE*

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

Exemplu:

Să se afișeze salariile (nume angajat, salariu, job) a căror valoare este mai mare decât valoarea medie a tuturor salariilor din firma.

```
create or replace function "VALOARE_MEDIE2"  
return NUMBER is  
  v_val_mediu emp.sal%TYPE;  
begin  
  SELECT AVG(sal)  
  INTO  v_val_mediu  
  FROM  emp;  
  RETURN v_val_mediu;  
end;
```

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

Referirea acestei funcții într-o comandă SQL se poate face prin:

```
SELECT ename, sal, job  
FROM emp  
WHERE sal >= valoare_medie2;
```

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

Există restricții referitoare la folosirea funcțiilor definite de utilizator într-o comandă **SQL**. Câteva dintre acestea, care s-au păstrat și pentru versiunile ulterioare **Oracle9i**:

1. funcția definită de utilizator trebuie să fie o funcție stocată (procedurile stocate nu pot fi apelate în expresii **SQL**), nu poate fi locală unui alt bloc
2. funcția definită de utilizator trebuie să fie o funcție linie și nu una grup (restricția dispare în **Oracle9i**)
3. funcția apelată dintr-o comandă **SELECT**, sau din comenzi DML: **INSERT**, **UPDATE** și **DELETE** nu poate modifica tabelele bazei de date

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

4. funcția apelată dintr-o comandă **UPDATE** sau **DELETE** nu poate interoga sau modifica tabele ale bazei reactualizate chiar de aceste comenzi (**table mutating**)
5. funcția apelată din comenzile **SELECT**, **INSERT**, **UPDATE** sau **DELETE** nu poate executa comenzi LCD (**COMMIT**), **ALTER SYSTEM**, **SET ROLE** sau comenzi LDD (**CREATE**)
6. funcția nu poate să apară în clauza CHECK a unei comenzi **CREATE/ALTER TABLE**
7. funcția nu poate fi folosită pentru a specifica o valoare implicită pentru o coloană în cadrul unei comenzi **CREATE/ALTER TABLE**

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

8. funcția poate fi utilizată într-o comandă **SQL** numai de către proprietarul funcției sau de utilizatorul care are privilegiul EXECUTE asupra acesteia
9. funcția definită de utilizator, apelabilă dintr-o comandă **SQL**, trebuie să aibă doar parametri de tip IN, cei de tip OUT și IN OUT nefiind acceptați
10. parametrii unei funcții **PL/SQL** apelate dintr-o comandă **SQL** trebuie să fie specificați prin poziție (specificarea prin nume nefiind permisă)

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

11. parametrii formali ai unui subprogram funcție trebuie să fie de tip specific bazei de date (NUMBER, CHAR, VARCHAR2, ROWID, LONG, LONGROW, DATE), nu tipuri **PL/SQL** (**BOOLEAN** sau **RECORD**)
12. tipul returnat de un subprogram funcție trebuie să fie un tip intern pentru server, nu un tip **PL/SQL**
13. funcția nu poate apela un subprogram care nu respectă restricțiile anterioare

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

Exemplu de restricție a utilizării funcțiilor în comenzi SQL:

Restricția

4. funcția apelată dintr-o comandă **UPDATE** sau **DELETE** nu poate interoga sau modifica tabele ale bazei reactualizate chiar de aceste comenzi (**table mutating**)

- Comanda **UPDATE** va returna o eroare deoarece tabelul *emp* este **mutating**.

2. Utilizarea în expresii SQL a funcțiilor definite de utilizator

The screenshot shows the Oracle SQL Developer interface. At the top right, the name of the object being edited is "CALCUL". Below this, there are tabs for "Code", "Dependencies", "Errors", and "Grants". The "Code" tab is active, showing a SQL script with the following content:

```
1 create or replace function "CALCUL"  
2 (p_sal in NUMBER)  
3 return NUMBER  
4 is  
5 begin  
6     INSERT INTO emp(empno, ename, job, hiredate, sal);  
7     VALUES (1358, 'popa', SYSDATE, 700000);  
8     RETURN (p_sal*7);  
9 end;
```

Below the code editor, there are several buttons: "Save & Compile", "Find & Replace", "Undo", "Redo", "Download Source", and "Drop".

This section shows the execution controls and the SQL statement being executed. On the left, there is a "Rows" dropdown set to "10", a "Save" button, and a "Run" button. Below these, the SQL statement is displayed:

```
UPDATE emp  
SET sal = calcul (550000)  
WHERE empno = 7531;
```

The "Results" tab is active, displaying an error message in a yellow box:

```
ORA-06575: Package or function CALCUL is in an invalid state
```

At the bottom left of the interface, the execution time is shown as "0.04 seconds".

Cuprins

Funcții în PL/SQL (partea II)

1. Recursivitate
2. Utilizarea în expresii SQL a funcțiilor definite de utilizator
3. Informații referitoare la subprograme
4. Dependența subprogramelor

3. Informații referitoare la subprograme

- Informațiile referitoare la subprogramele **PL/SQL** și modul de acces la aceste informații sunt următoarele:
 - codul sursă, utilizând vizualizarea **USER_SOURCE** din dicționarul datelor (DD)
 - informații generale, utilizând vizualizarea **USER_OBJECTS** din dicționarul datelor
 - tipul parametrilor (IN, OUT, IN OUT), utilizând comanda DESCRIBE
 - erorile la compilare, utilizând vizualizarea **USER_ERRORS** din dicționarul datelor sau comanda SHOW ERRORS
 - informații de depanare, utilizând pachetul **DBMS_OUTPUT**.

3. Informații referitoare la subprograme

Vizualizarea **USER_OBJECTS** conține informații generale despre toate obiectele manipulate în BD, în particular și despre subprogramele stocate.

Vizualizarea **USER_OBJECTS** are următoarele câmpuri:

1. **OBJECT_NAME** – numele obiectului
2. **OBJECT_TYPE** – tipul obiectului (PROCEDURE, FUNCTION etc.)
3. **OBJECT_ID** – identificator intern al obiectului
4. **CREATED** – data când obiectul a fost creat
5. **LAST_DDL_TIME** – data ultimei modificări a obiectului
6. **TIMESTAMP** – data și momentul ultimei recompilări
7. **STATUS** – starea obiectului (VALID sau INVALID)

3. Informații referitoare la subprograme

Pentru a verifica dacă recompilarea explicită (**ALTER**) sau implicită a avut succes se poate verifica starea subprogramelor utilizând **USER_OBJECTS**.

Orice obiect are o stare (status) sesizată în DD, care poate fi:

- **VALID** (obiectul a fost compilat și poate fi folosit când este referit)
- **INVALID** (obiectul trebuie compilat înainte de a fi folosit)

3. Informații referitoare la subprograme

Exemplu:

Să se listeze procedurile și funcțiile deținute de utilizatorul curent, precum și starea acestora.

The screenshot shows the Oracle Application Express interface. The top navigation bar includes 'Home', 'Application Builder', 'SQL Workshop', 'Team Development', and 'Administration'. The current page is 'SQL Workshop' > 'SQL Commands'. Below the navigation is a 'Rows' selector set to '10', a 'Save' button, and a 'Run' button. The SQL query entered is: `SELECT OBJECT_NAME, OBJECT_TYPE, STATUS FROM USER_OBJECTS WHERE OBJECT_TYPE IN ('PROCEDURE', 'FUNCTION');`. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table with three columns: 'OBJECT_NAME', 'OBJECT_TYPE', and 'STATUS'. The table contains 10 rows of data. At the bottom of the results, it says 'More than 10 rows available. Increase rows selector to view more rows.' and '10 rows returned in 0.04 seconds' with a 'Download' link.

OBJECT_NAME	OBJECT_TYPE	STATUS
VALOARE_MEDIE2	FUNCTION	VALID
VALOARE_MEDIE1	FUNCTION	VALID
VALOARE_MEDIE	FUNCTION	VALID
VALOARE.MEDIE	FUNCTION	VALID
RAISE_SALARY	PROCEDURE	INVALID
RAISE_SAL	PROCEDURE	INVALID
QUERY_EMP	PROCEDURE	VALID
GET_SAL	FUNCTION	VALID
FORMAT_PHONE	PROCEDURE	VALID
EXPLAIN_OUT	PROCEDURE	VALID

3. Informații referitoare la subprograme

După ce subprogramul a fost creat, codul sursă al acestuia poate fi obținut consultând vizualizarea **USER_SOURCE** din DD, care are următoarele câmpuri:

1. **NAME** - numele obiectului
2. **TYPE** - tipul obiectului
3. **LINE** - numărul liniei din codul sursă
4. **TEXT** - textul liniilor codului sursă

3. Informații referitoare la subprograme

Exemplu:

Să se afișeze codul complet pentru funcția
valoare_medie.

```
SELECT TEXT  
FROM USER_SOURCE  
WHERE NAME = 'valoare_medie'  
ORDER BY LINE;
```

3. Informații referitoare la subprograme

Exemplu:

Să se scrie o procedură care recompilează toate obiectele invalide din schema personală.

```
CREATE OR REPLACE PROCEDURE sterge IS  
CURSOR obj_curs IS  
  SELECT OBJECT_TYPE, OBJECT_NAME  
  FROM   USER_OBJECTS  
  WHERE  STATUS = 'INVALID'  
  AND    OBJECT_TYPE IN ('PROCEDURE', 'FUNCTION',  
  PACKAGE', 'PACKAGE BODY', 'VIEW');  
BEGIN  
  FOR obj_rec IN obj_curs LOOP  
    DBMS_DDL.ALTER_COMPILE(obj_rec.OBJECT_TYPE,  
                          USER, obj_rec.OBJECT_NAME);  
  END LOOP;  
END sterge;
```

3. Informații referitoare la subprograme

- Dacă se recompilează un obiect **PL/SQL**, atunci server-ul va recompila orice obiect invalid de care depinde.
- Dacă recompilarea automată implicită a procedurilor locale dependente are probleme, atunci starea obiectului va rămâne **INVALID** și server-ul **Oracle** semnalează eroare.

3. Informații referitoare la subprograme

Deci:

- este preferabil ca recompilarea să fie manuală (recompilare explicită utilizând comanda **ALTER: PROCEDURE, FUNCTION, TRIGGER, PACKAGE**) cu opțiunea **COMPILE**
- este necesar ca recompilarea să se facă cât mai repede, după definirea unei schimbări referitoare la obiectele bazei

Cuprins

Funcții în PL/SQL(partea II)

1. Recursivitate
2. Utilizarea în expresii SQL a funcțiilor definite de utilizator
3. Informații referitoare la subprograme
4. **Dependența subprogramelor**

4. Dependența subprogramelor

- Când este compilat un subprogram, toate obiectele **Oracle** care sunt referite vor fi înregistrate în **dicționarul datelor (DD)**.
- Subprogramul este dependent de aceste obiecte.
- *Un subprogram care are erori la compilare este marcat ca “invalid” în dicționarul datelor.*
- Un subprogram stocat poate deveni, de asemenea, *invalid dacă o operație **LDD** este executată asupra unui obiect de care depinde.*

4. Dependența subprogramelor

<u>Obiecte dependente</u>	<u>Obiecte referite</u>
<i>View</i>	<i>Table</i>
<i>Procedure</i>	<i>View</i>
<i>Function</i>	<i>Procedure</i>
<i>Package Specification</i>	<i>Function</i>
<i>Package Body</i>	<i>Synonym</i>
<i>Database Trigger</i>	<i>Package Specification</i>

4. Dependența subprogramelor

° Dacă se modifică definiția unui obiect referit, obiectul dependent poate (sau nu) să continue să funcționeze normal.

Există două tipuri de dependențe:

1. **dependență directă**, în care obiectul dependent (procedure sau function) face referință direct la un table, view, sequence, procedure, function.
2. **dependență indirectă**, în care obiectul dependent (procedure sau function) face referință indirect la un table, view, sequence, procedure, function prin intermediul unui view, procedure sau function.

Testare

1. Exemple de funcții
2. Probleme propuse spre rezolvare

1. Exemple de funcții

Funcție care calculează circumferința unui cerc:

```
FUNCTION circumf (angle NUMBER:=360,radius  
NUMBER)
```

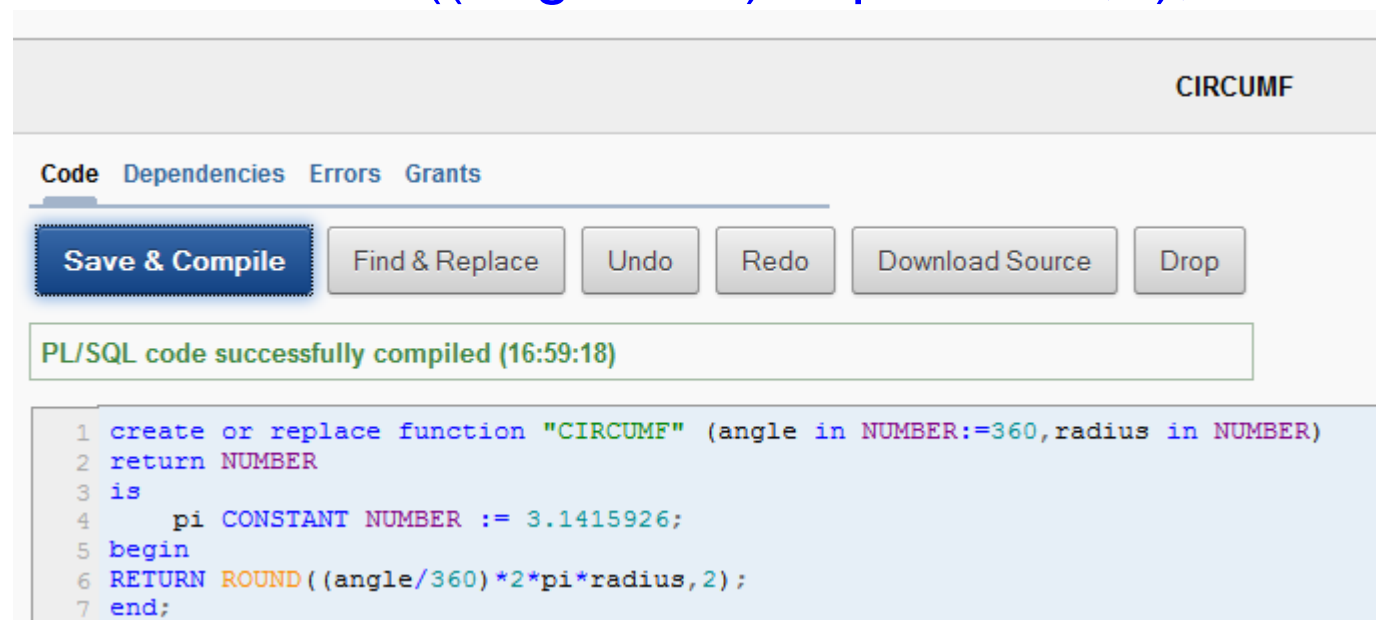
```
RETURN NUMBER IS
```

```
    pi CONSTANT NUMBER := 3.1415926;
```

```
BEGIN
```

```
    RETURN ROUND((angle/360)*2*pi*radius,2);
```

```
END;
```



The screenshot displays a web-based interface for PL/SQL code execution. At the top right, the function name "CIRCUMF" is visible. Below it, there are tabs for "Code", "Dependencies", "Errors", and "Grants". A toolbar contains buttons for "Save & Compile", "Find & Replace", "Undo", "Redo", "Download Source", and "Drop". A green message box indicates "PL/SQL code successfully compiled (16:59:18)". The code editor shows the following PL/SQL code:

```
1 create or replace function "CIRCUMF" (angle in NUMBER:=360,radius in NUMBER)
2 return NUMBER
3 is
4     pi CONSTANT NUMBER := 3.1415926;
5 begin
6 RETURN ROUND((angle/360)*2*pi*radius,2);
7 end;
```

1. Exemple de funcții

Funcție care verifica daca un numar este par

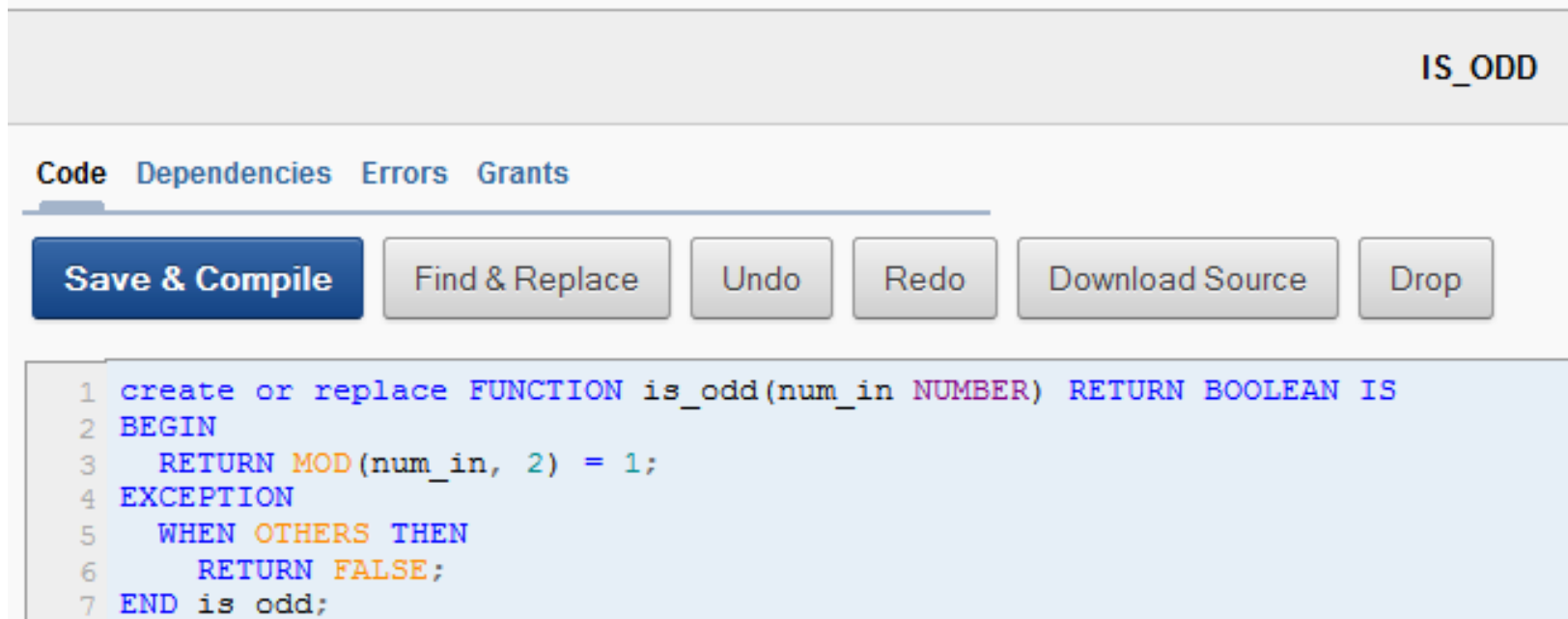
```
CREATE OR REPLACE FUNCTION is_even(num_in NUMBER)
  RETURN BOOLEAN IS
BEGIN
  IF MOD(num_in, 2) = 0 THEN
    RETURN TRUE;
  END IF;
  EXCEPTION
  WHEN OTHERS THEN
    RETURN FALSE;
END is_even;
```



The screenshot shows a database IDE interface. At the top right, the object name 'IS_EVEN' is displayed. Below it, there are tabs for 'Code', 'Dependencies', 'Errors', and 'Grants'. A toolbar contains buttons for 'Save & Compile', 'Find & Replace', 'Undo', 'Redo', 'Download Source', and 'Drop'. The main area displays the SQL code for the 'is_even' function, with line numbers 1 through 9 on the left. The code is: 1 create or replace FUNCTION is_even(num_in NUMBER) RETURN BOOLEAN IS, 2 BEGIN, 3 IF MOD(num_in, 2) = 0 THEN, 4 RETURN TRUE;, 5 END IF;, 6 EXCEPTION, 7 WHEN OTHERS THEN, 8 RETURN FALSE;, 9 END is_even;.

1. Exemple de funcții

° Funcție care verifica dacă un număr este impar



The screenshot displays a database IDE interface for a function named `IS_ODD`. The interface includes a header with the function name, a menu bar with options like `Code`, `Dependencies`, `Errors`, and `Grants`, and a toolbar with buttons for `Save & Compile`, `Find & Replace`, `Undo`, `Redo`, `Download Source`, and `Drop`. The main area shows the following SQL code:

```
1 create or replace FUNCTION is_odd(num_in NUMBER) RETURN BOOLEAN IS
2 BEGIN
3     RETURN MOD(num_in, 2) = 1;
4 EXCEPTION
5     WHEN OTHERS THEN
6         RETURN FALSE;
7 END is_odd;
```

Testare

1. Exemple de funcții
2. Probleme propuse spre rezolvare

2. Probleme propuse spre rezolvare

Să se creeze tabelele Angajati_pnu, Departamente_pnu (în șirul de caractere "pnu", p reprezintă prima literă a prenumelui, iar nu reprezintă primele două litere ale numelui dumneavoastră).

DEPARTAMENTE_PNU

department_id number(3) cheie primara (PK)
department_id varchar2(20)
manager_id varchar2(3)
location_id varchar2(100)

ANGAJATI_PNU

employee_id number(3) cheie primara (PK)
department_id number(3) referinta (FK) la tabela **DEPARTAMENTE_PNU**
last_name varchar2(40)
first_name varchar2(40)
function varchar2(25)
salary number(7)
manager_id varchar2(3)
hiredate date
commission_pct number(5)

2. Probleme propuse spre rezolvare

1. Să se declare o procedură locală într-un bloc **PL/SQL** anonim prin care să se introducă în tabelul DEPARTAMENTE_PNU o nouă înregistrare precizând, prin intermediul parametrilor, valori pentru toate câmpurile.

Invocați procedura în cadrul blocului.

Interogați tabelul DEPARTAMENTE_PNU și apoi anulați modificările (**ROLLBACK**).

2. Probleme propuse spre rezolvare

° 2. Să se declare o procedură locală care are parametrii următori:

- p_rezultat (parametru de tip **OUT**) de tipul coloanei last_name din tabelul ANGAJATI_PNU;
- p_comision (parametru de tip **IN**) de tipul coloanei commission_pct din ANGAJATI_PNU, inițializat cu **NULL**;
- p_cod (parametru de tip **IN**) de tipul coloanei employee_id din ANGAJATI_PNU, inițializat cu **NULL**.
Dacă p_comision nu este **NULL** atunci în p_rezultat se va memora numele salariatului care are salariul maxim printre salariații având comisionul respectiv.
În caz contrar, în p_rezultat se va memora numele salariatului al cărui cod are valoarea dată la apelarea procedurii.

2. Probleme propuse spre rezolvare

3. Să se creeze o procedură stocată fără parametri care afișează un mesaj “Programare PL/SQL”, ziua de astăzi în formatul DD-MONTH-YYYY și ora curentă, precum și ziua de ieri în formatul DD-MON-YYYY.

2. Probleme propuse spre rezolvare

4. Să se creeze o procedură stocată care pentru un anumit cod de departament (dat ca parametru) calculează prin intermediul unor funcții locale numărul de salariați care lucrează în el, suma salariilor și numărul managerilor salariaților care lucrează în departamentul respectiv.

2. Probleme propuse spre rezolvare

5. Să se creeze o funcție stocată care determină numărul de salariați din problema4_pnu angajați după 1995, într-un departament dat ca parametru.
Să se apeleze această funcție într-un bloc PL/SQL.

2. Probleme propuse spre rezolvare

6. Să se calculeze recursiv numărul de permutări ale unei mulțimi cu n elemente, unde n va fi transmis ca parametru.

2. Probleme propuse spre rezolvare

7. Să se afișeze numele, job-ul și salariul angajaților al căror salariu este mai mare decât media salariilor din tabelul `angajati_pnu`.



Întrebări?