Šesta Nedelja
– SQL: DDL, DCL –

Autori: Mr. Miloš Cvetanović

DDL – Tipovi podataka

- **INTEGER**
- **FLOAT**
- **DECIMAL** [m [, n]]
- **BOOLEAN**
- **CHARACTER** [(n)]
- **CHARACTER VARYING** (n)
- **TIMESTAMP**
- **DATE**
- **TIME**

DDL – Kreiranje tabela

- **NaredbaKreiranjaTabela ::=**
  CREATE TABLE Tabela (DefinicijaKolone , …
  [OgranicenjeTabele , …]) ;
- **DefinicijaKolone ::=**
  Kolona { Tip | Domen } OgranicenjeKolone …
- **NOT NULL**
  UNIQUE
  PRIMARY KEY
  CHECK (Predikat)
  DEFAULT = Const
- **REFERENCES Tabela [(Kolona)]**
  [ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
  [ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

- **UNIQUE (Kolona , …)**
  PRIMARY KEY (Kolona , …)
  CHECK (Predikat)
- **FOREIGN KEY (Kolona , …)**
  REFERENCES Tabela [(Kolona , …)]
  [ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
  [ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

DDL – Primer

```
CREATE TABLE
        Oblast   (
                 SifO     CHAR(2)          PRIMARY KEY,
                 Naziv    CHAR(20)         NOT NULL
                                           UNIQUE
                 );
CREATE TABLE
        Knjiga   (
                 SifK     CHAR(3)          PRIMARY KEY,
                 SifN     CHAR(4)          NOT NULL
                                           REFERENCES Naslov
                                                    ON UPDATE CASCADE
                                                    ON DELETE  NO ACTION
                 );
CREATE TABLE
        Rezervacija  (
                        SifN          CHAR(4)     REFERENCES Naslov
                                                    ON UPDATE CASCADE
                                                    ON DELETE  CASCADE,
                        SifC          CHAR(3)     REFERENCES Clan
                                                    ON UPDATE CASCADE
                                                    ON DELETE  CASCADE,
                        Naziv         TIMESTAMP   NOT NULL,

                        PRIMARY KEY (SifN, SifC)

                        );
```

DDL – Opšti integritet

- **NaredbaKreiranjaPravila ::=**
  CREATE ASSERTION Pravilo CHECK (Ogranicenje) ;

- **NaredbaUklanjanjaPravila ::=**
  DROP ASSERTION Pravilo ;

- **CREATE ASSERTION Max3Rezervacije**
  CHECK (NOT EXISTS (SELECT SifC
  FROM Rezervacija
  GROUP BY SifC
  HAVING COUNT (*) > 3
)
  );

DDL – Izvršavanje ograničenja

- **NaredbaImenovanjaPravila ::=**
  CONSTRAINT Pravilo CHECK (Ogranicenje)
  [{ INITIALLY DEFERRED | INITIALLY IMMEDIATE }]
  [{ DEFERRABLE | NOT DEFERRABLE }] ;

- **NaredbaPostvljanjaModaPravila ::=**
  SET CONSTRAINT { Pravilo, ... | ALL }{ DEFERRED | IMMEDIATE }

DDL – Izmena tabela

- **NaredbaIzmeneTabele ::=**
  ALTER TABLE Tabela SpecIzmeneTabele ;
- **SpecIzmeneTabele ::=**
  SpecIzmeneKolona | SpecIzmeneOgranicenjaTabele
- **SpecIzmeneKolona ::= SpecIzmeneJedneKolone , …**
- **SpecIzmeneJedneKolone ::=**
  DodavanjeKolone | IzmenaKolone | UklanjanjeKolone
- **DodavanjeKolone ::= ADD [COLUMN] DefinicijaKolone**
- **IzmenaKolone ::=**
  DodavanjePodrazumevanja | UklanjanjePodrazumevanja
- **DodavanjePodrazumevanja ::=**
  ALTER [COLUMN] Kolona SET DEFAULT = Const
- **UklanjanjePodrazumevanja ::=**
  ALTER [COLUMN] Kolona DROP DEFAULT
- **UklanjanjeKolone ::=**
  DROP [COLUMN] Kolona { RESTRICT | CASCADE }

DDL – Izmena tabela (nastavak)

- **SpecIzmeneOgranicenjaTabele ::=**
  SpecIzmeneJednogOgranicenjaTabele , ...
- **SpecIzmeneJednogOgranicenjaTabele ::=**
  SpecDodavanjaOgranicenja | SpecUklanjanjaOgranicenja
- **SpecDodavanjaOgranicenja ::=**
  ADD CONSTRAINT OgranicenjeTabele
- **SpecUklanjanjaOgranicenja ::=**
  DROP CONSTRAINT { RESTRICT | CASCADE }

DDL – Uklanjanje tabela

- **NaredbaUklanjanjaTabele ::= DROP TABLE Tabela ;**

DDL – Indeksi

- **NaredbaKreiranjaIndeksa ::=**
  CREATE [UNIQUE] INDEX Indeks ON Tabela (Kolona , ...) ;
- **NaredbaUklanjanjaIndeksa ::=**
  DROP INDEX Indeks ;
- **Primer**
  CREATE INDEX Je_Autor1 ON Je_Autor (SifA);
  CREATE INDEX Je_Autor2 ON Je_Autor (SifN);

- **NaredbaKreiranjaPogleda ::=**
  CREATE VIEW Pogled [(Kolona , ...)] AS Upit ;
- **NaredbaUklanjanjaPogleda ::=**
  DROP VIEW Pogled ;
- **Prednosti upotrebe pogleda:**
  - Vrsta "podprograma" u SQL-u (WHERE, HAVING)
  - Pojednostavljenje komplikovanih i čestih upita
  - Rešenje problema viška podataka u svodnim upitima
  - Olakšano uspostavljanje kontrole pristupa
- **Ažurabilnost pogleda:**
  - Upit mora biti nad jednom tabelom
  - Upit sme da sadrži samo imena kolona
  - Upit ne sme da sadrži DISTINCT kaluzulu
  - Upit ne sme biti svodni
  - Ponekad: mora da ima PRIMARY KEY, i/ili da nema podupite

Primer – Veleprodajni lanac prodavnica

- **Data je šema relacione baze podataka veleprodajnog lanca prodavnica:**

  PRODAVNICA(SifP, Adresa, SifM);

  MESTO(SifM, Naziv);

  KLIJENT(SifK, Naziv, SifM);
  RACUN(SifR, SifK, SifP, SifRa, Datum);

  PROIZVOD(SifPr, Naziv, Cena);
  STAVKA_RACUNA(SifS, SifR, SifPr, RedniBr, Kolicina, Iznos);
  RADNIK(SifRa, Ime, SifP);

- **Sastaviti SQL skript kojim se formira tabela STAVKA_RACUNA, ukoliko je poznato da se stavke jednog računa moraju unositi redom (sukcesivni RedniBr) i da Iznos mora biti definisan količinom i cenom proizvoda na koji se posmatrana stavka odnosi. Jedan proizvod se na računu ne sme pojavljivati u više stavki.**

Primer – Veleprodajni lanac prodavnica

```
CREATE TABLE STAVKA_RACUNA
(    SifS    INT PRIMARY KEY,
     SifR    INT NOT NULL REFERENCES RACUN (SifR) ON UPDATE CASCADE,
     SifPr   INT NOT NULL REFERENCES PROIZVOD (SifPr) ON UPDATE CASCADE,
     Kolicina INT NOT NULL CHECK (Kolicina > 0),
     RedniBr INT NOT NULL CHECK (RedniBr=(SELECT COUNT(*)
                                          FROM STAVKA_RACUNA S
                                          WHERE S.SifR=STAVKA_RACUNA.SifR
                                          )
                                ),
     Iznos FLOAT NOT NULL,
     UNIQUE (SifR, SifPr),
     UNIQUE (SifR, RedniBr)
);
```

U vidu provere na nivou tabele:

```
CHECK ( STAVKA_RACUNA.RedniBr
                    = (SELECT COUNT(*)
                       FROM STAVKA_RACUNA S
                       WHERE S.SifR=STAVKA_RACUNA.SifR
                       )
      )                          ???
```

```
CREATE ASSERTION ProveraIznosa
CHECK (NOT EXISTS (SELECT *
                   FROM PROIZVOD P, STAVKA_RACUNA S
                   WHERE (P.SifPr=S.SifPr) AND ( S.Iznos <>(P.Cena*S.Kolicina))
                  )
      );
```

- **Data je šema relacione deo baze podataka fakulteta:**

  STUDENT (SifS, Ime, BrIndeksa);
  PROFESOR (SifP, Ime, SifO);
  ODSEK (SifO, Naziv);
  KURS (SifK, Naziv, BrKredita, SifO) UČIONICA (SifU, BrMesta);
  PREDUSLOV (SifK, SifKP) POHAĐA(SifS, SifR);
  RASPORED (SifR, SifP, SifK, SifU, Termin, Dan, Br.Prijavljenih);

- **Sastaviti SQL skript kojim se formira tabela RASPORED, ukoliko je poznato da samo jedan profesor može držati po rasporedu predavanje u jednom terminu u jednoj učionici. Pri tom broj prijavljenih mora biti manji od broja mesta u učionici koja je rasporedu predviđena za taj kurs, a takođe vrednost atributa termin mora biti celobrojna vrednost u opsegu od 1 do 7, a vrednost atributa dan iz skupa vrednosti {Pon, Uto, Sre, Cet, Pet}.**

```
CREATE TABEL RASPORED
(
    SifR    INT PRIMARY KEY,
    SifP    INT NOT NULL REFERENCES PROFESOR ON UPDATE CASCADE,
    SifK    INT NOT NULL REFERENCES KURS ON UPDATE CASCADE,
    SifU    INT NOT NULL REFERENCES UCIONICA ON UPDATE CASCADE,
    Termin INT NOT NULL CHECK (Termin BETWEEN 1 and 7),
    Dan CHAR(3) NOT NULL CHECK (Dan IN ('PON', 'UTO', 'SRE', 'CET', 'PET')),
    BrPrijavljenih INT NOT NULL,
    UNIQUE (SifP, Termin, Dan),
    UNIQUE (SifU, Termin, Dan)
);
```

U vidu provere na nivou tabele:

```
CHECK (NOT EXISTS (SELECT *
                   FROM UCIONICA U
                   WHERE U.SifU=RASPORED.SifU
                   AND U.BrMesta<RASPORED.BrPrijavljenih
                   )
      )
```

```
CREATE ASSERTION ProveraBrPrijavljenih
CHECK (NOT EXISTS (SELECT *
                   FROM UCIONICA U, RASPORED R
                   WHERE (U.SifU=R.SifU) AND ( R.BrPrijavljenih > (U.BrMesta))
                  )
      );
```

Primer – Fudbalski savez

- **Data je šema relacione baze podataka fubalskog saveza za potrebe evidencije utakmica jedne sezone (pretpostavka je da fudbaleri ne mogu da menjaju tim u kome igraju, u toku sezone):**

  FUDBALER (SifF, Ime, SifT);
  TIM (SifT, Naziv, Mesto);
  UTAKMICA (SifU, SifTDomaci, SifTGost, Kolo, Ishod, Godina);
  IGRAO (SifF, SifU, PozicijaIgraca);
  GOL (SifG, SifU, SifF, RedniBrGola, Minut);
  KARTON (SifK, SifU, SifF, Tip, Minut);

- **Sastaviti SQL skript kojim se formira tabela UTAKMICA, ukoliko je poznato da utakmicu igraju dva različita tima čije se šifre nalaze u tabeli TIM, da ishod utakmice može biti iz skupa vrednosti {X-nerešeno, 1-pobeda domaćih, 2-pobeda gostiju} i da postoji svega 30 kola u kojima se utakmice igraju. U toku sezone svaki par timova odigra dve utakmice, pri čemu je jedna na domaćem, a druga na gostujućem terenu.**

Primer – Fudbalski savez

```
CREATE TABLE UTAKMICA
(
    SifU    INT PRIMARY KEY,
    SifTDomaci INT NOT NULL REFERENCES TIM(SifT) ON UPDATE CASCADE,
    SifTGost INT NOT NULL REFERENCES TIM(SifT) ON UPDATE CASCADE,
    Kolo INT NOT NULL CHECK (Kolo BETWEEN 1 and 30),
    Ishod CHAR NOT NULL CHECK (Ishod IN ( '1', '2', 'X')),
    Godina INT,
    CHECK (SifTDomaci <> SifTGost),
    UNIQUE (SifTDomaci, SifTGost)
);
```

Primer – Skladište robe

- **Data je šema relacione baze podataka za potrebe skladišta robe u toku jedne godine:**

  ROBA(SifR, Naziv, Opis, SifD);

  DOBAVLJAC(SifD, Naziv, Adresa);
  NABAVKA(SifN, Datum, Kolicina, Cena, SifR);

- **Sastaviti SQL skript koji formira tabelu NABAVKA ukoliko je poznato da se datum predstavlja kao celobrojna vrednost u opsegu od 1 do 365, i da cena predstavlja jediničnu cenu koja ne sme biti skuplja za više od 10% od cene pri prethodnoj nabavci te robe. Takođe je poznato da jednog datuma može biti najviše jedna nabavka jedne vrste robe.**

Primer – Skladište robe

```
CREATE TABLE  NABAVKA
(
     SifN    INT PRIMARY KEY,
     Datum INT NOT NULL CHECK (Datum BETWEEN 1 and 365),
     Cena INT NOT NULL CHECK (Cena >0),
     Kolicina INT NOT NULL CHECK (Kolicina >0),
     SifR INT NOT NULL REFERENCES ROBA(SifR) ON UPDATE CASCADE,
     UNIQUE (SifR, Datum),
     CHECK (NOT EXISTS (SELECT *
                              FROM NABAVKA N1
                              WHERE N1.Cena <= 1.10 *(SELECT N.Cena
                                                            FROM NABAVKA N2
                                                            WHERE N2.SifR=N1.SifR
                                                            AND N2.Datum = (SELECT MAX(N3.Datum)
                                                                                 FROM Nabavka N3
                                                                                 WHERE N3.SifR=N1.SifR
                                                                                  AND N3.Datum<N1.Datum
                                                                                 )
                                                              )
                              )
               ),
);
```

Primer – Matrice

- **Data je šema relacione baze podataka, za potrebe izračunavanja pomoću matrica:**

  MATRICA(SifM, Naziv, BrVrsta, BrKolona);

  PODACI(SifP, I, J, Vrednost, SifM);

- **Sastaviti SQL skript koji formira tabelu PODACI ukoliko je poznato da svaka matrica mora imati za svaki element po tačno jednu celobrojnu vrednost.**

```
CREATE TABLE  PODACI
(
      SifP     INT PRIMARY KEY,
      I INT NOT NULL CHECK (I > 0),
      J INT NOT NULL CHECK (J > 0),
      Vrednost INT NOT NULL,
      SifM INT NOT NULL REFERENCES MATRICA(SifM) ON UPDATE CASCADE,
      UNIQUE (SifM, I, J),
      CHECK (NOT EXISTS (SELECT *
                              FROM PODACI N1
                              GROUP BY SifM
                              HAVING MAX (I) <> COUNT (I)
                              )
            ),
       CHECK (NOT EXISTS (SELECT *
                              FROM PODACI N1
                              GROUP BY SifM
                              HAVING MAX (J) <> COUNT (J)
                              )
            ),
);

CREATE ASSERTION Dimenzije
CHECK (NOT EXISTS (SELECT *
                        FROM PODACI P, MATRICA M
                        WHERE P.SifM=M.SifM
                        GROUP BY M.SifM, M.BrVrsta, M.BrKolona
                        HAVING (BrVrsta <> COUNT(P.I)) OR (BrKolona <> COUNT(P.J))
                      )
         );
```

DCL – Opšta prava

- **NaredbaKreiranjaKorisnika ::=**
  GRANT OpstePravo TO Korisnik IDENTIFIED BY Lozinka ;
- **CONNECT**
  RESOURCE
  DBA
- **NaredbaDodeleOpstegPrava ::=**
  GRANT OpstePravo TO Korisnik ;
- **NaredbaUklanjanjaOpstegPrava ::=**
  REVOKE OpstePravo FROM Korisnik ;

DCL – Posebna prava

- **NaredbaDodelePosebnogPrava ::=**
  GRANT { PosebnoPravo , ... } | ALL
  ON Tabela | Pogled
  TO { Korisnik , ... } | PUBLIC
  [WITH GRANT OPTION] ;
- **PosebnoPravo ::= PravoOcitavanja | PravoAzuriranja**
- **PravoOcitavanja ::= PravoUpita | PravoReferisanja**
- **PravoAzuriranja ::=**
  PravoUbacivanja | PravoIzmene | PravoBrisanja
- **PravoUpita ::= SELECT [(Kolona , ...)]**
- **PravoReferisanja ::= REFERENCE (Kolona , ...)**
- **PravoUbacivanja ::= INSERT [(Kolona , ...)]**
- **PravoIzmene ::= UPDATE [(Kolona , ...)]**
- **PravoBrisanja ::= DELETE**

- **NaredbaUklanjanjaPosebnogPrava ::=**
  REVOKE { PosebnoPravo , ... } | ALL
  ON Tabela | Pogled
  FROM { Korisnik , ... } | PUBLIC ;

- **GRANT CONNECT**
  TO Clan IDENTIFIED BY LozinkaClana ;
- **GRANT CONNECT**
  TO Sef IDENTIFIED BY LozinkaSefa ;

- **GRANT SELECT**
  ON Oblast, Naslov, Autor, Je_Autor
  TO PUBLIC ;
- **GRANT INSERT, UPDATE, DELETE**
  ON Oblast, Naslov, Autor, Je_Autor
  TO Sef ;

- **GRANT DELETE ON Clan TO Sef ;**

- **Ograničavanje pristupa pojedinim redovima**
- **Ograničavanje pristupa pojedinim kolonama**
- **Ograničavanje pristupa na svodne rezultate**

- **Pretpostavke:**
  - Pravo nad pogledom, a ne tabelom
  - Pravo nad pogledom važi i nad tabelom za vreme definicionog upita

- **Primer**

```
CREATE VIEW NaslovPJ
AS    SELECT *
      FROM Naslov
      WHERE SifO = 'PJ' ;

GRANT SELECT ON NaslovPJ TO ClanPJ ;
```