Računarstvo i informatika

Katedra za računarstvo
Elektronski fakultet u Nišu

Sistemi baza podataka
Napredni SQL (I deo)

Letnji semestar 2013/2014

Sadžaj

- Šta je PL/SQL
- Osnovna struktura PL/SQL programa
- Promenljive i tipovi podataka
- Kontrola toka u PL/SQL programima
- PL/SQL petlje
- PL/SQL kursori
- PL/SQL obrada izuzetaka

Šta je PL/SQL

- **Procedural Langauge extensions to SQL**
- PL/SQL je jezik koji predstavlja **proceduralno proširenje SQL-a**.
- Normalno SQL ne poseduje elemente koji bi omogućili razvoj strukturnih programa.
- Kada je korisnik ograničen samo na SQL on prosleđuje jednu po jednu naredbu DBMS-u.
- PL/SQL prevazilazi ovo ograničenje i **dodaje strukturne elemente** SQL-u.
- PL/SQL predstavlja strukturni programski jezik za ORACLE.

Šta je PL/SQL

- **Osnovni ciljevi PL/SQL**:
  - povećanje ekspresivnosti SQL-a
  - pristup rezultatima upita korišćenjem slogova (tuple-oriented way)
  - optimizacija kombinacija SQL naredbi
  - razvoj modularnih aplikacija za rad sa bazom podataka
  - višestruko korišćenje programskog koda
  - smanjenje cene održavanja i izmene aplikacija

Osnovna struktura PL/SQL programa

- **Osnovna gradivna jedinica** PL/SQL programa je **blok**.
- PL/SQL program se sastoji od blokova koji se mogu međusobno ugnježdavati.
- Tipično svaki blok predstavlja jednu (imenovanu) logičku operaciju u okviru programa.
- Blokovi koji predstavljaju funkcije, procedure ili pakete moraju imati ime.
- PL/SQL blok ima opcionu sekciju za deklaracije, obavezan deo koji sadrži PL/SQL naredbe i opcioni deo za obradu grešaka i izuzetaka.
- PL/SQL program, u obaveznom delu, može da sadrži SQL naredbe iz grupe DML naredbi (SELECT; INSERT, UPDATE, DELETE, ...).
- PL/SQL program **ne sme da sadrži SQL naredbe iz grupe DDL naredbi** (CREATE, DROP, ALTER, ...). Umesto ovih naredbi koriste se funkcije iz posebno definisanih paketa.

Osnovna struktura PL/SQL programa

```
[<Zaglavlje bloka>]
[DECLARE
/* Opciona sekcija deklaracija: promenljive, tipovi i lokalni potprogrami. */
<Konstante>
<Promenljive>
<Kursori>
<Korisnički definisani izuzeci>]
BEGIN
/* Izvršna sekcija: proceduralne i SQL naredbe. */
/* Jedina sekcija bloka koja je neophodna */
<PL/SQL naredbe>
[EXCEPTION
/* Opciona sekcija za obradu grešaka. */
<Obrada izuzetaka>]
END;
```

Osnovna struktura PL/SQL programa

- Sintaksa PL/SQL jezika ne pravi razliku između malih i velikih slova.
- Prilikom rada sa podacima (konstante, promenljive, podaci iz tabela) potrebno je voditi računa o malim i velikim slovima.
- Za komentare se mogu koristiti:
  - /* ... */ - komentarisanje čitavog bloka teksta
  - -- - komentarisanje jedne linije teksta
- PL/SQL program sadrži:
  - deklaracije promenljivih
  - naredbe dodele
  - naredbe za kontrolu toka
  - naredbe petlji
  - pozive funkcija i procedura
  - pozive trigger-a
  - naredbe za obradu grešaka i izuzetaka

Osnovna struktura PL/SQL programa

- Zaglavlje bloka (block header) definiše da li se radi o bloku funkcije, procedure ili paketa.
- Ukoliko zaglavlje bloka nije definisano tada se rado o **anonimnom bloku** (anonymous block).
- Svaki PL/SQL blok predstavlja novi PL/SQL izraz pa se PL/SQL blokovi mogu ugnježdavati kao izrazi nekog standardnog programskog jezika.
- Opseg važenja deklarisanih promenljivih je sličan opsegu važenja promenljivih kod standardnih programskih jezika.

Promenljive i tipovi podataka

- Promenljive se koriste za razmenu podataka između baze podataka i PL/SQL programa.
- Svaka promenljiva ima specifičan tip koji je vezan za nju (kao i kod standardnih programskih jezika).
- Tip promenljive može biti:
  - jedan od tipova koje SQL koristi za kolone baze podataka
  - jedan od generičkih tipova koji se koriste u PL/SQL-u (BOOLEAN – ima tri vrednosti: TRUE, FALSE i NULL)
  - deklarisan tako da je isti kao kod neke kolone baze podataka.

Promenljive i tipovi podataka

- Konstante, promenljive, kursori i izuzeci koji se koriste u nekom bloku moraju biti deklarisani u DECLARE sekciji tog bloka.
- Za deklaraciju promenljivih i konstanti se koristi sledeća sintaksa:

  <variablename> [**CONSTANT**] <datatype> [**NOT NULL**][:= <expression>];
- **NOT NULL** – deklarisana promenljiva ne može uzeti vrednost NULL
- <expression> - izraz koji se koristi za inicijilizaciju promenljive. Ukoliko izraz nije specificiran promenljiva se inicijalizuje na vrednost NULL.

Promenljive i tipovi podataka

```
DECLARE
plata NUMBER := 0.0;
datum_rodj DATE;
ime VARCHAR(20);
koef CONSTANT NUMBER(3,2) :=
1.5;
...
BEGIN ... END;
```

NUMBER je najčešće korišćen numerički tip u PL/SQL programima. Može se koristiti i za celobrojne i za razlomljene brojne vrednosti.

VARCHAR je najčešće korišćeni znakovni tip podataka. n predstavlja maksimalnu dužinu stringa u bajtovima.

Promenljive plata je eksplicitno inicijalizovana na vrednost 0.0.

Promenljive datum_rodj i ime su implicitno inicijalizovane na NULL.

Konstantama nakon inicijalizacije više nije moguće promeniti vrednost.

Promenljive i tipovi podataka

```
DECLARE
plata NUMBER;
ime RADNIK.LIME%TYPE;
...
BEGIN ... END;
```

U pojedinim situacijama je jako bitno da promenljiva ima isti tip kao i kolona u određenoj tabeli. Ukoliko se to ne obezbedi dodela i poređenje vrednosti mogu da funkcionišu na pogrešan način.

U takvim situacijama se koristi %TYPE operator.

Promenljiva ime je deklarisana korišćenjem identičnog tipa kao i kolona LIME u tabeli RADNIK.

```
DECLARE
r RADNIK%ROWTYPE;
...
BEGIN ... END;
```

Umesto vrednosti pojedinih kolona promenljiva se može deklarisati tako da prihvata vrednosti čitavih vrsta iz određene tabele.

Promenljiva r može da prihvati čitavu vrstu iz tabele RADNIK.

Podacima se pristupa korišćenjem sledeće sintakse: <variable_name>.<column_name>.

PL/SQL dodela vrednosti

- Najednostavniji način za dodelu vrednosti u PL/SQL programu je korišćenje operatora **:=.**

```
DECLARE
counter INTEGER :=0;
...
BEGIN
   counter:=counter+1;
   ...
END;
```

Dodela vrednosti prilikom deklaracije promenljive.

Dodela vrednosti u toku izvršavanja programa.

- Promenljivama se mogu dodeljivati i vrednosti koje se preuzimaju iz baze podataka.
- U tu svrhu se koristi **modifikovani oblik** SELECT naredbe kod koga sve pribavljene vrednosti moraju da se dodele promenljivama.

PL/SQL dodela vrednosti

SELECT <kolone(e)> **INTO** <lista promenljivih>
FROM <tabela(e)> **WHERE** <uslov>**;**

- SELECT naredba može da vrati **samo jedan rezultujući red**. Ukoliko vraća više od jednog reda sistem prijavljuje grešku.
- Vrednosti koje vraća SELECT naredba **moraju se proslediti promenljivama**. U suprotnom sistem prijavljuje grešku.
- Vrednosti koje vraća SELECT naredba moraju se po tipu poklopiti sa promenljivama kojima se prosleđuju.

PL/SQL dodela vrednosti

```
DECLARE
prosek NUMBER;
maks NUMBER;
radnik_rec RADNIK%ROWTYPE;
...
BEGIN
   SELECT *
   INTO radnik_rec
   FROM PREDUZECE.RADNIK
   WHERE MATBR = 101;

   SELECT AVG(PLATA), MAX(PLATA)
   INTO prosek, maks
   FROM PREDUZECE.RADNIK;
   ...
END;
```

Vrednosti koje vraća SELECT naredba se smeštaju u promenljivu koja predstavlja vrstu.

Vrednosti koje vraća SELECT naredba se smeštaju u pojedinačne promenljive.

PL/SQL kontrola toka

- Za kontrolu toka PL/SQL nudi **IF..THEN..ELSE** konstrukciju.

IF <uslov> **THEN** < sekvenca PL/SQL naredbi >
[ELSIF] <uslov> **THEN** <sekvenca PL/SQL naredbi>
...
[ELSE] < sekvenca PL/SQL naredbi > **ENDIF;**

- Ponašanje **IF..THEN..ELSE** odgovara sličnih konstrukcija kod drugih programskih jezika.
- Izvršava se prva sekvenca PL/SQL naredbi za koju uslov ima vrednost TRUE. U suprotnom izvršava se sekvenca PL/SQL naredbi u ELSE bloku.

PL/SQL petlje

- PL/SQL petlje se koriste da se izvršavanje sekvence PL/SQL naredbi ponovi više puta.
- **WHILE .. LOOP** petlja

```
[<< <ime labele> >>]
WHILE <uslov> LOOP
<sekvenca PL/SQL naredbi>;
END LOOP [<ime labele>];
```

- Petlji se može dodeliti ime. Dodeljeno ime se može iskoristiti kada se petlja napušta bezuslovno korišćenjem naredbe **EXIT** <ime labele>;.

PL/SQL petlje

- **FOR .. LOOP** petlja

```
[<< <ime labele> >>]
FOR <index> IN [REVERSE] <donja granica>..<gornja granica> LOOP
<sekvenca PL/SQL naredbi>;
END LOOP [<ime labele>];
```

- Brojač petlje <index> je **implicitno deklarisan** i opseg njegovog važenja je vezan samo za telo petlje.
- U okviru tela petlje vrednost brojača se može referencirati kao kod konstanti ali se **ne može menjati**.
- Ključna reč **REVERSE** nalaže da se vrednost brojača u petlji menja u suprotnom smeru: od gornje ka donjoj granici.

PL/SQL petlje

- Naredba **EXIT** [<ime labele>]; se koristi za bezuslovno napuštanje tela petlje.
- Naredba **EXIT** [<ime labele>] **WHEN** <uslov>; se koristi za napuštanje tela petlje kada je ispunjen određeni uslov.
- Ukoliko <ime labele> nije navedeno napušta se petlja u čijem telu se naredba EXIT nalazi.

PL/SQL kursori

- Kursor predstavlja kolekciju slogova (vrsta) koja se dobija kao rezultat izvršenja neke SQL naredbe.
- Kursor omogućava da se rezultati SQL naredbi obrađuju slog po slog (vrsta po vrsta).

CURSOR <ime kursora> [(<lista parametara>)] **IS** <SELECT naredba>;

- Parametri kursora se koriste kako bi se parametrizovalo izvršenje SELECT naredbe u kursoru.

<ime parametra> < tip parametra >

- Za tipove parametara se mogu koristiti standardni SQL tipovi podataka koje Oracle podržava.

PL/SQL kursori

```
DECLARE
CURSOR radnici (datum DATE) IS
SELECT LIME,PREZIME
FROM RADNIK
WHERE DATRODJ > datum
AND EXISTS (SELECT *
            FROM PREDUZECE.CLAN_PORODICE
            WHERE MATBRRAD=MBR);
BEGIN .. END;
```

Kursor vraća imena i prezimena radnika koji imaju članove porodice a rođeni su nakon specificiranog datuma.

PL/SQL kursori

- Da bi kursor mogao da se koristi mora da se najpre otvori.

```
OPEN <ime kursora> [(<lista parametara>)];
```

- Otvaranjem kursora izvršava se SELECT naredba nad kojom je kursor definisan a prvi slog rezultujuće tabele postaje aktivan.
- Rezultujućoj tabeli može da se pristupa korišćenjem naredbe FETCH.

```
FETCH <ime kursora> INTO [(<lista promenljivih>)];
```

PL/SQL kursori

- Naredba FETCH vrednosti kolona tekućeg sloga smešta u promenljive koje su navedene u listi.
- Naredni slog u rezultujućoj tabeli postaje aktivan odnosno tekući.
- Promenljive u listi po redosledu i tipu moraju da odgovaraju kolonama u rezultujućoj tabeli.
- Nakon što su obrađene sve vrste kursora on se zatvara korišćenjem naredbe CLOSE.

CLOSE <ime kursora>;

PL/SQL kursori

```
DECLARE
CURSOR radnici (datum DATE) IS
SELECT  *
FROM PREDUZECE.RADNIK
WHERE DATRODJ > datum
AND EXISTS (SELECT *
        FROM PREDUZECE.CLAN_PORODICE
        WHERE MATBRRAD=MATBR);
radnik  PREDUZECE.RADNIK%ROWTYPE;
ime VARCHAR2(20);
BEGIN
   OPEN radnici(' 01-JAN-1972 ');
    LOOP
        FETCH radnici INTO radnik;
        EXIT WHEN radnici%NOTFOUND;
        ime:=radnik.LIME;
    END LOOP;
    CLOSE radnici;
END;
```

Kursor vraća imena i prezimena radnika koji imaju članove porodice a rođeni su nakon specificiranog datuma.

%NOTFOUND je predikat koji proverava poslednju FETCH naredbu.

Ukoliko je poslednja FETCH naredba vratila slog %NOTFOUND ima vrednost FALSE.

PL/SQL kursori

```
DECLARE
CURSOR radnici (datum DATE) IS
SELECT  *
FROM PREDUZECE.RADNIK
WHERE DATRODJ > datum
AND EXISTS (SELECT *
        FROM PREDUZECE.CLAN_PORODICE
        WHERE MATBRRAD=MATBR);
ime VARCHAR2(20);
BEGIN
   FOR radnik IN radnici(' 01-JAN-1972 ')
   LOOP
      ime:=radnik.LIME;
    END LOOP;
END;
```

FOR petlja značajno pojednostavljuje korišćenje kursora.

Promenljiva koja se koristi za prihvatanje slogova iz rezultujuće tabele kursora se implicitno deklariše.

Petlja se brine o OPEN, FETCH, EXIT i CLOSE naredbi umesto korisnika.

PL/SQL kursori

```
DECLARE
ime VARCHAR2(20);
BEGIN
   FOR radnik IN (SELECT *
                  FROM PREDUZECE.RADNIK )
   LOOP
     ime:=radnik.LIME;
    END LOOP;
END;
```

Nema potrebe da se kursor eksplicitno deklariše.

Petlja FOR obavlja sve umesto korisnika.

PL/SQL kursori

- Ukoliko se kursori koriste za izmenu podataka prilikom deklaracije kursora potrebno je navesti koje kolone rezultujuće tabele se modifikuju.
- Na kraju deklaracije kursora se dodaje FOR UPDATE konstrukcija.

```
FOR UPDATE [OF<kolona(e)>];
```

- FOR UPDATE ima efekat da su specificirane kolone u rezultujućoj tabeli zaključane za ostale korisnike sve dok se kursor koristi.
- Ukoliko se kursori koriste u kombinaciji sa UPDATE ili DELETE naredbama mora se voditi računa da se ove naredbe odnose samo na tekući slog kursora.

PL/SQL kursori

```
DECLARE
CURSOR radnici IS SELECT  *
                  FROM PREDUZECE.RADNIK;
FOR UPDATE OF PLATA;
BEGIN
   FOR radnik IN radnici LOOP
      UPDATE RADNIK
       SET PLATA = PLATA * 1.15
       WHERE CURRENT OF radnici;
END LOOP;
END;
```

Kursor vraća svim radnicima povećava platu za 15%.

Ključna reč CURRENT OF se koristi da bi ukazala na tekući slog kursora.

PL/SQL izuzeci

- PL/SQL poseduje podršku za obradu izuzetaka koje mogu da se jave prilikom izvršavanja PL/SQL programa.
- Svaka greška ili upozorenje koje se javi prilikom izvršavanja PL/SQl programa dovodi do pojave izuzetka.
- PL/SQL razlikuje dve vrste izuzetaka:
  - sistemski definisane izuzetke – automatski se javlaju kad god se javi greška ili upozorenje za koje su vezani
  - korisnički definisani izuzeci – moraju se eksplicitno deklarisati u PL/SQl bloku u kome će se koristiti. Takođe, ne javljaju se automatski već ih korisnik mora eksplicitno kreirati.

PL/SQL izuzeci

```
DECLARE
CURSOR radnici IS SELECT  *
                  FROM PREDUZECE.RADNIK
FOR UPDATE ;
velika_plata EXCEPTION;
BEGIN
   FOR radnik IN radnici LOOP
      IF radnik.PLATA < 10000 THEN
         DELETE FROM RADNIK
         WHERE CURRENT OF radnici;
      ELSE RAISE velika_plata;
      END IF;
END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN /*neka obrada*/
    WHEN velika_plata THEN /*neka obrada*/
END;
```

Korisnički izuzetak mora eksplicitno da se kreira.

Obrada sistemskog izuzetka.

Obrada korisnički definisanog izuzetka.