Kapitola 10

View

10.1 Zadanie cvičenia - View

1. Definujte pohľad *pohlad_st* – menný zoznam študentov (meno, priezvisko, st_skupina)
2. Definujte pohľad *pohlad_uc* – menný zoznam učiteľov, ktorí niečo učia
3. Definujte pohľad *osoby* s atribútmi meno, priezvisko, rod_cislo (len z tabuľky os_udaje).
4. Vyberte si rodné číslo ľubovoľného študenta (zapamätajte si ako sa volá a aké má osobné číslo), zmažte najprv všetky jeho zapísané predmety a študentské údaje. Potom ho vymažte z pohľadu *osoby*.
   Skontrolujte čo ostalo v pohľade a čo v tabuľke .
5. Definujte pohľad *studenti* s atribútmi meno, priezvisko, rod_cislo, os_cislo.
6. Vyberte si **osobné číslo** ľubovoľného študenta z pohľadu *studenti* (zapamätajte si ako sa volá a aké má osobné a rodné číslo), zmažte najprv všetky jeho zapísané predmety. Potom ho vymažte z pohľadu *studenti*.
   Skontrolujte čo ostalo v pohľade a čo v tabuľkách.
7. Vyberte si **rodné číslo** ľubovoľného študenta z pohľadu *studenti* (zapamätajte si ako sa volá a aké má osobné a rodné číslo), zmažte najprv všetky jeho zapísané predmety. Potom ho vymažte z pohľadu *studenti*.
   Skontrolujte čo ostalo v pohľade a čo v tabuľkách.
8. Vytvorte trigger, tak aby vymazávanie z pohľadu *studenti* fungovalo korektne.
9. Definujte pohľad *bakalarske_predmety* – zoznam bakalárskych predmetov z tabuľky predmet (ich čísla začínajú na 'B'), obsahujúci všetky not null atribúty.(Bez klauzuly CHECK OPTION)
10. Vložte do pohľadu *bakalarske_predmety* nový predmet IE12 – Prognostika.

10.2 Doplnenie cvičenia - Letisko - View

1. Vytvorte trigger na automatické priradenie id_letenky zo sekvencie.
2. Vytvorte trigger, ktorý na letenku automaticky pridelí voľné sedadlo z danej triedy pre daný let.
3. Pomocou triggra skontrolujte korektnosť dátumu pri zadávaní údajov o lete. (Nemôže pristáť skôr ako odletel, maximálna dĺžka letu nech je 12 hod.)
4. Pomocou triggra zabráňte predaj leteniek na už realizované lety, prípadne menej ako 30 minút pred odletom lietadla.

10.3 View (Pohlad)

View (Pohľad) je predpripravený select, ktorý sa vykoná keď sa odkážeme na view.

10.3.1 Syntax

```
CREATE [OR REPLACE] [ FORCE | NOFORCE ]
VIEW [schema.] pohľad [(alias_stlpca [,...])]
AS Select-prikaz
[WITH [ READ ONLY | CHECK OPTION [CONSTRAINT obmedzenie] ] ]
|
CREATE VIEW [schema.] pohľad [(alias_stlpca [,...])]
AS Select-prikaz
[WITH [ CASCADED | LOCAL] CHECK OPTION ]
```

Poznámky

- schema – názov schémy, v ktorej sa má pohľad nachádzať
- OR REPLACE – predefinovanie pohľadu, ak už bol definovaný
- FORCE – tento druh pohľadu je možné vytvoriť aj vtedy, ak tabuľky (objekty), z ktorých má byť pohľad odvodený neexistujú, alebo užívateľ, ktorý vytvára pohľad nemá na ne práva
- NOFORCE – implicitne – pohľad je možné vytvoriť len vtedy, ak bázické tabuľky (objekty) existujú a užívateľ má na ne práva.
- READ ONLY – nedovolí operácie INSERT, UPDATE ani DELETE nad pohľadom.
- CHECK OPTION – kontroluje dodržiavanie podmienky WHERE pri operáciách INSERT, UPDATE, DELETE do pohľadu
- CONSTRAINT – pomenovanie obmedzenia
- CASCADED - kontrola podmienok v odvodených pohľadoch
- LOCAL – obmedzenie kontroly podmienok len na podmienku definovanú v danom pohľade

Príklad 10.1: Jednoduchý pohľad

```
CREATE OR REPLACE VIEW pohl1
AS
SELECT meno, priezvisko
  FROM os_udaje;
```

Nie je možný INSERT.

Príklad 10.2: Jednoduchý pohľad-upravený

```
CREATE OR REPLACE VIEW pohl1a
AS
SELECT rod_cislo, meno, priezvisko
FROM os_udaje;
```

Už je možný INSERT, ak všetky atribúty sú NOT NULL.

Príklad 10.3: Pohľad s predefinovaním stĺpcov

```
CREATE OR REPLACE VIEW zoznam(meno, priezvisko, pohlavie) AS
  select ou.meno, ou.priezvisko,
         decode(substr(ou.rod_cislo,3,1),'5', 'zena', '6', 'zena','muz')
  from os_udaje ou;
```

ALEBO

```
CREATE OR REPLACE VIEW zoznam AS
  select ou.meno, ou.priezvisko,
         decode(substr(ou.rod_cislo,3,1),'5', 'zena', '6', 'zena','muz') as pohlavie
  from os_udaje ou;
```

10.3.2 Check option

Príklad 10.4: Pohľad s podmienkou

```
CREATE OR REPLACE VIEW  pohl2 AS
  select ou.meno, ou.priezvisko, ou.rod_cislo
  from os_udaje ou
  where ou.meno like 'K%'

WITH CHECK OPTION
```

Pozor! Je možné vložiť do pohľadu pohl2 aj údaje, ktoré pri selecte nebudete vidieť, ale dáta budú vložené do zdrojovej tabuľky

!

```
INSERT INTO pohl2 (meno, priezvisko,rod_cislo)
VALUES ('Martinko','Klingacik','0512224/0000');
```

10.3.3 Read only

Príklad 10.5: Zakázanie deštruktívnych operácií

```
CREATE OR REPLACE VIEW  pohl2 AS
  select ou.meno, ou.priezvisko, ou.rod_cislo
  from os_udaje ou
  where ou.meno like 'K%'
  WITH READ ONLY;
```

Príklad 10.6: Pohľad s použitím funkcií

```
CREATE OR REPLACE VIEW pohl3 (meno, priezvisko, priemer)
AS
SELECT ou.meno, ou.priezvisko,
       avg(decode (zp.vysledok,'A',1,'B',1.5,'C',2,'D',2.5,'E',3, 4) )
FROM os_udaje ou JOIN student st USING(rod_cislo)
     JOIN zap_predmety zp USING(os_cislo)
GROUP BY ou.meno, ou.priezvisko, st.os_cislo
WITH READ ONLY;
```

10.3.4 Pohľad a trigger

Príklad 10.7: Pohľad z viacerých tabuliek

```
CREATE OR REPLACE VIEW pohl4
AS
SELECT ou.meno, ou.priezvisko, st.rocnik,
       st.st_skupina, rod_cislo, st.os_cislo
FROM os_udaje ou JOIN student st USING(rod_cislo);

INSERT INTO pohl4 (meno, priezvisko, rocnik, skupina, rod_cislo, os_cislo)
VALUES ('Peter','Novy',1,'5ZI011','841231/1212',55);
```

! Tento insert nefunguje, aby fungoval je potrebné definovať trigger namiesto Insertu.

```
CREATE OR REPLACE TRIGGER pohl4_ins
   INSTEAD OF INSERT ON  pohl4
referencing new as novy
Begin
    insert into os_udaje (meno, priezvisko, rod_cislo)
    values (:novy.meno,:novy.priezvisko, :novy.rod_cislo);

    insert into student
     ( rod_cislo, os_cislo, st_skupina, rocnik, st_odbor, st_zameranie)
    values (:novy.rod_cislo,:novy.os_cislo,:novy.skupina,:novy.rocnik, 100, 0);

-- 100, 0 je to potrebne, aby boli dodržané pravidlá referenčnej integrity,
-- Informatika bez zamerania
END;
/
```

Príklad 10.8: Delete z pohľadu z viacerých tabuliek

```
DELETE FROM pohl4
WHERE os_cislo = 55;
```

Tento delete funguje 'záhadne' - z pohľadu síce riadok zmizne, ale v tabuľke `os_udaje` zostanú údaje o študentovi - Peter Novy.

```
CREATE OR REPLACE TRIGGER pohl4_del
  INSTEAD OF DELETE ON pohl4
  referencing old as stary
begin
  delete from student
  where os_cislo = :stary.os_cislo;

  delete from os_udaje
  where rod_cislo = :stary.rod_cislo;
end;
/
```

Príklad 10.9: Pohľad z pohľadu - DELETE

```
CREATE OR REPLACE VIEW pohl5
AS
SELECT meno, priezvisko, rod_cislo
FROM pohl4;
```

Insert nebude fungovať, lebo nemáte všetky potrebné údaje. Ale nasledovný DELETE vymaže nielen z tabuľky os_udaje, ako by sa zdalo, ale aj z tabuľky študent.

```
DELETE FROM pohl5
WHERE rod_cislo = '0512224/0000';
```

Príklad 10.10: Pohľad z pohľadu - CHECK OPTION

```
CREATE OR REPLACE VIEW pohl6 (meno, priezvisko,rod_cislo)
AS
SELECT ou.meno, ou.priezvisko, ou.rod_cislo
FROM os_udaje ou
where ou.meno like 'S%';

CREATE OR REPLACE VIEW pohl7
AS
SELECT * FROM pohl6
where rod_cislo like '79%'
WITH CHECK OPTION;
```

- Nasledovný insert funguje

  ```
  INSERT INTO pohl6
  VALUES ( 'Karol', 'Novy', '790502/1212');
  ```

- Tento insert nefunguje. Klausula WITH CHECK OPTION kontroluje aj zdedené podmienky.

  ```
  INSERT INTO pohl7
  VALUES ( 'Karol', 'Novy', '790502/1212');
  ```

10.3.5 Zhrnutie

- ak SELECT pohľadu obsahuje PK a všetky ostatné NOT NULL stĺpce tabuľky, potom je možné vykonať INSERT
- ak je pohľad definovaný ako READ ONLY, nie sú možné operácie INSERT, DELETE a UPDATE
- ak je pohľad definovaný ako WITH CHECK OPTION, potom do pohľadu je možné vložiť len riadky, ktoré zodpovedajú podmienkam SELECTu a zdedeným podmienkam
- ak je SELECT z viacerých tabuliek, pre operácie INSERT, DELETE a UPDATE je nutné definovať triggre INSTEAD OF