Šesti čas
PL/SQL

Skraćenica za **P**rocedural **L**anguage extension to **SQL** (proširenje SQL-a). Služi za složeniju i precizniju obradu podataka.

Mogu se kreirati: neimenovani blokovi, procedure, funkcije i okidači.

ANONIMNI BLOKOVI
Anonymous PL/SQL block
Sve naredbe u PL/SQL-u se grupišu u blokove. Ako ih neimenujemo imamo kod za jednokratnu upotrebu.

PROCEDURE
Stored procedures
Niz akcija koje se izvršavaju nakon poziva procedure, nema povratnih vrednosti. Procedure mogu izvršiti SQL naredbe i manipulisati podacima u tabelama

FUNKCIJE
Stored functions
Predstavljaju procedure koje vraćaju neku vrednost.

OKIDAČI
Triggers
Za raliku od procedura i funkcija koje se neposredno pozivaju, okidači se prave tako da se automatski definišu nakon nekog događaja, definisanog u okidaču. Na primer, moguće je napisati okidač koji će biti pokrenut kada je pokrenuta neka od naredbi INSERT, UPDATE ili DELETE, nakon logovanja korisnika, pojave neke greške, pri startovanju određene baze, itd.
Razlikuje se od procedura i funkcija po tome što:

- Nemoguće je pozvati okidač unuar koda, okidače pokreće Oracle automatski kao odgovor na neku definisanu akciju ili događaj
- Okidači nemaju listu argumenata
- Specifikacija okidača sadrži druge podatke u odnosu na specifikaciju procedure.

ANONIMNI BLOKOVI

Struktura osnovnih blokova:

1. zaglavlje

```
<<naziv_bloka>>
DECLARE
```

2. deklaracije promenjivih (opciono)

naredbe između DECLARE i BEGIN
sadrži deklaracije promenjivih, konstanti, izuzetaka, procedura i funkcija koje će biti korišćene u izvršnom delu i delu za izuzetke

3. izvršni deo

počinje sa ključnom rečju BEGIN, a završava se ili sa END (ako nema dela za obradu izuzetaka) ili sa EXCEPTION (ako tog dela ima)

```
BEGIN
      Jedan ili više PL/SQL izraza
[deo za obradu izuzetaka]
END [naziv funkcije ili procedure];
```

4. deo za obradu izuzetaka (opciono)

niz naredbi koje se pokreću kada nastupi iziuzetak

```
EXCEPTION
      WHEN naziv_izuzetka
      THEN
            Akcije koje se obavljaju kad izuzetak nastupi
      WHEN naziv_izuzetka
      THEN
            Akcije koje se obavljaju kad izuzetak nastupi
      ...
```

Anonimni blokovi nemaju zaglavlje, pokreću se unutar SQL*plus-a ili unutar funkcija, procedura ili okidača.

Da bi informacije koje programi ispišu bile vidljive u SQL*plus-u treba uneti:

```
set serveroutput on
```

Primer koda anonimnog bloka:

```
DECLARE
      Num_a NUMBER :=6;
      Num_b NUMBER;
BEGIN
      Num_b:=0;
      Num_a:=Num_a / Num_b;
      dbms_output.put_line(' Vrednost promenjive Num_b'||Num_b);
EXCEPTION
      WHEN ZERO_DIVIDE
THEN
      dbms_output.put_line('Deljenje nulom nastupilo ');
      dbms_output.put_line('Vrednost promenjive Num_a: '||Num_a);
      dbms_output.put_line('Vrednost promenjive Num_b: '||Num_b);
END;
/
```

Primer bez deljenja nulom:

```
DECLARE
      Num_a NUMBER :=6;
      Num_b NUMBER;
BEGIN
      Num_b:=0;
      Num_a:=Num_a / 5;
      dbms_output.put_line('Vrednost promenjive Num_a: '||Num_a);
EXCEPTION
      WHEN ZERO_DIVIDE
THEN
      dbms_output.put_line('Deljenje nulom nastupilo ');
      dbms_output.put_line('Vrednost promenjive Num_a: '||Num_a);
      dbms_output.put_line('Vrednost promenjive Num_b: '||Num_b);
END;
/
```

Zadatak 1: Napisati neimenovani blok koji ispisuje poruku: POZDRAV SVIMA!

```
DECLARE
      poruka VARCHAR2(20);
BEGIN
      poruka:='Pozdrav svima!!';
      dbms_output.put_line(poruka);
END;
/
```

Zadatak 2. Napisati anonimni blok koji ispisuje najmanju platu radnika

```
DECLARE
      minimalna NUMBER(15);
BEGIN
      SELECT min(plata)
      INTO minimalna
      FROM radnik;
dbms_output.put_line('Najmanja plata iznosi: '|| minimalna ||'
dinara.');
END;
/
```

Zadatak 2. Napisati anonimni blok koji ispisuje ime i prezime najmlađeg radnika.

```
DECLARE
      ime varchar2(20);
      prezime varchar2(20);
BEGIN
      SELECT ime, prez
      INTO ime, prezime
      FROM radnik
      WHERE datr = (select max(datr) from radnik);
dbms_output.put_line('Najmladji je: '||ime||' '||prezime ||'!');
END;
/
```

Osnovni elementi PL/SQL-a:

Deklaracija promenjivih

```
naziv_promenjive tip_promenjive [[NOT NULL] :=default_vrednost];

broj NUMBER NOT NULL :=0;
```

Deklaracija konstanti

```
naziv_konstante tip_konstante CONSTANT := vrednost konstante;

pi NUMBER CONSTANT := 3.14;
```

Dodela vrednosti promenjivima

```
naziv_promenjive:=izraz;

broj:=7;
```

IF naredba

```
IF uslov_1 THEN
      akcije_1;
[ELSIF uslov_2 THEN
      akcije_2;]
      ...
[ELSE
```

```
      akcije_poslednje;]
END IF;

IF (plata > 55000) THEN
      plata:=plata + 10000;
ELSIF (plata = 55000) THEN
      plata:=plata + 9000;
ELSE
      plata:=plata + 8000;
END IF;
```

Petlje LOOP, WHILE i FOR

LOOP

```
<<naziv_petlje>>
LOOP
      izrazi;
      EXIT naziv_petlje [WHEN uslov_izlaska];
      izrazi;
END LOOP;
```

Zadatak 1. Ispisati prvih pet brojeva pomoću LOOP naredbe.

```
DECLARE
      broj NUMBER:=1;
BEGIN
      <<prvih_pet>>
      LOOP
            dbms_output.put_line(broj);
      EXIT prvih_pet WHEN (broj>=5);
      broj:=broj+1;
      END LOOP;
END;
/
```

WHILE

```
WHILE uslov_while_petlje
LOOP
      izrazi;
END LOOP;
```

Zadatak 2. Ispisati prvih pet brojeva pomoću WHILE naredbe.

```
DECLARE
      broj NUMBER:=1;
BEGIN
      WHILE broj<=5
      LOOP
            dbms_output.put_line(broj);
```

```
            broj:=broj+1;
      END LOOP;
END;
/
```

FOR

```
FOR brojac IN [REVERSE] donja_granica..gornja_granica
LOOP
      izrazi;
END LOOP;
```

Zadatak 3. Ispisati prvih pet brojeva u opadajućem rasporedu pomoću FOR naredbe.

```
BEGIN
      FOR brojac IN REVERSE 1..5
      LOOP
            dbms_output.put_line(brojac);
      END LOOP;
END;
/
```

PROCEDURE

Sintaksa za kreiranje procedure:

```
CREATE [OR REPLACE] PROCEDURE specifikacija_procedure IS
telo_procedure
```

Sintaksa specifikacije procedure:

```
naziv_procedure(deklaracija_prom1,
                deklaracije_prom2,
                ...
                deklaracija_promN)
```

Poziv procedure:

```
      naziv_procedure();
ili   naziv_procedure;
ili   naziv_procedure(izraz1, izraz2,...,izrazN)
```

Zadatak 1. Napisati standardnu HELLO WORLD proceduru.

```
CREATE PROCEDURE hello_world IS
      poruka VARCHAR2(20);
BEGIN
      poruka:= 'pozdrav korisniku';
      dbms_output.put_line(poruka);
```

```
END hello_world;
/
```

pokretanje procedure iz SQL*plus-a:

```
EXECUTE hello_world;
```

Moguće je pozvati proceduru i iz neimenovanog bloka sa:

```
BEGIN
     Hello_world;
END;
/
```

Zadatak 2. Napisati proceduru koja povećava neku vrednost za određeni procenat

```
CREATE OR REPLACE PROCEDURE povecanje(stara NUMBER,
                            procenat NUMBER,
                            nova OUT NUMBER)
IS
BEGIN
     nova := stara + stara * procenat / 100;
END povecanje;
/
```

pokretanje:

```
DECLARE
     stara_cena NUMBER:=1000;
     procenat NUMBER:=15;
     nova_cena NUMBER;
BEGIN
     dbms_output.put_line('Stara cena je: '||stara_cena);
     dbms_output.put_line('Procenat povecanja je: '|| procenat);
     povecanje(stara_cena, procenat, nova_cena);
     dbms_output.put_line('Nova cena iznosi: '||nova_cena);
END;
/
```

Zadatak 3. Napisati proceduru koja povećava plate svih radnika za određeni procenat

```
CREATE OR REPLACE PROCEDURE povecanje_plata(procenat NUMBER)
IS
BEGIN
     UPDATE RADNIK
     SET plata = plata * (1 + procenat/100);
END povecanje_plata;
/
```

pokretanje:

```
execute povecanje_plata(8);
```

Zadatak 4. Napisati proceduru za unos podataka o novom radniku u bazu

```
CREATE OR REPLACE PROCEDURE unos_radnika
                           (mbr IN radnik.mbr%TYPE,
                            ime IN radnik.ime%TYPE,
                            prezime IN radnik.prez%TYPE,
                            plata IN radnik.plata%TUPE)
IS
BEGIN
     INSERT INTO radnik (mbr, ime, prez, plata)
     VALUES (mbr, ime, prezime, plata);
END unos_radnika;
/
```

Pokretanje:

```
EXECUTE unos_radnika(199, 'Kontic', 'Marija', 57000);
```

Pozivanje procedura sa manjim brojem argumenata

Moguće je u deklaraciji procedure dodeliti nekim promenjivima podrazumevajuće vrednosti. U tom slučaju pri pozivanju imamo dve solucije:

1. moguće je postaviti sve promenjive koje imaju podrazumevajuće vrednosti na kraj liste, na primer sa N argumenata bez i M argumenata sa definisanom podrazumevajućom vrednošću:

```
CREATE PROCEDURE naziv_procedure(
                      argument1 tip1,
                      argument2 tip2,
                      ...
                      argumentN tipN,
                      argumentd1 tipd1:=default_vrednost1,
                      argumentd1 tipd2:=default_vrednost2,
                      ...
                      argumentdM tipdM:=default_vrednostM
                      )
IS
```

Tada proceduru možemo pozvati sa

```
naziv_procedure(arg1,arg2,...,argP);
```

gde je N <= P <= M, pa se argumenti dodeljuju redom argumentima procedure. Za argumente procedure koji ne dobiju vrednost uzimaju se njihove podrazumevajuće vrednosti.

2. ako argumenti proceduri sa podrazumevajućom vrednošću nisu postavljeni na kraj liste parametara tada se pri pozivu procedure moraju dodeliti vrednosti na sledeći način. Ako je zaglavlje procedure:

```
CREATE PROCEDURE naziv_procedure(
                      arg1 tip1,
                      arg2 tip2,
                      ...
                      argN tipN)
IS
```

Tada se poziv može vršiti sa:

```
naziv_procedure(arg1 => stvarni_param1,
                arg2 => stvarni_param2,
                ...
                argN => stvarni_paramN);
```

Gde će svim argumentima iz zaglavlja procedure koji nemaju definisane podrazumevajuće vrednosti biti dodeljen po jedan stvarni parametar

```
show errors
```

FUNKCIJE

Funkcije su slične procedurama (imaju specifikaciju funkcije i telo funkcije) sa tom razlikom da vraćaju povratnu vrednost.

Sintaksa funkcije:

```
CREATE [OR REPLACE] FUNCTION specifikacija_funkcije IS
telo_funkcije;
```

Specifikacija funkcije:

```
Naziv_funkcije(lista_argumenata) RETURN povratni_tip
```

Zadatak 1. Napisati funkciju koja vraća vrednost najveće plate radnika

```
CREATE OR REPLACE FUNCTION najveca_plata RETURN NUMBER
IS
     maximum NUMBER;
BEGIN
```

```
      SELECT max(plata)
      INTO maximum
      FROM radnik;
      RETURN maximum;
END najveca_plata;
/
```

pokretanje:

```
BEGIN
      dbms_output.put_line(najveca_plata);
END;
/
```

ili:

```
BEGIN
      dbms_output.put_line('Najveca  plata  je:  '||  najveca_plata
||' dinara!');
END;
/
```

Zadatak 2. Napisati funkciju koja vraća broj uvećan za 8 odsto, a zatim je iskoristiti u proceduri koja povećava platu svih radnika za taj procenat

```
CREATE  OR  REPLACE  FUNCTION  pov_za_procenat(cifra  NUMBER)  RETURN
NUMBER
IS
BEGIN
   RETURN cifra * 1.08;
END pov_za_procenat;
/

CREATE OR REPLACE PROCEDURE povisica
IS
BEGIN
      UPDATE radnik
      SET plata = pov_za_procenat(plata);
END povisica;
/
```

Zadatak 3. Napisati funkciju koja za vrednost argumenta vraća vrednost poreza (od 18 posto) i iskoristiti tu funkciju za listanje imena, prezimena, plata svih radnika i poreza koje firma plaća na osnovu njihovih plata.

```
CREATE OR REPLACE FUNCTION porez(procenat NUMBER) RETURN NUMBER
IS
BEGIN
   RETURN procenat * 0.08;
```

```
END porez;
/

SELECT ime, prez, plata, porez(plata) porez
FROM radnik
```

Listanje postojećih funkcija i procedura

```
SELECT object_name
FROM user_objects
WHERE object_type = 'FUNCTION';
```

ili

```
SELECT object_name
FROM user_objects
WHERE object_type = 'FUNCTION';
```

Listanje koda

```
SELECT text
FROM user_source
WHERE name = 'POREZ'
/
```

OKIDAČI

Okidači su PL/SQL procedure koje se pokreću automatski kada se desi neki ranije definisani događaj.

Sintaksa okidača

```
CREATE OR REPLACE TRIGGER naziv_okidaca
                          vreme_okidanja
                          dogadjaj
      ON naziv_tabele
      [FOR EACH ROW]
      [WHEN ogranicenje_okidaca]
      [DECLARE
            deklaracije_promenjivih]
      BEGIN
            Izrazi
      [EXCEPTION
            WHEN naziv_izuzetka
            THEN ...]
END naziv_okidaca;
```

da li će se pokrenuti pre ili nakon što se događaj pojavi:

```
vreme_okidanja: BEFORE ili AFTER
```

vrste događaja:

```
INSERT, UPDATE [OF kolona] ili DELETE
```

da li želimo da okine za svaki red ili jednom za po događaju:

```
FOR EACH ROW
```

Moguće je kombinovati više događaja:

```
DELETE OR INSERT ostatak_izraza
```

Kad se koristi UPDATE moguće je definisati listu kolona

```
UPDATE OF kolona1, kolona2,...
```

Umesto tela trigera moguće je pozvati proceduru sa:

```
CALL ime_procedure;
```

Zadatak 1. Napisati okidač koji sprečava unos plate van nekog opsega

```
CREATE OR REPLACE TRIGGER provera_plate
BEFORE INSERT OR UPDATE OF plata ON radnik
FOR EACH ROW
BEGIN
     IF    :NEW.plata < 15000 OR
           :NEW.plata > 150000 THEN
     RAISE_APPLICATION_ERROR (-20000, 'plata van opsega!');
     END IF;
END;
/
```

upit:

```
UPDATE radnik
SET plata = 200000
WHERE mbr=101;
```

Javlja grešku:

```
UPDATE radnik
       *
ERROR at line 1:
ORA-20000: plata van opsega!
ORA-06512: at "SUBP99.PROVERA_PLATE", line 4
ORA-04088:     error     during     execution     of     trigger
'SUBP99.PROVERA_PLATE'
```

Pravila za OLD i NEW kvalifikatore:

- Koriste se samo uz FOR EACH ROW
- Potreban je prefix : u svakoj SQL ili PL/SQL komandi
- Ne koristi se prefiks : samo kada se OLD i NEW referenciraju u WHEN delu PL/SQL koda

Zadatak 2. Napisati okidač koji sprečava unos plate koja je manja od postojeće:

```
CREATE OR REPLACE TRIGGER provera_plate
BEFORE INSERT OR UPDATE OF plata ON radnik
FOR EACH ROW
WHEN (NEW.plata < OLD.plata)
BEGIN
     RAISE_APPLICATION_ERROR   (-20000,   'plata   ne   sme   biti
umanjena!');
END;
/
```

ili:

```
CREATE OR REPLACE TRIGGER provera_plate
BEFORE INSERT OR UPDATE OF plata ON radnik
FOR EACH ROW
BEGIN
     IF    :NEW.plata < :OLD.plata THEN
           RAISE_APPLICATION_ERROR (-20000, ' plata ne sme biti
umanjena');
     END IF;
END;
/
```

Zadatak 3. Napisati okidač koji sprečava brisanje proivoljnog radnika firme

```
CREATE OR REPLACE TRIGGER nema_brisanja
BEFORE DELETE ON radnik
FOR EACH ROW
BEGIN
   RAISE_APPLICATION_ERROR (-20005, 'Ne sme se brisati radnik!');
END nema_brisanja;
/
```

Zadatak 4. Napisati okidač koji sprečava brisanje direktora firme (mbr = 100)

```
CREATE OR REPLACE TRIGGER nema_brisanja_direktora
BEFORE DELETE ON radnik
FOR EACH ROW
WHEN (OLD.mbr = 100)
BEGIN
   RAISE_APPLICATION_ERROR   (-20005,   'Ne   sme   se   brisati
Direktor!');
END nema_brisanja_direktora;
/
```

Zadatak 5. Napisati triger koji briše podatke iz tabele radproj ako dođe do brisanja podataka iz tabele radnik.

```
CREATE OR REPLACE TRIGGER brisanje_iz_radproj
AFTER DELETE ON radnik
FOR EACH ROW
BEGIN
    DELETE FROM radproj WHERE mbr = :OLD.mbr;
END brisanje_iz_radproj;
```

Zadatak 6. modifikovati triger iz prethodnog zadatka tako da u slučaju brisanja radnika dođe do brisanja iz radproj a ako dođe do novog unosa radnika ubacuje novu kolonu u tabelu radproj sa sifrom radnika i sifrom projekta 200.

```
CREATE OR REPLACE TRIGGER brisanje_iz_radproj
AFTER DELETE OR INSERT ON radnik
FOR EACH ROW
BEGIN
     IF DELETING THEN
          DELETE FROM radproj WHERE mbr = :OLD.mbr;
     ELSE
          INSERT INTO radproj (mbr, sifp)
          VALUES (:NEW.mbr, 200);
     END IF;
END brisanje_iz_radproj;
```

IZUZECI

Definisani izuzeci	Opis
DUP_VAL_ON_INDEX	Pokušaj da se unese dupla vrednost u kolonu gde se zahteva jedinstvenost tog obeležja
NO_DATA_FOUND	Neuspeli pokušaj naredbe SELECT INTO gde naredba SELECT ne vraća ništa
TOO_MANY_ROWS	SELECT INTO vraća više od jednog reda
VALUE_ERROR	Greška pri konverziji podataka, upotrebi funkcija ili ograničenja na podacima
ZERO_DIVIDE	Pokušaj delenja nulom
OTHERS	Ako ne znamo koji imenovani izuzetak obrađujemo