Baze de Date II

(Proiectarea Sistemelor cu Baze de Date)

Cornelia TUDORIE

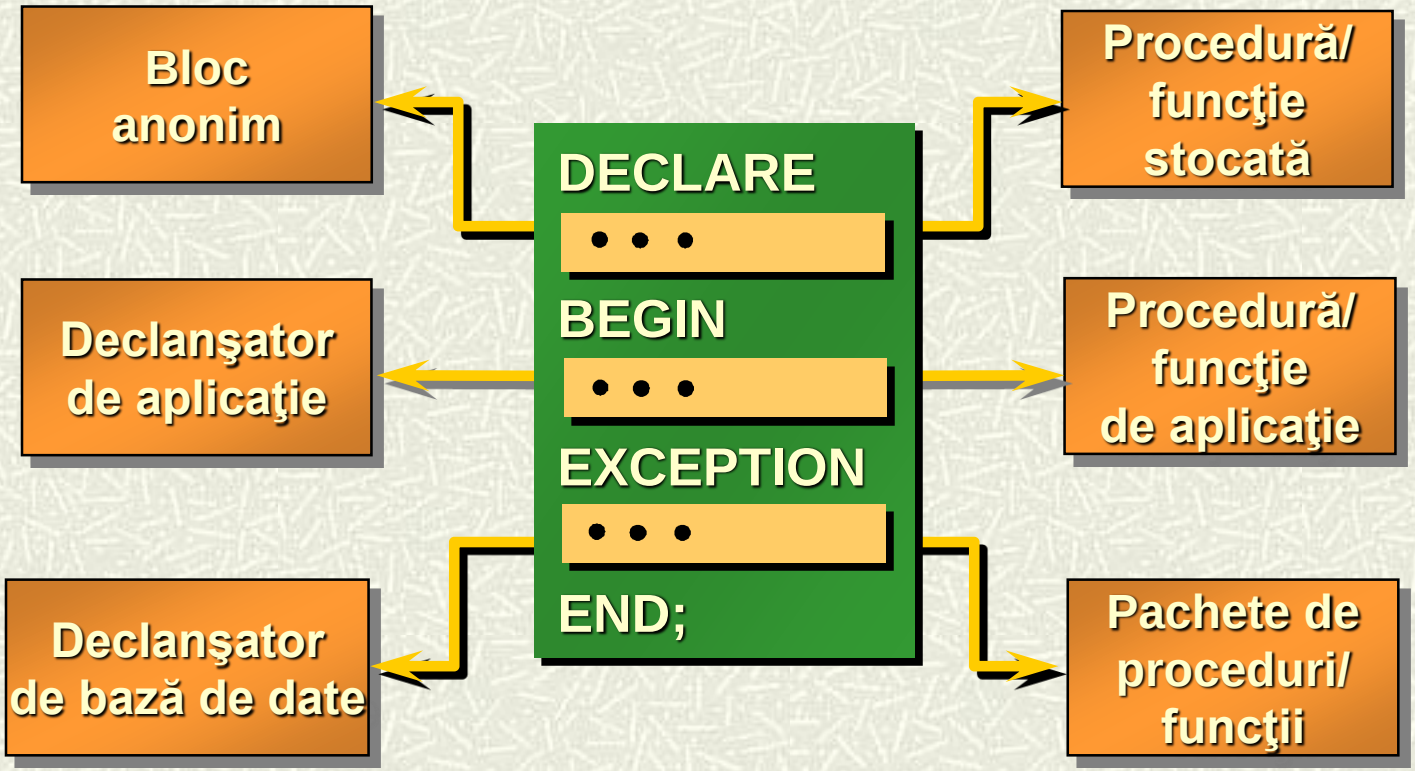
Proiectarea Sistemelor cu Baze de Date - Cuprins

A. **Modelul Relaţional.**
B. **Limbajul SQL. Lucrul cu tabele.**
C. **Sisteme cu baze de date. Performanţa.**
D. **Obiecte ale sistemului de baze de date. Performanţa în utilizare**
E. **Programe pentru baze de date. Performanţa în programare.**
F. **Protecţia datelor.**
G. **Sisteme Informatice. Proiectarea sistemelor de baze de date.**

E. Programe pentru baze de date. Performanţa în programare.

1. Fişiere de comenzi
2. PL/SQL. Tipuri de programe
3. Blocuri anonime
4. Subprograme stocate. Pachete
5. Declanşatori
6. Tranzacţii

E. 4. Subprograme stocate. Pachete.



PL/SQL, limbaj bazat pe blocuri

Blocuri: ***anonime*** **(principale sau imbricate)**

cu nume**:**

- **subprograme (funcţii, proceduri)**
- **subprograme stocate (catalogate)**
- **pachete**
- **declanşatori**

PL/SQL, limbaj bazat pe blocuri

Bloc anonim

[DECLARE]
sectiune declarativa
BEGIN
sectiune executabila
[EXCEPTION]
sectiune de exceptii
END;

Bloc cu nume

antet
IS
sectiune declarativa
BEGIN
sectiune executabila
[EXCEPTION]
sectiune de exceptii
END;

PL/SQL, limbaj bazat pe blocuri

Bloc anonim

```
DECLARE
        astazi  DATE DEFAULT SYSDATE;
BEGIN
        dbms_output.put_line('Buna ziua! Suntem in data de' || astazi);
END;
```

Bloc cu nume

```
CREATE OR REPLACE PROCEDURE salut
IS
        astazi  DATE DEFAULT SYSDATE;
BEGIN
        dbms_output.put_line('Buna ziua! Suntem in data de' || astazi);
END;
```

Subprograme

***Subprogram*:**

- local: - definit în interiorul unei alte unităţi de programare (la sfârşitul secţiunii declarative!!!)

- definit în cadrul unui pachet

- stocat, ca obiect al bazei de date

Subprograme

***Subprogram stocat* :**

- set de instrucţiuni PL/SQL şi comenzi SQL grupate logic, care efectuează o anumită sarcină

- este considerat obiect al schemei şi este stocat în baza de date

- poate fi executat direct sau inclus în aplicaţie

Subprograme

Avantaje subprograme stocate :

- **flexibilitate în programare**
- **modularizarea programelor**
- **independenţa faţă de aplicaţie**
- **accesibilitate**
- **reutilizare (fără recomplilare)**
- **securitatea bazei de date**

Subprograme

Componente :

- **antetul** (declararea numelui subprogramului şi a parametrilor)

- **corpul** (blocul de definire a subprogramului)

Crearea procedurilor locale

PROCEDURE *procedura* **[(** *<lista parametri>* **)] IS**
[*<declaraţii>* **]**
BEGIN
<secţiune executabilă>
[EXCEPTION
<secţiune de excepţii> **]**
END [*procedura* **];**

<lista parametri> **:**
parametru **[IN | OUT | IN OUT]** *tip* **[DEFAULT** *valoare* **] , ……**

Crearea procedurilor locale

Exemplu

```
procedure prima
          ( p_deptno in number, p_procent in number )  is
begin

 update emp set sal = sal + (sal * p_procent * 0.01)
       where deptno = p_deptno;
 commit;

end prima;
```

Crearea procedurilor stocate

CREATE [OR REPLACE] PROCEDURE *procedura* **[(** *<lista parametri>* **)] IS**
    [*<declaraţii>* **]**
BEGIN
    <secţiune executabilă>
[EXCEPTION
    <secţiune de excepţii> **]**
END;

<lista parametri> **:**
    parametru **[IN | OUT | IN OUT]** *tip* **[DEFAULT** *valoare* **] , ……**

Crearea procedurilor stocate

Exemplu

```
CREATE PROCEDURE prima
    ( p_deptno in number, p_procent number DEFAULT 10)  is
begin

 update emp set sal = sal + (sal * p_procent * 0.01)
       where deptno = p_deptno;
 commit;

end;
```

Apelul procedurilor stocate

Din program :

procedura **[(** *<lista parametri actuali>* **)] ;**

Notaţia poziţională :

<lista parametri actuali> **:**
valoare parametru **,** *valoare parametru* **,** **……**

Notaţia prin nume :

<lista parametri actuali> **:**
parametru **=>** *valoare parametru* **,** **……**

Apelul procedurilor stocate

Exemple

```
prima(20,15);
```

sau

```
prima(p_procent => 15, p_deptno => 20);
```

sau

```
prima(p_deptno => 20, p_procent => 15);
```

```
prima(20,10);
```

sau

```
prima(20);
```

Apelul procedurilor stocate

Din SQL*Plus :

EXECUTE *procedura* **[(** *<lista parametri actuali>* **)] ;**

Notaţia poziţională :

<lista parametri actuali> **:**
valoare parametru **,** *valoare parametru* **,** **……**

Notaţia prin nume :

<lista parametri actuali> **:**
parametru **=>** *valoare parametru* **,** **……**

Apelul procedurilor stocate

Exemple

```
SQL> EXECUTE prima(20,15)
```

```
SQL> EXECUTE prima(20)
```

Crearea funcţiilor stocate

CREATE [OR REPLACE] FUNCTION *functie* **([** *<lista parametri>* **])**
RETURN *tip* **IS**
[*<declaraţii>* **]**
BEGIN
<secţiune executabilă>
[EXCEPTION
<secţiune de excepţii> **]**
END;

<lista parametri> **:**
parametru **[IN]** *tip* **[DEFAULT** *valoare* **]** **, ……**

! *<secţiune executabilă>* ***trebuie să conţină o instrucţiune*** **RETURN** *<valoare>*

Crearea funcţiilor stocate

Exemplu

```
CREATE FUNCTION impozit
    ( p_emp number )  RETURN number IS
v_grade salgrade.grade%TYPE;
v_sal   emp.sal%TYPE;
BEGIN
  SELECT grade, sal INTO v_grade, v_sal FROM salgrade,emp
     WHERE empno = p_emp AND sal between losal and hisal;
  CASE v_grade
     WHEN 1 THEN RETURN v_sal*0.15;
     WHEN 2 THEN RETURN v_sal*0.20;
     WHEN 3 THEN RETURN v_sal*0.30;
     ELSE RETURN v_sal*0.40;
  END CASE;
END;
```

Apelul funcţiilor stocate

În expresii :

functia ([*<lista parametri actuali>*]) ;

Notaţia poziţională :

<lista parametri actuali> :
valoare parametru , *valoare parametru* , ……

~~**Notaţia prin nume**~~

Apelul funcţiilor stocate

Exemplu

```
SQL> set serveroutput on
SQL> begin
  2  dbms_output.put_line(impozit(7566));
  3  end;
  4  /
892.5

PL/SQL procedure successfully completed.
```

Obţinerea de informaţii asupra subprogramelor

Dicţionarul de date

disponibile pentru utilizator :

- **USER_OBJECTS**
- **USER_SOURCE**
- **USER_DEPENDENCIES**
- **ALL_SOURCE**

. . . . . .

Obţinerea de informaţii asupra subprogramelor

Exemplu

```
SQL> select * from user_source where name='PRIMA';
OWNER  NAME   TYPE         LINE TEXT
------ ------ ----------- ---- --------------------------------
SCOTT  PRIMA  PROCEDURE       1 procedure prima
SCOTT  PRIMA  PROCEDURE       2 ( p_deptno in number, p_procent number DEFAULT 10)  is
SCOTT  PRIMA  PROCEDURE       3 begin
SCOTT  PRIMA  PROCEDURE       4
SCOTT  PRIMA  PROCEDURE       5 update emp set sal = sal + (sal * p_procent * 0.01)
SCOTT  PRIMA  PROCEDURE       6        where deptno = p_deptno;
SCOTT  PRIMA  PROCEDURE       7 commit;
SCOTT  PRIMA  PROCEDURE       8
SCOTT  PRIMA  PROCEDURE       9 end;

9 rows selected.
```

Obţinerea de informaţii asupra subprogramelor

Exemplu

```
SQL> select * from USER_OBJECTS ;

OBJECT_NAME OBJECT_ID OBJECT_TYPE     CREATED   LAST_DDL   TIMESTAMP           STATUS
----------- --------- ----------- ----------- --------- ------------------- -----------

EMP             11454 TABLE          08-APR-03 08-APR-03  2003-04-08:16:12:58  VALID
EMP_EMPNO_PK    11455 INDEX          08-APR-03 08-APR-03  2003-04-08:16:12:58  VALID
IMPOZIT         11474 FUNCTION       10-APR-03 10-APR-03  2003-04-10:18:09:23  VALID
PRIMA           11473 PROCEDURE      08-APR-03 08-APR-03  2003-04-08:16:57:22  VALID
PROD            10909 TABLE          05-MAR-03 05-MAR-03  2003-03-05:15:05:34  VALID
PROD_PK         10910 INDEX          05-MAR-03 05-MAR-03  2003-03-05:15:05:34  VALID
SALES           11472 VIEW           08-APR-03 08-APR-03  2003-04-08:16:13:00  VALID
SALGRADE        11457 TABLE          08-APR-03 08-APR-03  2003-04-08:16:12:58  VALID
VANZARE         10911 TABLE          05-MAR-03 05-MAR-03  2003-03-05:15:05:34  VALID
. . . . . .
```

Obţinerea de informaţii asupra subprogramelor

Exemplu

```
SQL> select * from USER_DEPENDENCIES;

NAME       TYPE          REFERENCED_OWNER REFERENCED_NAME REFERENCED_TYPE
---------- ------------ ---------------- --------------- ---------------
PRIMA      PROCEDURE     SCOTT            EMP             TABLE
IMPOZIT    FUNCTION      SCOTT            EMP             TABLE
IMPOZIT    FUNCTION      SCOTT            SALGRADE        TABLE
SALES      VIEW          SCOTT            ORD             TABLE
SALES      VIEW          SCOTT            ITEM            TABLE

. . . . . .
```

Ştergerea subprogramelor stocate

DROP PROCEDURE *procedura* **;**

DROP FUNCTION *functie* **;**

E. 4. Subprograme stocate. Pachete.

***Pachet* (*package*):**

- colecţie încapsulată de obiecte de programare (variabile, constante, proceduri, funcţii, cursoare, excepţii), care pot fi apelate din alte programe

- este considerat obiect al schemei şi este stocat în baza de date

Pachete

Avantaje : - independente de aplicaţie

- contribuie la modularizare
- uşor accesibile
- reutilizabile
- variabilele şi cursoarele persistă pe durata sesiunii şi pot fi partajate
- contribuie la securitatea bazei de date

Pachete

Componente :

- **antetul** sau **specificaţia** (declararea elementelor accesibile din afara pachetului)

- **corpul** (definirea completă a elementelor, inclusiv a celor locale)

Sunt stocate separat.
Obiectele dependente sunt legate numai de specificaţie.
Modificarea corpului nu afectează obiectele dependente.

Crearea pachetelor

```
CREATE [OR REPLACE] PACKAGE pachet IS
    <declaraţii publice şi antete de subprograme>
END;
```

```
CREATE [OR REPLACE] PACKAGE BODY pachet IS
    <declaraţii private şi definiri de subprograme>
END;
```

Crearea pachetelor

Exemplu

```
CREATE PACKAGE salarii  IS

 procedure prima ( p_deptno in number, p_procent number);
 function impozit( p_emp number )  RETURN number ;

END salarii;
```

Crearea pachetelor

Exemplu

```
CREATE PACKAGE BODY salarii IS

 procedure prima ( p_deptno in number, p_procent number) IS
   begin
      . . . . . .
   end prima;

 function impozit ( p_emp number )  RETURN number IS
   v_grade salgrade.grade%TYPE;
   v_sal   emp.sal%TYPE;
   begin
      . . . . .
   end impozit;

END salarii ;
```

Utilizarea elementelor unui pachet

pachet . nume_obiect **[(** *<lista parametri actuali>* **)]**

Exemplu

```
salarii.prima(20,15);
```

Obţinerea de informaţii asupra pachetelor

Dicţionarul de date

disponibile pentru utilizator :

- **USER_OBJECTS**
- **USER_SOURCE**
- **USER_DEPENDENCIES**
- **ALL_SOURCE**

. . . . . .

Obţinerea de informaţii asupra pachetelor

Exemplu

```
SQL> select * from user_source where name='SALARII';

OWNER  NAME     TYPE         LINE TEXT
------ -------  ------------ ---- --------------------------------
SCOTT  SALARII  PACKAGE         1 package salarii is
SCOTT  SALARII  PACKAGE         2 procedure prima ( p_deptno in number, p_procent in number);
SCOTT  SALARII  PACKAGE         3 function impozit ( p_emp emp.empno%TYPE ) RETURN number;
SCOTT  SALARII  PACKAGE         4 end salarii;
SCOTT  SALARII  PACKAGE BODY    1 PACKAGE BODY salarii IS
SCOTT  SALARII  PACKAGE BODY    2 procedure prima ( p_deptno in number, p_procent number) IS
SCOTT  SALARII  PACKAGE BODY    3 begin
. . . . . .
SCOTT  SALARII  PACKAGE BODY   28 END salarii ;

32 rows selected.
```

Obţinerea de informaţii asupra pachetelor

Exemplu

```
SQL> select * from USER_OBJECTS where OBJECT_NAME='SALARII';

OBJECT_NAME OBJECT_ID OBJECT_TYPE     CREATED   LAST_DDL   TIMESTAMP           STATUS
----------- --------- ----------- ----------- --------- ------------------- -----------

SALARII         11475 PACKAGE        15-APR-03 15-APR-03  2003-04-15:16:42:51  INVALID
SALARII         11476 PACKAGE BODY   15-APR-03 15-APR-03  2003-04-15:16:42:51  INVALID
```

Ştergerea pachetelor

DROP PACKAGE BODY *pachet* **;**

DROP PACKAGE *pachet* **;**

Pachete predefinite

disponibile pentru utilizator :

- **DBMS_OUTPUT (transmitere de texte spre afişare în mediu)**
- **DBMS_SQL (execuţie de comenzi SQL create dinamic)**
- **DBMS_JOB (planificare de lucrări)**
- **DBMS_PIPE (comunicare între procese)**
- **DBMS_ALERT (avertizare la apariţia unor evenimente)**
- **DBMS_UTILITY (execuţia de operaţii utilitare)**
- **DBMS_LOB (manipulare de obiecte LOB)**
- **UTL_FILE (operaţii pe fişiere text)**