Sistemi baza podataka

PL/SQL – Upravljanje transakcijama i trigeri

Sadržaj

- Upravljanje transakcijama
- Trigeri

Upravljanje transakcijama

Upravljanje transakcijama

- Zaključavanje podataka u toku transakcije
  - Mehanizam očuvanja konzistentnosti podataka u višekorisničkom režimu rada
    - Podaci u BP – deljeni resurs u vremenu
  - Načini zaključavanja podataka u DBMS
    - Implicitno
    - Eksplicitno

Načini zaključavanja podatka u DBMS-u

- Implicitno
  - automatsko zaključavanje
  - realizuje ga sam DBMS
  - obezbeđuje minimum smanjenja paralelizma u radu, neophodan za očuvanje konzistentnosti podataka u BP u višekorisničkom režimu rada
  - SELECT naredba – ne izaziva zaključavanja
  - DML naredbe – izazivaju ekskluzivno zaključavanje torki koje su predmet ažuriranja

Načini zaključavanja podatka u DBMS-u

- Eksplicitno
  - realizuje ga programer transakcionog programa
  - "pooštrava" restriktrivnost pristupa resursima – dodatno snižava mogući stepen paralelizma u radu

Upravljanje transakcijama

- Naredbe za upravljanje transakcijama
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

Upravljanje transakcijama

- Zahtev za potvrđivanje transakcije i oslobađanje resursa

  COMMIT

- Zahtev za kompletno poništavanje transakcije i oslobađanje resursa

  ROLLBACK

Upravljanje transakcijama

- Obeležavanje vremenske tačke napretka transakcije

  SAVEPOINT savepoint_name

- Zahtev za delimično poništavanje transakcije

  ROLLBACK TO [SAVEPOINT] savepoint_name

Upravljanje transakcijama

```
BEGIN
        INSERT INTO tabela VALUES (1, 2);
        ROLLBACK;
        INSERT INTO tabela VALUES (3, 4);
        COMMIT;
END;
```

Upravljanje transakcijama

```
BEGIN
        INSERT INTO tabela VALUES (5, 6);
        SAVEPOINT sp_1;
        INSERT INTO tabela VALUES (7, 8);
        SAVEPOINT sp_2;
        INSERT INTO tabela VALUES (9, 10);
        ROLLBACK sp_1;
        INSERT INTO tabela VALUES (11, 12);
        COMMIT;
END;
```

Trigeri

Trigeri baza podataka

- Definicija trigera baze podataka
  - Mehanizam koji se pokreće na događaj, vezan za manipulaciju podacima, ili samom bazom podataka i pokreće PL/SQL program
    - Događaji:
      - DML naredbe (INSERT, UPDATE, DELETE)
      - DDL naredbe (CREATE, ALTER, DROP)
      - DBMS događaji
        - AFTER SERVERERROR
        - AFTER LOGON
        - BEFORE LOGOFF
        - AFTER STARTUP
        - BEFORE SHUTDOWN

DML trigeri - komponente

- Oblast definisanosti
  - jedna tabela, ili
  - jedan pogled
- Vreme okidanja
  - BEFORE - *neposredno pre akcije naredbe*
  - AFTER - *neposredno nakon akcije naredbe*
  - INSTEAD OF - *umesto same akcije naredbe (samo za poglede)*

DML trigeri - komponente

- Pokretači trigera
  - INSERT
  - UPDATE [OF lista_kolona]
  - DELETE
- Frekvencija aktiviranja (tip) trigera
  - Svaka torka, koja je predmet DML naredbe – FOR EACH ROW
    - **Row Level Trigger**
    - dodatno, logički uslov pokretanja Row Level trigera
  - DML naredba u celini
    - **Statement Level Trigger**
- Aktivnost (procedura – PL/SQL blok), koju triger realizuje, kada je pokrenut

Oblikovanje trigera baze podataka

```
CREATE [OR REPLACE] TRIGGER Naziv_Trigera
BEFORE | AFTER | INSTEAD OF
    INSERT | DELETE | UPDATE [OF ListaObeležja]
        [OR INSERT | DELETE | UPDATE [ OF ListaObeležja ] ... ]
ON Naziv_Tabele
[FOR EACH ROW [WHEN (LogičkiUslovPokretanjaTrigera)]]
[REFERENCING OLD AS NazivOld NEW AS NazivNew]
[DECLARE
    Deklarativni deo - lokalne deklaracije
    ]
BEGIN
    Izvršni_deo -- proceduralni deo, specifikacija aktivnosti
[EXCEPTION Deo_za_obradu_izuzetaka –- naredbe oblika WHEN...THEN]
END [Naziv_Trigera];
```

Oblikovanje trigera baze podataka

```
ALTER TRIGGER Naziv_Trigera DISABLE | ENABLE;

ALTER TRIGGER Naziv_Trigera COMPILE;

DROP TRIGGER Naziv_Trigera;
```

Pokretanje trigera

- Obavlja se automatski
  - Pokretanjem naredbe koja predstavlja okidač trigera
  - **Ne može se izbeći voljom korisnika**
  - Ako su ostvareni svi uslovi za pokretanje trigera, Izvršava se PL/SQL blok, pridružen trigeru
- Ne postoji način da se triger, na bilo koji način "pozove" direktno
- **Ukoliko triger generiše izuzetak, operacija koja ga je aktivirala se prekida.**

Oblikovanje PL/SQL bloka trigera

- Realizuje se po uobičajenoj sintaksi PL/SQL-a
- Napomena**:** Zabranjeno je upravljanje transakcijom (upotreba naredbi COMMIT, ROLLBACK i SAVEPOINT)

Referenciranje predmetnih podataka i Row Level trigerima

- Podaci koji su predmet pokretačke DML naredbe, mogu biti referencirani unutar tela trigera:
  - :OLD.naziv_kolone
    - "stara" vrednost kolone - *before image*
    - ovakvo referenciranje ima smisla u slučaju pokretačke UPDATE ili DELETE naredbe
  - :NEW.naziv_kolone
    - "nova" vrednost kolone - *after image*
    - ovakvo referenciranje ima smisla u slučaju pokretačke UPDATE ili INSERT naredbe

Primer

- Primer trigera koji kontroliše da li korisnik pokušava za vrednost kolone Pre u tabeli Radnik da zada negativnu vrednost. Ukoliko je to slučaj, umesto vrednosti koju zadaje korisnik, kolona Pre treba da dobije vrednost 0.

```
CREATE OR REPLACE TRIGGER Trg_Radnik_Pre_INSUPD
BEFORE INSERT OR UPDATE OF Pre
ON RADNIK
FOR EACH ROW
WHEN (NEW.Pre < 0)
BEGIN
    :NEW.Pre := 0;
END Trg_Radnik_Pre_INSUPD;
```

Primer kreiranja funkcije za proveru valjanosti trinaeste, kontrolne cifre jedinstvenog matičnog broja građanina

```
CREATE OR REPLACE FUNCTION F_ProveraContrBr (P_Jmbg IN VARCHAR)
RETURN BOOLEAN IS
    v_KonCif CHAR(12) := '765432765432';
    v_RAZ NUMBER(4) := 0;
BEGIN
    FOR i IN 1..12 LOOP
        v_RAZ := v_RAZ + TO_NUMBER(SUBSTR(P_Jmbg, i, 1)) *
        TO_NUMBER(SUBSTR(v_KonCif, i, 1));
    END LOOP;
    v_RAZ :=  11 - MOD(v_RAZ, 11);
    IF v_RAZ != 10 AND
    MOD(v_RAZ, 11) = TO_NUMBER(SUBSTR(P_Jmbg, 13, 1)) THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END F_ProveraContrBr;
```

Primer

- Primer trigera koji kontroliše ispravnost unosa, ili modifikacije vrednosti jedinstvenog matičnog broja građanina (Jmbg) u tabeli Radnik, kontrolom trinaeste cifre novozadate vrednosti. Zabranjuje se unos neispravnog matičnog broja.

```
ALTER TABLE RADNIK ADD (Jmbg VARCHAR(13));

CREATE OR REPLACE TRIGGER Trg_Radnik_Jmbg_INSUPD
BEFORE INSERT OR UPDATE OF Jmbg
ON RADNIK
FOR EACH ROW
WHEN (NEW.Jmbg != OLD.Jmbg)
BEGIN
    IF NOT F_ProveraContrBr (:NEW.Jmbg) THEN
        Raise_Application_Error (-20000, 'GRESKA: Pogresan JMBG...');
    END IF;
END Trg_Radnik_Jmbg_INSUPD;
```

Logičke funkcije ispitivanja pokretačke naredbe trigera

- U telu trigera koji može biti pokrenut od strane više vrsta DML naredbi, moguće je ispitati, koja vrsta DML naredbe je pokrenula triger.
- Logičke funkcije ispitivanja vrste pokretačke DML naredbe
  - INSERTING
    - TRUE, ako je pokretač trigera bila naredba INSERT, inače FALSE
  - UPDATING [('naziv_kolone')]
    - TRUE, ako je pokretač trigera bila naredba UPDATE (opciono, nad navedenoj koloni u listi iza ključne reči UPDATE), inače FALSE
  - DELETING
    - TRUE, ako je pokretač trigera bila naredba DELETE, inače FALSE

Zadatak

- Napisati triger koji za svaku operaciju ažuriranja tabele Radnik, upisuje odgovarajuće podatke u arhivsku (journal) tabelu Radnik_JN. Za operaciju INSERT, isti podaci se prenose i u tabelu Radnik_JN. Za operaciju UPDATE ili DELETE, stare vrednosti torke se prenose u tabelu Radnik_JN.

Rešenje

```
CREATE TABLE Radnik_JN
(
    Dat DATE NOT NULL,
    Ope VARCHAR(3) NOT NULL,
    Mbr INTEGER NOT NULL,
    Ime VARCHAR(20),
    Prz  VARCHAR(25),
    Plt  DECIMAL(10, 2),
    CONSTRAINT radnik_JN_PK PRIMARY KEY (Dat, Ope, Mbr)
)
```

Rešenje

```
CREATE OR REPLACE TRIGGER Trg_Radnik_JN_INSUPDDEL
BEFORE INSERT OR UPDATE OF Prz, Ime, Plt OR DELETE
ON RADNIK
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO Radnik_JN (Dat, Ope, Mbr, Ime, Prz, Plt) VALUES
        (SYSDATE, 'INS', :NEW.Mbr, :NEW.Ime, :NEW.Prz, :NEW. Plt);
    ELSIF UPDATING ('Prz') THEN
        INSERT INTO Radnik_JN (Dat, Ope, Mbr, Prz) VALUES  (SYSDATE, 'UPD',
    :OLD.Mbr, :OLD.Prz);
    ELSIF UPDATING ('Ime') THEN
        INSERT INTO Radnik_JN (Dat, Ope, Mbr, Ime) VALUES (SYSDATE, 'UPD',
    :OLD.Mbr, :OLD.Ime);
    ELSIF UPDATING ('PLT') THEN
        INSERT INTO Radnik_JN (Dat, Ope, Mbr, Plt) VALUES   (SYSDATE, 'UPD',
    :OLD.Mbr, :OLD. Plt); -- Nastavak na sledećem slajdu
```

Rešenje

```
    ELSIF DELETING THEN
        INSERT INTO Radnik_JN (Dat, Ope, Mbr, Ime, Prz, Plt) VALUES
        (SYSDATE, 'DEL', :OLD.Mbr, :OLD.Ime, :OLD.Prz, :OLD. Plt);
    END IF;
END Trg_Radnik_JN_INSUPDDEL;
```

Oblasti primene trigera

- Realizacija kontrole ograničenja podataka, na nivou DBMS-a
- Realizacija pravila poslovanja, koja rezultuju u obavezama primene određenih operacija nad BP, u zahtevanom redosledu, u okrivu iste DBMS transakcije
- Realizacija zaštite BP od neovlašćenog pristupa
- Praćenje aktivnosti (izvršenja operacija) nad podacima u BP (Journaling)
- Automatsko prosleđivanje podataka ili poruka, ili pokretanje programa, kao rezultat ažuriranja BP
- Osvežavanje materijalizovanih pogleda (replikacionih kopija) u distribuiranim bazama podataka

Trigeri baza podataka - zadaci

- **Zadatak 1.** Formirati triger koji će, nad tabelom Radnik, zabraniti bilo koji pokušaj modifikacije vrednosti primarnog ključa (matičnog broja radnika).
- **Zadatak 2.** Formirati triger koji će, nad tabelom Radnik, obezbediti da se prilikom unosa nove torke, uvek zada vrednost matičnog broja kao prva sledeća vrednost iz kreiranog generatora sekvence, bez obzira na to šta je korisnik zadao za vrednost Mbr u klauzuli VALUES.

Rešenje 1.

```
CREATE OR REPLACE TRIGGER Trg_Radnik_mbr_UPD
BEFORE UPDATE OF mbr
ON RADNIK
FOR EACH ROW
WHEN (NEW.mbr != OLD.mbr)
DECLARE
    exc EXCEPTION;
BEGIN
     RAISE exc;
EXCEPTION
     WHEN exc THEN
        Raise_Application_Error (-20000, 'GRESKA: MBR se ne moze menjati');
END Trg_Radnik_mbr_UPD;
```

Rešenje 2.

```
CREATE OR REPLACE TRIGGER KeyCon_Radnik_PK_GenSeq
    BEFORE INSERT
    ON RADNIK
    FOR EACH ROW
    BEGIN
       SELECT Seq_Mbr.NEXTVAL
       INTO :NEW.MBR
       FROM SYS.DUAL;
    END KeyCon_Radnik_PK_GenSeq;
```

Trigeri baza podataka - zadaci

- **Zadatak 3.** Formirati triger koji će, nad tabelom Radnik, obezbediti da se prilikom unosa nove torke ili izmene, imena i prezimena radnika uvek zadaju velikim slovima.

Rešenje 3.

```
CREATE OR REPLACE TRIGGER Trg_Radnik_Zad3
BEFORE INSERT OR UPDATE OF ime, prz
ON RADNIK
FOR EACH ROW
BEGIN
    SELECT UPPER(:NEW.Ime)
    INTO :NEW.Ime
    FROM SYS.DUAL;
    SELECT UPPER(:NEW.Prz)
    INTO :NEW.Prz
    FROM SYS.DUAL;
END Trg_Radnik_Zad3;
```

Trigeri baza podataka - zadaci

- **Zadatak 4.** Formirati triger koji će, nad tabelom Radnik, obezbediti da se prilikom unosa nove torke, brisanja ili izmene vrednosti obeležja pre triger aktivira samo ukoliko je stara plata veća od 8000. Ukoliko se radi o brisanju zabraniti ga. Ukoliko se radi o unosu, uneti samo one čija je premija veća od deset posto plate a ukoliko se radi o izmeni dozvoliti izmenu samo ako je nova vrednost premije veća pet posto od prethodne vrednosti premije.

Rešenje 4.

```
CREATE OR REPLACE TRIGGER Trg_Radnik_zad4
BEFORE INSERT OR DELETE OR UPDATE OF pre
ON RADNIK
FOR EACH ROW
WHEN (old.PLT>8000)
DECLARE
      exc1 EXCEPTION;
      exc2 EXCEPTION;
      exc3 EXCEPTION;
BEGIN
     IF INSERTING THEN
            IF NOT(:NEW.Pre >:NEW.Plt*0.1) THEN
              RAISE exc1;
            END IF;
     ELSIF UPDATING ('Pre') THEN
            IF NOT(:NEW.Pre >:OLD.Pre*1.05) THEN
              RAISE exc2;
            END IF;    -- Nastavak na sledećem slajdu
```

Rešenje 4.

```
    ELSIF DELETING THEN
          RAISE exc3;
    END IF;
EXCEPTION
    WHEN exc1 THEN
      Raise_Application_Error (-20000, 'GRESKA: Ne moze se uneti novi radnik');
    WHEN exc2 THEN
       Raise_Application_Error (-20000, 'GRESKA: Ne moze se menjati radnik');
    WHEN exc3 THEN
       Raise_Application_Error (-20000, 'GRESKA: Ne moze se brisati radnik');
END Trg_Radnik_zad4;
```

Kraj!

Hvala na pažnji!