PLSQL vežba – zadaci i rešenja

Date su šeme relacija:

Dept ({deptno, dname, loc}, {deptno})
Emp ({empno, ename, job, mgr, hiredate, sal, comm, deptno}, {empno})
Customer ({custid, name, address, city, state, zip, area, phone, repid, creditlimit, comments}, {custid})
Ord ({ordid, orderdate, commplan, custid, shipdate, total}, {ordid})
Item ({ordid, itemid, prodid, actualprice, qty, itemtot}, {ordit, itemid})
Salgrade ({grade, losal, hisal},{grade})
Jobs ({job, grade}, {job})

sa referencijalnim integritetima:

Emp[deptno] $\subseteq$ Dept [deptno]
Emp[mgr] $\subseteq$ Emp[empno]
Customer[repid] $\subseteq$ Emp[empno]
Ord[custid] $\subseteq$ Customer [custid]
Item [ordid] $\subseteq$ Ord [ordid]
Jobs [grade] $\subseteq$ Salgrade [grade]

Značenja mnemonika:

Mnemonik	**Opis**	**Mnemonik**	**Opis**
DEPTNO	br. departmenta	CUSTID	id klijenta
DNAME	naziv departmenta	NAME	naziv klijenta
LOC	lokacija	ADDRESS	adresa
EMPNO	id zaposlenog	CITY	grad
ENAME	prezime zaposlenog	STATE	država
JOB	posao	ZIP	poštanski broj
MGR	šef (menadžer)	AREA	oblast
HIREDATE	datum zapošljavanja	PHONE	telefon
SAL	plata	REPID	id prodavca
COMM	bonus	CREDITLIMIT	limit kredita
GRADE	platni razred	COMMENTS	komentari
ORDID	id narudžbine	LOSAL	donja granica plate
ORDERDATE	datum naruđžbine	HISAL	gornja granica plate
COMMPLAN	pritužbe		
SHIPDATE	datum isporuke		
TOTAL	ukupan iznos naruđžbine		
ITEMID	id stavke		
PRODID	id proizvoda		
ACTUALPRICE	prava cena		
QTY	količina		
ITEMTOT	iznos stavke		

1. Napisati proceduru koja za prosleđeni broj departmana (deptno) ispisuje odgovarajuće poruke za zaposlene u tom departmanu i to:
   a. Ako je plata radnika veća od prosečne plate u tom departmanu:
      Radnik <prezime radnika> ima platu veću od posečne.
   b. Ako je plata radnika manja od prosečne plate u tom departmanu:
      Radnik <prezime radnika> ima platu manju od posečne
   c. Inače prikazati poruku:
      Radnik <prezime radnika> ima prosečnu platu.

   Zadatak realizovati upotrebom kursora.

REŠENJE

```
CREATE OR REPLACE
PROCEDURE prosecnaplata (p_deptno dept.deptno%type)
IS
   CURSOR emp_cursor IS
   SELECT ename,sal,d.prosplt
   FROM emp, (SELECT avg(sal) prosplt FROM emp WHERE deptno=p_deptno) d
   WHERE deptno=p_deptno;

BEGIN
```

```
   FOR emp_rec IN emp_cursor LOOP
      IF emp_rec.sal>emp_rec.prosplt THEN
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima platu vecu od
   prosecne.');
      ELSIF emp_rec.sal<emp_rec.prosplt THEN
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima platu manju od
   prosecne.');
      ELSE
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima prosecnu platu.');
      END IF;
   END LOOP;
END prosecnaplata;
/
begin prosecnaplata(30); end;
```

Druga varijanta

```
CREATE OR REPLACE
PROCEDURE prosecnaplata2 (p_deptno dept.deptno%type)
IS
   CURSOR emp_cursor IS
   SELECT ename,sal
   FROM emp WHERE deptno=p_deptno;
   prosplt NUMBER;
BEGIN
   SELECT avg(sal) INTO prosplt FROM emp WHERE deptno=p_deptno;
   FOR emp_rec IN emp_cursor LOOP
      IF emp_rec.sal>prosplt THEN
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima platu vecu od
   prosecne.');
      ELSIF emp_rec.sal<prosplt THEN
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima platu manju od
   prosecne.');
      ELSE
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ima prosecnu platu.');
      END IF;
   END LOOP;
END PROSECNAPLATA2;
/
```

2. Napisati triger koji će se okidati pri svakom upisu ili modifikaciji tabele Emp i to tako da se implementira poslovno pravilo: ako je nova vrednost za deptno=30 (prodaja) tada, ako je nova vrednost za posao (job) različita od (‘MANAGER’, ‘SALESMAN’, ‘CLERK’) prijaviti grešku da u departmanu prodaje na može raditi radnik na mestu <novi posao>.

REŠENJE

```
CREATE OR REPLACE TRIGGER prodaja_chk
BEFORE INSERT OR UPDATE OF deptno ON emp
FOR EACH ROW
WHEN (NEW.deptno=30)
BEGIN
   IF UPPER(:NEW.job) NOT IN ('MANAGER','SALESMAN','CLERK') THEN
      RAISE_APPLICATION_ERROR(-20006, 'U departmanu prodaje ne moze raditi radnik na
   mestu '||:new.job);
   END IF;
END prodaja_chk;
/
insert into emp(empno,ename,job,deptno) values (1,'savic','ANALYST',30);
insert into emp(empno,ename,job,deptno) values (1,'savic','ANALYST',20);
insert into emp(empno,ename,job,deptno) values (2,'savic','CLERK',30);
```

3. Napisati proceduru koja za prosleđeni datum (hiredate) ispisuje ime, platu i datum zapošljavanja za sve radnike koji su zapošljeni pre tog datuma i čija je plata veća od 10000. Zadatak realizovati upotrebom kursora.

REŠENJE

```
CREATE OR REPLACE
PROCEDURE RadniciDatum(p_hiredate emp.hiredate%TYPE)
AS
   CURSOR date_cursor IS
   SELECT ENAME,SAL,HIREDATE FROM EMP
   WHERE hiredate < p_hiredate
   AND sal>1000;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Radnici zaposleni pre '|| p_hiredate);
   FOR date_rec IN date_cursor LOOP
      DBMS_OUTPUT.PUT_LINE (date_rec.ename|| ' plata=' || date_rec.sal|| ', zaposljen
   ' || date_rec.hiredate);
   END LOOP;
END;
/

begin radnicidatum(to_date('28.09.1981','dd.mm.yyyy')); end;
```

4. Napisati triger koji će implementirati poslovno pravilo: može da postoji samo jedan predsednik, tj. samo jedan zaposleni čiji je posao 'PRESIDENT'.

REŠENJE

```
CREATE OR REPLACE TRIGGER moze_biti_samo_jedan
BEFORE INSERT OR UPDATE OF job ON emp
FOR EACH ROW
WHEN (UPPER(NEW.JOB)='PRESIDENT')
DECLARE
  broj NUMBER;
BEGIN
  SELECT COUNT(*) INTO BROJ FROM EMP WHERE UPPER(JOB)='PRESIDENT';
  IF broj>0 THEN
      RAISE_APPLICATION_ERROR(-20008, 'Moze postojati samo jedan predsednik!');
  END IF;
END moze_biti_samo_jedan;
/
insert into emp(empno,ename,job,deptno) values (3,'savic','PRESIDENT',30);
insert into emp(empno,ename,job,deptno) values (3,'savic','CLERK',10);
```

5. Napisati proceduru koja za prosleđni broj departmana (deptno) ispisuje odgovarajuće poruke za zaposlene u tom departmanu i to:
   a. Ako je plata radnika <5000 i ako mu je šef (mgr) radnik sa brojem 101 ili 124, prikazati poruku:
      Radnik <prezime radnika> treba da dobije povišicu plate.
   b. Inače prikazati poruku:
      Radnik <prezime radnika> ne treba da dobije povišicu plate.

   Zadatak realizovati upotrebom kursora.

REŠENJE

```
CREATE OR REPLACE PROCEDURE povisica(p_deptno dept.deptno%type)
IS
   CURSOR emp_cursor IS
   SELECT ename, mgr, sal
   FROM emp
   WHERE deptno=p_deptno;

BEGIN
   FOR emp_rec IN emp_cursor LOOP
      IF emp_rec.sal<3000 AND emp_rec.mgr IN (7839, 7566) THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' treba da dobije povisicu
   plate.');
      ELSE
            DBMS_OUTPUT.PUT_LINE('Radnik '||emp_rec.ename||' ne treba da dobije
   povisicu plate.');
      END IF;
   END LOOP;
END povisica;
/
begin povisica(20); end;
```

6. Napisati triger koji će se okidati pri svakoj modifikaciji tabele Dept i to tako da se implementira poslovno pravilo: kada se departman seli na novu lokaciju (loc) svi zaposleni na tom departmanu dobijaju povisicu plate 2%.

REŠENJE

```
CREATE OR REPLACE TRIGGER povisica
BEFORE UPDATE OF loc ON dept
FOR EACH ROW
WHEN (NEW.loc!=OLD.loc)
--declare
--  pragma AUTONOMOUS_TRANSACTION;
BEGIN
   UPDATE emp
   SET sal=sal*1.02
   WHERE emp.deptno=:NEW.deptno;
-- COMMIT;
END povisica;
/
update dept set loc='Novi Sad' where deptno=10;
```

(Drugo rešenje je kada se dodaju i naredbe koje su stavljene kao komentari – forsira sa COMMIT).

7. Napisati PLSQL proceduru koja za prosleđeni id broj narudžbine ispisuje sve stavke te narudžbine i ažurira kolonu total u tabeli Ord tako što izračuna ukupan iznos narudzbine i postavi u kolonu total za datu narudžbinu.

REŠENJE

```
CREATE OR REPLACE PROCEDURE narudzbina(p_ordid ord.ordid%TYPE)AS
   v_total  ord.total%TYPE;

   CURSOR ITEM_CUR IS
   SELECT prodid, qty, actualprice, itemtot
   FROM item
   WHERE ordid=p_ordid;

BEGIN
   SELECT SUM(itemtot) INTO V_TOTAL FROM ITEM   WHERE ORDID=P_ORDID;
   DBMS_OUTPUT.PUT_LINE('Narudzbina broj '|| P_ORDID ||' sadrzi sledece stavke: ');
   DBMS_OUTPUT.PUT_LINE(' R.broj   Proizvod   Kolicina   Cena   Ukupno');
   DBMS_OUTPUT.PUT_LINE('*****************************************************');

   FOR item_rec IN item_cur LOOP
      DBMS_OUTPUT.PUT_LINE(item_cur%rowcount||'    '||ITEM_REC.PRODID||'     '||
   ITEM_REC.QTY||'      '|| item_rec.actualprice||'     '||item_rec.itemtot);
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Ukupna vrednost narudžbine je: '||V_TOTAL);
   UPDATE ORD SET TOTAL=V_TOTAL WHERE ORDID=P_ORDID;
   COMMIT;
END narudzbina;
```

8. Napisati PLSQL triger koji će se okidati svaki put pri unosu ili ažuriranju tabele Ord koji implementira poslovno pravilo: ako je ukupan iznos narudžbine veći od limita kredita klijenta ispisati poruku da je prekoračen limit kredita za tog klijenta.

REŠENJE

```
CREATE OR REPLACE TRIGGER ord_tot
AFTER INSERT OR UPDATE OF total ON ord
FOR EACH ROW
DECLARE
   v_creditlimit customer.creditlimit%type;
BEGIN
  select creditlimit
  into v_creditlimit
  from customer
  where custid=:new.custid;
  IF V_CREDITLIMIT<:NEW.TOTAL THEN
      raise_application_error(-20110,'Prekoracen limit.' );
  END IF;
END;
/
```

9. Napisati proceduru pod nazivom PROMOTE_CLERK, koja sve radnike iz tabele EMP koji rade na poslu CLERK i koji zarađuju preko 1000, promoviše tako što vednost obeležja JOB postavlja na SR.CLERK, a platu povećava za 10%.

REŠENJE

```
CREATE OR REPLACE PROCEDURE promote_clerk IS
   CURSOR c_emp IS
   SELECT empno,job FROM emp
   WHERE  UPPER(job)='CLERK' AND sal>1000
   FOR update OF job;
BEGIN
   FOR r_emp IN c_emp LOOP
      UPDATE emp
      SET job='SR.CLERK',
      sal=1.1*sal
      WHERE CURRENT OF c_emp;
   END LOOP;
   COMMIT;
END promote_clerk;
/
```

10. Kreirati triger CHCK_SAL_TG, koji će se okidati pre svakog upisa ili ažuriranja tabele EMP za obeležje SAL. Potrebno je proveriti da li je vrednost obeležja SAL između vrednosti LOSAL i HISAL iz tabele SALGRADE za taj posao. Ako nije ispisati odgovarajuću poruku.

REŠENJE

```
CREATE OR REPLACE TRIGGER CHCK_SAL_TG
BEFORE INSERT OR UPDATE OF SAL ON EMP
FOR EACH ROW
DECLARE
   V_LOSAL SALGRADE.LOSAL%TYPE;
   V_HISAL SALGRADE.HISAL%TYPE;

BEGIN
   SELECT LOSAL, HISAL INTO V_LOSAL, V_HISAL
   FROM JOBS J, SALGRADE S
   WHERE UPPER(J.JOB)=UPPER(:NEW.JOB) AND J.GRADE = S.GRADE;
   IF (:NEW.SAL<V_LOSAL) OR (:NEW.SAL>V_HISAL) THEN
      RAISE_APPLICATION_ERROR(-20001,'GRESKA,PLATA VAN GRANICA');
   END IF;
END CHCK_SAL_TG;
/
```

11. Napisati proceduru pod nazivom **SET_COMM**, koja za sve radnike iz tabele EMP proverava kolika im je godišnja plata i u zavisnosti od iznosa postavlja vrednost kolone COMM na sledeći način:

Godišnja plata	COMM
>=20 000	2 000
10 000 =< plata <20 000	1 000
<10 000	500

REŠENJE

```
CREATE OR REPLACE PROCEDURE SET_COMM
IS
   CURSOR EMP_SAL_CUR IS
   SELECT EMPNO, SAL*12 AS GODISNJA
   FROM EMP
   FOR UPDATE OF COMM;

   V_COMM EMP.COMM%TYPE;
BEGIN
   FOR R_EMP IN EMP_SAL_CUR LOOP
      IF R_EMP.GODISNJA>=20000 THEN
            V_COMM:=2000;
      ELSIF ((R_EMP.GODISNJA>=10000) AND (R_EMP.GODISNJA<20000)) THEN
            V_COMM:=1000;
      ELSE
            V_COMM:=500;
      END IF;
      UPDATE EMP  SET COMM=V_COMM WHERE CURRENT OF EMP_SAL_CUR;

   END LOOP;
   COMMIT;
END SET_COMM;
/
```

12. Kreirati triger **SET_SAL_TG**, koji će se okidati nakon svakog ažuriranja obeležja LOSAL u tabeli SALGRADE. Ako je vrednost LOSAL povećana postaviti za svakog zaposlenog iz tabele EMP koji radi na poslu iz tog platnog razreda i imao je minimalnu platu, vrednost obeležja SAL na novu vrednost. Ako je nova vrednost za LOSAL >HISAL javiti grešku.

REŠENJE

```
CREATE OR REPLACE TRIGGER SET_SAL_TG
BEFORE UPDATE OF LOSAL ON SALGRADE
FOR EACH ROW
BEGIN
   IF :NEW.LOSAL>:NEW.HISAL THEN
      RAISE_APPLICATION_ERROR(-200001,'GRESKA');
   ELSIF :NEW.LOSAL>:OLD.LOSAL THEN
      UPDATE EMP
      SET SAL=:NEW.LOSAL
      WHERE JOB IN (SELECT JOB FROM JOBS WHERE JOBS.GRADE= :OLD.GRADE)
      AND SAL<:NEW.LOSAL;
   END IF;
END SET_SAL_TG;
/
```

13. Napisati proceduru koja kao ulazne vrednosti ima broj departmana, donju granicu i gornju granicu za vrednost plate. Procedura vrši povećanje plate za radnike zadatog departmana i to na sledeći način:
    - ako je plata radnika manja od donje granice, povećanje je 10%,
    - ako je plata radnika između donje i gornje granice, povećanje je 5%,
    - ako je plata radnika veća od gornje granice, povećanje je 2%,

U slučaju da je doslo do greške i nema podataka javiti odgovarajuću poruku.

REŠENJE

```
CREATE OR REPLACE
PROCEDURE DEPT_RAISE(P_DEPTNO DEPT.DEPTNO%TYPE, P_LOSAL EMP.SAL%TYPE, P_HISAL
   EMP.SAL%TYPE )
IS
   CURSOR EMP_SAL_CUR IS
   SELECT SAL AS OLDSAL
   FROM EMP
   WHERE DEPTNO= P_DEPTNO
   FOR UPDATE OF SAL;
   R_EMP EMP_SAL_CUR%ROWTYPE;

   V_RAISE NUMBER;
   GRESKA EXCEPTION;

BEGIN
  OPEN EMP_SAL_CUR;
   LOOP
    FETCH EMP_SAL_CUR INTO R_EMP;
    IF EMP_SAL_CUR%ROWCOUNT<1 OR EMP_SAL_CUR%NOTFOUND IS NULL THEN
       RAISE GRESKA;
    END IF;
    EXIT WHEN EMP_SAL_CUR%NOTFOUND OR EMP_SAL_CUR%NOTFOUND IS NULL;
    IF R_EMP.OLDSAL<P_LOSAL THEN
            V_RAISE:=1.1;
      ELSIF ((R_EMP.OLDSAL >= P_LOSAL) AND (R_EMP.OLDSAL<= P_HISAL)) THEN
            V_RAISE:=1.05;
      ELSE
            V_RAISE:=1.02;
      END IF;
      UPDATE EMP
      SET SAL=SAL*V_RAISE
      WHERE CURRENT OF EMP_SAL_CUR;
   END LOOP;
   CLOSE EMP_SAL_CUR;
   COMMIT;
EXCEPTION
   WHEN GRESKA THEN
      RAISE_APPLICATION_ERROR(-20001,'NEMA RADNIKA U TOM DEPARTMANU');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'NEKA GRESKA');

END DEPT_RAISE;
/
```

14. Kreirati triger koji će se okidati pri pokušaju ažuriranja obeležja DEPTNO u tabeli EMP. Potrebno je proveriti da li je to validan broj departmana, tj. da li ta vrednost postoji u tabeli DEPT. U slučaju da ne postoji javiti odgovarajuću grešku.

REŠENJE

```
CREATE OR REPLACE TRIGGER DEPT_CHECK
BEFORE UPDATE OF DEPTNO ON EMP
FOR EACH ROW
DECLARE
  V_DEPTNO DEPT.DEPTNO%TYPE;
BEGIN
  SELECT DEPTNO INTO V_DEPTNO FROM DEPT WHERE DEPTNO=:NEW.DEPTNO;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20101,'NE POSTOJI DEPARTMAN SA TIM BROJEM!');
END;
```

15. Napisati funkciju koja kao ulazni podatak ima vrednost plate, a kao izlaz vraća id zaposlenog. Ispisati ime i matični broj radnika sa tom platom. Obraditi moguće greške:
    a. Ako nema podataka javiti poruku: Nema radnika sa platom <vrednost plate>
    b. Ako ima suviše podataka javiti poruku: Više od jednog radnika ima platu <vrednost plate>
    c. Ako je neka druga greška: Došlo je do neke greske!

REŠENJE

```
CREATE OR REPLACE FUNCTION PLATA_ID
(P_SAL IN EMP.SAL%TYPE)
RETURN NUMBER AS
  V_ENAME EMP.ENAME%TYPE;
  V_EMPNO EMP.EMPNO%TYPE;
BEGIN
  SELECT ENAME, EMPNO INTO V_ENAME,V_EMPNO
  FROM EMP WHERE SAL=P_SAL;
  DBMS_OUTPUT.PUT_LINE('IME='||V_ENAME||' ID='||V_EMPNO);
  RETURN V_EMPNO;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20001, 'NEMA RADNIKA SA PLATOM '||P_SAL);
  WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20002, 'VIŠE OD JEDNOG RADNIKA IMA PLATU '||P_SAL);
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'DOŠLO JE DO NEKE GRESKE!');
END PLATA_ID;
```

16. Kreirati triger koji će se okidati pri pokušaju ažuriranja obeležja SAL u tabeli EMP. Potrebno je implementirati poslovno pravilo da se plata radnika ne može smanjivati, niti povećati za više od 10%.

REŠENJE

```
CREATE OR REPLACE TRIGGER CHECK_SALARY
BEFORE UPDATE OF SAL ON EMP
FOR EACH ROW
WHEN ((NEW.SAL < OLD.SAL) OR
      (NEW.SAL > (OLD.SAL * 1.1)))

BEGIN
   RAISE_APPLICATION_ERROR(-20508,'ZABRANJENO JE SMANJIVATI PLATU, KAO I POVECAVATI
   JE ZA VISE OD 10%');
END;
/
```

17. Kreirati set trigera koji će implementirati referencijalni integritet EMP[DEPTNO] $\subseteq$ DEPT [DEPTNO]. Treba obezbediti da se pri promeni vrednost obeležja DEPTNO u tabeli DEPT, azuriraju sve njegove pojave u zavisnoj tabeli EMP. Takođe, omogućiti kaskadno brisanje – kada se obriše departman obrisati i sve radnike koji rade u tom departmanu.

REŠENJE

```
CREATE OR REPLACE TRIGGER check_deptno
BEFORE INSERT OR UPDATE OF DEPTNO ON EMP
FOR EACH ROW
DECLARE
   V_OK INTEGER;
BEGIN
   SELECT COUNT(*) INTO V_OK
   FROM DEPT WHERE DEPTNO=:NEW.DEPTNO;
   IF V_OK<1 THEN
    RAISE_APPLICATION_ERROR(-20508,'NE POSTOJI TAJ DEPARTMAN!');
   END IF;
END;
/

CREATE OR REPLACE TRIGGER cascade_updates
BEFORE DELETE OR UPDATE of deptno ON dept
FOR EACH ROW
BEGIN
   IF DELETING THEN
    DELETE FROM emp WHERE deptno=:OLD.deptno;
   ELSE
    UPDATE emp SET deptno=:NEW.deptno WHERE deptno=:OLD.deptno;
   END IF;
END;
/
```