

# Sistemi baze podataka

---

PL/SQL – Paketi

Paketi

---

# Paketi u PL/SQL-u

- Definicija paketa
  - Paket - kolekcija PL/SQL deklaracija
    - tipova, promenljivih, konstanti
    - kursorskih područja i izuzetaka
    - procedura i funkcija

# Struktura paketa

- Javni (vidljivi) deo paketa
  - **Specifikacija paketa**
    - sadrži PL/SQL "**javne**" (**public**) deklaracije - koje su dostupne za upotrebu (vidljive) i izvan paketa i unutar paketa

# Struktura paketa

- Privatni (skriveni) deo paketa
  - **Telo paketa**
    - sadrži PL/SQL "**privatne**" (**private**) deklaracije - koje su dostupne za upotrebu (vidljive) samo unutar paketa
    - sadrži kompletnu specifikaciju (implementaciju, razradu) javnih deklaracija procedura i funkcija
    - sadrži kompletnu specifikaciju (implementaciju, razradu) privatnih procedura i funkcija
  - **Inicijalizacioni blok paketa**
    - sadrži imperativne PL/SQL naredbe, koje se jednokratno izvršavaju, pri prvom referenciranju paketa u sesiji

# Vrste paketa

- Serverski paket
  - paket, kreiran na nivou DBMS i memorisan u rečniku podataka DBMS
  - egzistira u rečniku podataka u dva oblika:
    - izvornom (source kod)
    - prekompajliranom (P-kod – izvršni kod, interpretabilan od strane DBMS i PL/SQL Engine-a)
- Klijentski paket
  - paket, deklarisan u okviru nekog alata iz Oracle Developer Suite
  - nalazi se i izvršava na srednjem sloju (aplikativnom serveru)

# Oblikovanje specifikacije paketa

```
CREATE [OR REPLACE] PACKAGE [schema.]package_name  
IS|AS  
    deklaracije javnih PL/SQL elemenata ...  
END [package_name];
```

```
ALTER PACKAGE [schema.]package_name COMPILE;
```

```
DROP PACKAGE [schema.]package_name;
```

# Oblikovanje specifikacije paketa

- U okviru specifikacije paketa moguće je deklarirati bilo koji PL/SQL element (tip, promenljiva, konstanta, kursor, izuzetak, procedura, funkcija)
- Tipovi, promenljive, konstante, kursori i izuzeci se deklariraju na uobičajen način

# Oblikovanje specifikacije paketa

- Procedure i funkcije se deklariraju samo navođenjem zaglavlja (header-a), prema sintaksi:

```
PROCEDURE procedure_name  
  [(parameter1 [IN | OUT | IN OUT] datatype1 [DEFAULT def_value],  
   parameter2 [IN | OUT | IN OUT] datatype2 [DEFAULT def_value],  
   ... ) ];
```

```
FUNCTION function_name  
  [(parameter1 [IN | OUT | IN OUT] datatype1 [DEFAULT def_value],  
   parameter2 [IN | OUT | IN OUT] datatype2 [DEFAULT def_value],  
   ... )]  
RETURN ret_datatype;
```

# Napomene u vezi kreiranja specifikacije paketa

- Svaka procedura ili funkcija, čije se zaglavlje pojavljuje u specifikaciji paketa mora biti kompletno specificirana (razrađena) u telu paketa, pod istim nazivom kao što je i zaglavlje paketa
  - u tom slučaju, obavezno je kreiranje tela paketa
- Specifikacija paketa koji nema u sebi deklarisanе funkcije ili procedure, ne zahteva kreiranje tela paketa
  - u tom slučaju, kreiranje tela paketa nije obavezno
- Sve promenljive, deklarisanе u specifikaciji paketa, po default-u, biće inicijalizovane na NULL vrednost
- Sve promenljive, deklarisanе u specifikaciji paketa, memorisaće vrednosti koje su jedinstvene na nivou jedne sesije korisnika
  - bez obzira koliko različitih i kojih PL/SQL programa preuzima ili ažurira vrednosti tih promenljivih
- Svako kompajliranje (izmena) specifikacije paketa, zahteva i ponovno kompajliranje tela paketa.

## Primer za kreiranje specifikacije paketa

```
CREATE OR REPLACE PACKAGE Var_Methods IS
  TYPE T_Tab IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  PROCEDURE Set_Val(P_Key IN BINARY_INTEGER, P_Val IN NUMBER);
  PROCEDURE Inc_Val(P_Key IN BINARY_INTEGER,
                   P_Stp IN NUMBER DEFAULT 1);
  PROCEDURE Rem_Key(P_Key IN BINARY_INTEGER);
  FUNCTION Get_Val(P_Key IN BINARY_INTEGER) RETURN NUMBER;
END Var_Methods;
```

## Oblikovanje tela paketa

```
CREATE [OR REPLACE] PACKAGE BODY [schema.]package_name  
IS|AS
```

deklaracije privatnih PL/SQL elemenata ...  
implementacije

```
[BEGIN
```

proceduralni deo – inicijalizacija promenljivih

```
]
```

```
END [package_name];
```

```
ALTER PACKAGE BODY [schema.]package_name COMPILE;
```

```
DROP PACKAGE BODY [schema.]package_name;
```

# Oblikovanje tela paketa

- U okviru tela paketa moguće je deklarirati bilo koji PL/SQL element (tip, promenljiva, konstanta, kursor, izuzetak, procedura, funkcija)
- Tipovi, promenljive, konstante, kursori i izuzeci se deklariraju na uobičajen način
- Procedure i funkcije, bez obzira da li su javne ili privatne, kompletno se specificiraju, na isti način kao i lokalne procedure i funkcije – na uobičajen način
- Proceduralni deo koda koji se nalazi u BEGIN – END delu tela paketa izvršava se samo jednom, na nivou sesije
  - prvi put, kada se referencira bilo koji element paketa i/ili kada se sadržaj paketa učitava u radnu memoriju DBMS-a

# Napomene u vezi kreiranje tela paketa

- Svaka procedura ili funkcija, čije se zaglavlje pojavljuje u specifikaciji paketa mora biti kompletno specificirana (razrađena) u telu paketa, pod istim nazivom kao što je i zaglavlje paketa
  - zaglavlje javne procedure ili funkcije, koja se specificira u telu paketa, mora u potpunosti da odgovara zaglavlju, deklarisanom u specifikaciji paketa
- Sve promenljive, deklarisanе u telu paketa, po default-u, biće inicijalizovane na NULL vrednost
- Sve promenljive, deklarisanе u telu paketa, memorisaće vrednosti koje su jedinstvene na nivou jedne sesije korisnika
  - bez obzira koliko različitih i kojih PL/SQL programa preuzima ili ažurira vrednosti tih promenljivih
- Kompajliranje (izmena) tela paketa, ne mora da zahteva ponovno kompajliranje specifikacije paketa.

## Primer kreiranja tela paketa

```
CREATE OR REPLACE PACKAGE BODY Var_Methods IS
  TabValue T_Tab;
  PROCEDURE Set_Val(P_Key IN BINARY_INTEGER, P_Val IN NUMBER) IS
  BEGIN
    TabValue(P_Key) := P_Val;
  END Set_Val;

  PROCEDURE Inc_Val(P_Key IN BINARY_INTEGER, P_Stp IN NUMBER
  DEFAULT 1) IS
  BEGIN
    IF TabValue.EXISTS(P_Key) AND TabValue(P_Key) IS NOT NULL THEN
      TabValue(P_Key) := TabValue(P_Key) + P_Stp;
    END IF;
  END Inc_Val;
  PROCEDURE Rem_Key(P_Key IN BINARY_INTEGER) IS
  BEGIN -- Nastavak na sledećem slajdu
```

## Primer kreiranja tela paketa

```
        IF TabValue.EXISTS(P_Key) THEN
            TabValue.DELETE(P_Key);
        END IF;
    END Rem_Key;
    FUNCTION Get_Val(P_Key IN BINARY_INTEGER) RETURN NUMBER IS
    BEGIN
        IF TabValue.EXISTS(P_Key) THEN
            RETURN TabValue(P_Key);
        ELSE
            RETURN -1;
        END IF;
    END Get_Val;
BEGIN
    TabValue(1) := 0;
END Var_Methods;
```

# Referenciranje elemenata PL/SQL paketa

- Paket, sam po sebi, nije izvršiva PL/SQL konstrukcija
  - To je koncept koji obezbeđuje bolje organizovanje programskog koda
- Elementi paketa mogu biti referencirani
  - iz PL/SQL konstrukcije koja je unutar paketa
  - iz PL/SQL konstrukcije koja je izvan paketa

# Referenciranje elemenata PL/SQL paketa

- **Referenciranje unutar paketa**

- Na uobičajeni način, navođenjem naziva referenciranog elementa, saglasno sintaksnim pravilima jezika PL/SQL

- **Referenciranje izvan paketa**

- Navođenjem naziva paketa, kao prefiksa, a zatim na uobičajeni način, saglasno sintaksnim pravilima jezika PL/SQL

# Referenciranje elemenata PL/SQL paketa

[schema.]Naziv\_paketa.Naziv\_tipa

[schema.]Naziv\_paketa.Naziv\_promenljive

[schema.]Naziv\_paketa.Naziv\_izuzetka

[schema.]Naziv\_paketa.Naziv\_kursora

[schema.]Naziv\_paketa.Naziv\_procedure [(lista\_stvarnih\_parametara)]

[schema.]Naziv\_paketa.Naziv\_funkcije [(lista\_stvarnih\_parametara)]

# Primer referenciranja elemenata paketa, od strane PL/SQL bloka izvan paketa

**DECLARE**

V\_b NUMBER;

**BEGIN**

Var\_Methods.Set\_Val(1, 0);

Var\_Methods.Inc\_Val(1, 2);

Var\_Methods.Inc\_Val(2, 1);

Var\_Methods.Rem\_Key(1);

Var\_Methods.Rem\_Key(2);

V\_b := Var\_Methods.Get\_Val(1);

**END;**

## Primer referenciranja elemenata paketa, od strane PL/SQL konstrukcije unutar i izvan paketa

```
CREATE OR REPLACE PACKAGE Ref_Intro IS
    TYPE T_Proj IS TABLE OF Projekat%ROWTYPE INDEX BY BINARY_INTEGER;
    FUNCTION SEL_Projekat RETURN T_Proj;
END Ref_Intro;
```

## Primer referenciranja elemenata paketa, od strane PL/SQL konstrukcije unutar i izvan paketa

```
CREATE OR REPLACE PACKAGE BODY Ref_Intro IS
  CURSOR C_P IS
    SELECT *
    FROM Projekat
    ORDER BY Nap;
  FUNCTION SEL_Projekat RETURN T_Proj IS
    i BINARY_INTEGER := 0;
    L_ProjTab T_Proj;
  BEGIN
    FOR C_Rec IN C_P LOOP
      i := i + 1;
      L_ProjTab(i) := C_rec;
    END LOOP;
    RETURN L_ProjTab;
  END SEL_Projekat;
END Ref_Intro;
```

# Primer referenciranja elemenata paketa, od strane PL/SQL konstrukcije unutar i izvan paketa

**DECLARE**

ProjTab Ref\_Intro.T\_Proj;

**BEGIN**

ProjTab := Ref\_Intro.SEL\_Projekat;

**END;**

# Preklapanje definicija procedura ili funkcija (overloading)

- Identifikacija jedinstvenosti procedure ili funkcije unutar paketa:
  - Naziv funkcije / procedure
  - Broj, tipovi i vrste deklariranih formalnih parametara
  - Tip povratnog podatka (samo za funkcije)
- Funkcije / procedure koje su iste po nazivu, ali se razlikuju po deklariranim listama parametara (i/ili tipu povratne vrednosti), smatraju se različitim funkcijama

# Preklapanje definicija procedura ili funkcija (overloading)

- Preklapanje funkcija ili procedura
  - deklarisanje funkcija / procedura sa istim nazivom, ali različitim listama formalnih parametara (ili tipom povratne vrednosti)
- Zahtev: potrebno je, deklaracijom i svim mogućim pozivima, obezbediti nedvosmislenost referenciranja procedure / funkcije
- **NAPOMENA:** moguće je obezbediti preklapanje samo procedura / funkcija koje pripadaju nekom (istom) paketu
  - Nije moguće obezbediti preklapanje samostalnih serverskih procedura ili funkcija
- Postoje primeri preklapanja ugrađenih SQL funkcija

## Primer referenciranja elemenata paketa, od strane PL/SQL konstrukcije unutar i izvan paketa

```
FUNCTION TO_CHAR (p1 DATE) RETURN VARCHAR2;
```

```
FUNCTION TO_CHAR (p1 DATE, P2 VARCHAR2) RETURN VARCHAR2;
```

```
FUNCTION TO_CHAR (p2 NUMBER) RETURN VARCHAR2;
```

```
FUNCTION TO_CHAR (p1 NUMBER, P2 VARCHAR2) RETURN VARCHAR2;
```

# Paketi u PL/SQL-u zadaci

- Formirati paket procedura i funkcija za rad s tabelom Radproj. Treba obezbediti sledeću funkcionalnost, putem poziva odgovarajućih funkcija, ili procedura:
  - selektovanje jedne torke iz tabele, saglasno zadatoj vrednosti ključa Spr+Mbr,
  - dodavanje jedne nove torke u tabelu, koja se prenosi kao parametar,
  - brisanje torke iz tabele, saglasno zadatoj vrednosti ključa Spr+Mbr i
  - modifikacija torke u tabeli, saglasno zadatoj vrednosti ključa Spr+Mbr i zadatoj vrednosti za obeležje koje se modifikuje.
- Korisnik treba da ima obezbeđenu indikaciju uspešnosti svake od navedenih operacija.

# Rešenje

```
CREATE OR REPLACE PACKAGE RadProj_Package IS
  FUNCTION Sel_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
    RADPROJ.MBR%TYPE) RETURN RADPROJ%ROWTYPE;
  FUNCTION Ins_RP(P_TORKA IN RADPROJ%ROWTYPE) RETURN BOOLEAN;
  FUNCTION Del_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
    RADPROJ.MBR%TYPE) RETURN BOOLEAN;
  FUNCTION Upd_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
    RADPROJ.MBR%TYPE) RETURN NUMBER;
END RadProj_Package;
```

# Rešenje

```
CREATE or REPLACE  
PACKAGE BODY RadProj_Package IS
```

```
    FUNCTION Sel_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN  
                   RADPROJ.MBR%TYPE)  
    RETURN RADPROJ%ROWTYPE  
    IS  
        REC RADPROJ%ROWTYPE;  
    BEGIN  
        SELECT * INTO REC FROM RADPROJ WHERE SPR=P_SPR AND  
            MBR=P_MBR;  
    RETURN REC;  
    END Sel_RP;
```

```
    PROCEDURE Ins_RP(P_TORKA IN RADPROJ%ROWTYPE)  
    IS  
    BEGIN -- Nastavak na sledećem slajdu
```

# Rešenje

```
INSERT INTO RADPROJ VALUES
(P_TORKA.SPR,P_TORKA.MBR,P_TORKA.BRC);
IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Uspesno uneta torka');
ELSE
    DBMS_OUTPUT.PUT_LINE('Nije uneta torka');
END IF;
END Ins_RP;
```

# Rešenje

```
PROCEDURE Del_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
RADPROJ.MBR%TYPE)
IS
BEGIN
DELETE RADPROJ WHERE SPR=P_SPR AND MBR= P_MBR;
IF SQL%FOUND THEN
DBMS_OUTPUT.PUT_LINE('Uspesno obrisana toraka');
ELSE
DBMS_OUTPUT.PUT_LINE('Nije obrisana toraka');
END IF;
END Del_RP;
```

```
PROCEDURE Upd_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
RADPROJ.MBR%TYPE, P_BRC IN RADPROJ.BRC%TYPE)
IS
BEGIN -- Nastavak na sledećem slajdu
```

# Rešenje

```
UPDATE RADPROJ SET
SPR=P_SPR,
MBR=P_MBR,
BRC=P_BRC;
IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Uspesno modifikovana toraka');
ELSE
    DBMS_OUTPUT.PUT_LINE('Nije modifikovana toraka');
END IF;
END Upd_RP;

END RadProj_Package;
```

# Paketi u PL/SQL-u zadaci

- Formirati paket procedura i funkcija za rad s tabelom Radproj. Treba obezbediti sledeću funkcionalnost, putem poziva odgovarajućih funkcija, ili procedura:
  - selektovanje niza torki iz tabele, saglasno zadatom kriterijumu,
  - dodavanje niza novih torki u tabelu, koji se prenosi kao parametar,
  - brisanje niza torki iz tabele, saglasno nizu vrednosti ključa Spr+Mbr, koji se prenosi kao parametar,
  - modifikacija niza torki u tabeli, saglasno nizu vrednosti ključa Spr+Mbr i modifikovanih vrednosti, koji se prenose kao parametar.
- Korisnik treba da ima obezbeđenu indikaciju uspešnosti svake od navedenih operacija.

# Rešenje

**DECLARE**

NizTorki radproj\_package.T\_RadProj;

**BEGIN**

    NizTorki(1).spr:= 80;

    NizTorki(1).mbr:= 10;

    NizTorki(1).brc:= 333;

    NizTorki(2).spr:= 80;

    NizTorki(2).mbr:= 40;

    NizTorki(2).brc:= 777;

    radproj\_package.Dodati(NizTorki);

**END;**

# Rešenje

```
CREATE or REPLACE
PACKAGE RadProj_Package IS
  TYPE T_RadProj IS TABLE OF RadProj%ROWTYPE INDEX BY
  BINARY_INTEGER;
  TYPE T_MbrSpr IS RECORD(
    Spr RadProj.Spr%TYPE,
    Mbr RadProj.Mbr%TYPE);
  TYPE T_NizMbrSpr IS TABLE OF T_MbrSpr INDEX BY BINARY_INTEGER;
  PROCEDURE Dodati(Torke IN T_RadProj);
  PROCEDURE Obrisati(Torke IN T_NizMbrSpr);
  FUNCTION Sel_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
  RADPROJ.MBR%TYPE) RETURN RADPROJ%ROWTYPE;
  PROCEDURE Ins_RP(P_TORKA IN RADPROJ%ROWTYPE);
  PROCEDURE Del_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
  RADPROJ.MBR%TYPE);
  PROCEDURE Upd_RP(P_SPR IN RADPROJ.SPR%TYPE, P_MBR IN
  RADPROJ.MBR%TYPE, P_BRC IN RADPROJ.BRC%TYPE);
END RadProj_Package;
```

# Rešenje

```
PROCEDURE Obrisati(Torke IN T_NizMbrSpr)
  IS
  i integer;
  BEGIN
    i:=Torke.FIRST;
    WHILE i<=Torke.LAST LOOP
      DELETE RADPROJ WHERE MBR=Torke(i).MBR AND SPR=Torke(i).SPR;
      i := Torke.NEXT(i);
    END LOOP;
  END Obrisati;
```

# Rešenje

**DECLARE**

NizTorki radproj\_package.T\_NizMbrSpr;

**BEGIN**

NizTorki(1).spr:= 80;

NizTorki(1).mbr:= 10;

NizTorki(2).spr:= 80;

NizTorki(2).mbr:= 40;

radproj\_package.Obrisati(NizTorki);

**END;**

# Rešenje

- Napisati proceduru za unos novih podataka u tabelu *Projekat*. Procedura kao ulazne vrednosti treba da prima podatke o projektu (SPR, NAZP, NAR), i kolekciju radnika koja sadrži par (MBR, BRC). Procedura ima dve izlazne vrednosti: uspešnost unosa torke u tabelu *Projekat* i broj unetih torki u tabelu *Radproj*. Procedura treba da obezbedi:
  - Proveru da li projekat sa datom šifrom može biti unet. Ukoliko već postoji projekat sa datom šifrom izazvati izuzetak sa odgovarajućom porukom o grešci.
  - Proveru da li radnik sa MBR u prosleđenoj kolekciji postoji. Ukoliko ne postoji radnik sa prosleđenim MBR izazvati izuzetak sa odgovarajućom porukom o grešci.
  - Obezbediti da se u bazu podataka ili unose sve prethodno navedene torke ili se ne unosi ni jedna. Na osnovu uspešnosti unosa torki dodeliti vrednosti izlaznim parametrima.

# Kraj!

Hvala na pažnji!