Csomagok

Csomag

- Adatbázis-objektum
- Programozási eszközök gyűjteménye
- Két részből áll
  - specifikáció
  - törzs (opcionális)

Csomagspecifikáció

```
CREATE [OR REPLACE] PACKAGE csomagnév
[AUTHID {DEFINER|CURRENT_USER}] {IS|AS}
  {típusdefiníciók |
  nevesített_konstans_deklarációk |
  változódeklarációk | kivételdeklarációk |
  kurzorspecifikációk |
  alprogramspecifikációk | pragmák}
END [csomagnév];
```

- Nincs futtatható kód!
- Nyilvános deklarációk, a csomag nevével minősítünk
- Az alprogramnevek túlterhelhetők

Csomagtörzs

```
CREATE [OR REPLACE] PACKAGE BODY csomagnév
{IS|AS}
  deklarációk
[BEGIN
  végrehajtható_utasítások]
END [csomagnév];
```

- Ha a specifikációban van kurzor- vagy alprogramspecifikáció, akkor kötelező megadni!
- Privát deklarációk, csak a csomagtörzsből látszanak
- A BEGIN után inicializáló utasítások vannak
- A törzs függ a specifikációtól!

```
CREATE OR REPLACE PACKAGE konyvtar_csomag AS
  ⋮
  CURSOR cur_lejart_kolcsonzesek(
    p_Datum DATE DEFAULT SYSDATE
  ) RETURN t_lejart_rec;
  ⋮
END;
```

```
CREATE OR REPLACE PACKAGE BODY konyvtar_csomag AS
  ⋮
  CURSOR cur_lejart_kolcsonzesek(p_Datum DATE
   DEFAULT SYSDATE ) RETURN t_lejart_rec IS
    SELECT kolcsonzo, konyv, datum, hosszabbitva,
           megjegyzes, napok
     FROM (SELECT kolcsonzo, konyv, datum,
           hosszabbitva, megjegyzes,
           TRUNC(p_Datum, 'DD') - TRUNC(datum, 'DD')
                  - 30*(hosszabbitva+1) AS napok
           FROM kolcsonzes)
        WHERE napok > 0
  ;
  ⋮
END;
```

Működés

- Fordítás: p-kódra

```
ALTER PACKAGE csomagnév COMPILE [DEBUG]
  {PACKAGE|SPECIFICATION|BODY};
DROP PACKAGE [BODY] csomagnév;
```

Példányosítás és inicializálás

- Amikor egy munkamenet egy csomagbeli elemre hivatkozik, Oracle példányosítja a csomagot ahhoz a munkamenethez. Minden olyan munkamenet, amely csomagot hivatkozik, rendelkezik a csomaghoz egy saját példánnyal.
- Amikor az Oracle példányosít egy csomagot, akkor inicializálja is. Az inicializáció a következőket jelenti:
  - A publikus konstansok értéket kapnak
  - A publikus változók megkapják azt az értéket, amely a deklarációban van meghatározva
  - A csomag törzsének az inicializációs része lefut

Csomag állapot

- A csomagban deklarált változók, konstansok és kurzorok értéke
- Állapottal rendelkező <-> állapot nélküli csomag
- A csomag állapota egy munkamenet végéig megmarad, kivéve:
  - SERIALLY_REUSABLE használata esetén
  - A csomagtörzs újrafordításakor
  - Ha a munkamenet által használt valamely csomag érvénytelenné válik vagy újra lesz érvényesítve.

Csomagkészítési irányelvek

- Általános célú csomagokat készítsünk
- Csak a nyilvánosságnak szánt eszközöket vegyük fel a specifikációba (a csomag nem absztrakt adattípus!)
- A munkameneteken vagy tranzakciókon átívelő adatelemeket helyezzük a csomagtörzs deklarációs részébe
- A specifikáció változása magával vonja minden rá hivatkozó alprogram érvénytelenítését

```
CREATE OR REPLACE PACKAGE emp_actions AS  -- spec
   TYPE EmpRecTyp IS RECORD (emp_id INT, salary REAL);
   CURSOR desc_salary RETURN EmpRecTyp;
   PROCEDURE hire_employee (
      ename  VARCHAR2,
      job    VARCHAR2,
      mgr    NUMBER,
      sal    NUMBER,
      comm   NUMBER,
      deptno NUMBER);
   PROCEDURE fire_employee (emp_id NUMBER);
END emp_actions;
```

```
CREATE OR REPLACE PACKAGE BODY emp_actions AS  -- body
 CURSOR desc_salary RETURN EmpRecTyp IS
      SELECT empno, sal FROM emp ORDER BY sal DESC;
 PROCEDURE hire_employee (ename  VARCHAR2,
      job    VARCHAR2, mgr    NUMBER,
      sal    NUMBER, comm   NUMBER,
      deptno NUMBER) IS
 BEGIN
  INSERT INTO emp VALUES (empno_seq.NEXTVAL, ename,
  job, mgr, SYSDATE, sal, comm, deptno);
 END hire_employee;
PROCEDURE fire_employee (emp_id NUMBER) IS
   BEGIN
      DELETE FROM emp WHERE empno = emp_id;
   END fire_employee;
END emp_actions;
```

```
CREATE OR REPLACE PACKAGE pkg IS
  n NUMBER := 5;
END pkg;
/
CREATE OR REPLACE PACKAGE sr_pkg IS
 PRAGMA SERIALLY_REUSABLE;
 n NUMBER := 5;
END sr_pkg;
/
BEGIN
pkg.n := 10;
sr_pkg.n := 10;
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE('pkg.n: ' || pkg.n);
DBMS_OUTPUT.PUT_LINE('sr_pkg.n: ' || sr_pkg.n);
END;
```

Standard csomag

- PL/SQL nyelvi környezetét határozza meg.
- Típusok, kivételek, alprogramok

Példa

Package.sql

SERIALLY_REUSABLE

- If a package is not SERIALLY_REUSABLE, the package state can persist for the life of a session.
- If a package is SERIALLY_REUSABLE, the package state persists only for the life of a server call. If a subsequent server call references the package, changes made to the package state in previous server calls are invisible.

```
-- Create bodiless SERIALLY_REUSABLE package:
CREATE OR REPLACE PACKAGE bodiless_pkg IS
  PRAGMA SERIALLY_REUSABLE;
  n NUMBER := 5;
END;
/
```

```
-- Create SERIALLY_REUSABLE package with
   specification and body:
CREATE OR REPLACE PACKAGE pkg IS
   PRAGMA SERIALLY_REUSABLE;
   n NUMBER := 5;
END;
/

CREATE OR REPLACE PACKAGE BODY pkg IS
   PRAGMA SERIALLY_REUSABLE;
BEGIN
   n := 5;
END;
/
```

```
CREATE OR REPLACE PACKAGE pkg IS
  n NUMBER := 5;
END pkg; /
CREATE OR REPLACE PACKAGE sr_pkg IS
  PRAGMA SERIALLY_REUSABLE;
  n NUMBER := 5;
END sr_pkg; /

BEGIN
  pkg.n := 10; sr_pkg.n := 10;
END; /

BEGIN
  DBMS_OUTPUT.PUT_LINE('pkg.n: ' || pkg.n);
  DBMS_OUTPUT.PUT_LINE('sr_pkg.n: ' || sr_pkg.n);
END; /

Result:
pkg.n: 10
sr_pkg.n: 5
```

```
DROP TABLE people;
CREATE TABLE people (name VARCHAR2(20));

INSERT INTO people (name) VALUES ('John Smith');
INSERT INTO people (name) VALUES ('Mary Jones');
INSERT INTO people (name) VALUES ('Joe Brown');
INSERT INTO people (name) VALUES ('Jane White');

CREATE OR REPLACE PACKAGE sr_pkg IS
  PRAGMA SERIALLY_REUSABLE;
  CURSOR c IS SELECT name FROM people;
END sr_pkg;
/
```

…

```
…
CREATE OR REPLACE PROCEDURE fetch_from_cursor IS
  name_  VARCHAR2(200);
BEGIN
IF sr_pkg.c%ISOPEN
 THEN DBMS_OUTPUT.PUT_LINE('Cursor is open.');
 ELSE DBMS_OUTPUT.PUT_LINE('Cursor is closed;
                            opening now.');
    OPEN sr_pkg.c;
END IF;
  FETCH sr_pkg.c INTO name_;
  DBMS_OUTPUT.PUT_LINE('Fetched: ' || name_);

  FETCH sr_pkg.c INTO name;
   DBMS_OUTPUT.PUT_LINE('Fetched: ' || name_);
  END fetch_from_cursor;
/
```

```
…
First call to server:
BEGIN
  fetch_from_cursor;
  fetch_from_cursor;
END;
/
Result:
Cursor is closed; opening now.
Fetched: John Smith
Fetched: Mary Jones
Cursor is open.
Fetched: Joe Brown
Fetched: Jane White
…
```

…

```
New call to server:
BEGIN
  fetch_from_cursor;
  fetch_from_cursor;
END;
/
Result:
Cursor is closed; opening now.
Fetched: John Smith
Fetched: Mary Jones
Cursor is open.
Fetched: Joe Brown
Fetched: Jane White
```