## CREATE DATABASE

**Primer 1.**

```
CREATE DATABASE STUDIJE
ON PRIMARY
(NAME = PODACI1,
 FILENAME = 'D:\ispit\primarna.mdf',
 SIZE = 5,
 MAXSIZE = 20,
 FILEGROWTH = 5),
FILEGROUP GRUPA2
(NAME = PODACI2,
 FILENAME = 'D:\ispit\sekundarna1.ndf',
 SIZE = 5,
 MAXSIZE = 25,
 FILEGROWTH = 5)
LOG ON
(NAME = fakultet_log,
FILENAME = 'D:\ispit\dnevnik\fak_dnevnik.ldf',
 SIZE = 5,
 MAXSIZE = 20,
 FILEGROWTH = 5)
```

**Primer 2.**

```
ALTER DATABASE STUDIJE
ADD FILE
(
    NAME = PODACI3,
    FILENAME = 'D:\ispit\sekundarna2.ndf',
    SIZE = 5MB,
    MAXSIZE = 50MB,
    FILEGROWTH = 5MB
)
TO FILEGROUP GRUPA2
```

**Primer 3.**

```
ALTER DATABASE STUDIJE
REMOVE FILE PODACI3
```

**Primer 4.**

```
CREATE LOGIN SavicMilos
    WITH PASSWORD = 'Ailo3ksz';
```

**Primer 5.**

```
use studije
go
CREATE SCHEMA NastOdr
CREATE TABLE NastOdr.Nastavnici (
Snast int not null,
Imen char(25)
)
```

**Primer 6.**

```
USE Studije;
CREATE USER Savic FOR LOGIN SavicMilos
    WITH DEFAULT_SCHEMA = NastOdr;
GO
```

**Primer 7.**

```
use [STUDIJE]
GO
GRANT TAKE OWNERSHIP ON SCHEMA::[NastOdr] TO [Savic]
GO
```

**Primer 8.**

```
CREATE TABLE Predmeti
   (Spred SMALLINT PRIMARY KEY,
    Nazivp CHAR(25) NOT NULL);
```

**Primer 9.**

```
ALTER TABLE NastOdr.Nastavnici
ADD CONSTRAINT pk_nast PRIMARY KEY CLUSTERED(Snast);
```

**Primer 10.**

```
CREATE TABLE Smer (
   Ssmer SMALLINT PRIMARY KEY NONCLUSTERED,
   Nazivs char(30) UNIQUE);
```

**Primer 11.**

```
CREATE TABLE Studenti
  (Indeks SMALLINT,
   Upisan SMALLINT DEFAULT(2000),
   Imes CHAR(25),
   Mesto CHAR(30),
   Datr DATETIME,
   Ssmer SMALLINT REFERENCES Smer(Ssmer),
   CONSTRAINT pk_stud PRIMARY KEY CLUSTERED(Indeks,Upisan));
```

**Primer 12.**

```
CREATE TABLE Planst (
   Ssmer SMALLINT REFERENCES Smer(Ssmer),
   Spred SMALLINT REFERENCES Predmeti(Spred),
   Semestar TINYINT,
 CHECK(Semestar IN ('1','2','3','4','5','6','7','8','9','10')));
```

**Primer 13.**

```
CREATE TABLE Angazovanje (
   Snast INT REFERENCES Nastavnici(Snast),
   Spred SMALLINT REFERENCES Predmeti(Spred),
   Ssmer SMALLINT REFERENCES Smer(Ssmer));
```

**Primer 14*.**

```
ALTER TABLE Smer ALTER COLUMN Nazivs CHAR(40) NOT NULL;
ALTER TABLE Smer NOCHECK CONSTRAINT SMER;
GO
ALTER TABLE Smer ALTER COLUMN Nazivs CHAR(40) NOT NULL;
GO
```

```
ALTER TABLE Smer CHECK CONSTRAINT SMER;
GO
```

**Primer 15.**

```
CREATE TABLE [dbo].[Prijave] (
	[Spred] [smallint] NOT NULL ,
	[Indeks] [smallint] NOT NULL ,
	[Upisan] [smallint] NOT NULL ,
	[Snast] [smallint] NULL ,
	[Datump] [datetime] NOT NULL ,
	[Ocena] [smallint] NULL
) ON [PRIMARY]
GO
```

**Primer 16.**

```
ALTER TABLE Prijave ADD CONSTRAINT tek_dat
DEFAULT GETDATE() FOR Datump;
```

**Primer 17.**

```
CREATE TYPE SSN
FROM varchar(11) NOT NULL ;
```

---

**Primer 18.**

```
CREATE TYPE IZNOS FROM SMALLMONEY
```

**Primer 19.**

```
CREATE TYPE IZNOS FROM SMALLMONEY
CREATE TABLE Uplate (
	Indeks SMALLINT,
	Upisan SMALLINT,
	Uplata  IZNOS)
INSERT INTO Uplate VALUES(1,2000,120.00)
```

```
CREATE TYPE IZNOS FROM SMALLMONEY
go
CREATE TABLE Uplate (
Indeks SMALLINT,
Upisan SMALLINT,
Uplata  IZNOS)
INSERT INTO Uplate
VALUES(1,2000,120.00)
go
```

---

**Primer 20.**

```
CREATE INDEX stud_ind ON Studenti(Imes,Mesto)
WITH PAD_INDEX, FILLFACTOR = 80
ON grupa2
```

**Primer 21.**

```
CREATE INDEX ocene_ind ON Prijave(Spred, Indeks, Upisan, Datump)
ON grupa2
```

**Primer 22.**

```
CREATE TABLE [dbo].[DIPLOMIRANI](
	[Indeks] [int] NULL,
	[Upisan] [int] NULL,
	[Ssmer] [int] NULL
) ON [PRIMARY]
```

**Primer 23.**

```
CREATE INDEX dipl_ind ON Diplomirani(Indeks, Upisan)
```

**Primer 24.**

```
ALTER INDEX ocene_ind ON Prijave
```

```
REBUILD WITH (FILLFACTOR = 80, SORT_IN_TEMPDB = ON,
              STATISTICS_NORECOMPUTE = ON);
```

**Primer 25.**

```
ALTER INDEX dipl_ind ON Diplomirani REBUILD
```

**Primer 26.**

```
ALTER INDEX ocene_ind ON Prijave
REBUILD WITH (FILLFACTOR = 80, SORT_IN_TEMPDB = ON,
              STATISTICS_NORECOMPUTE = ON);
```

**Primer 27.**

```
CREATE UNIQUE INDEX ANG_IND ON ANGAZOVANJE(Snast, Spred)
```

**Primer*** `INSERT INTO ANGAZOVANJE VALUES (1,1,NULL)`

**Primer 28.**

```
CREATE UNIQUE CLUSTERED INDEX planst_ind
ON Planst(Spred, Ssmer)
WITH PAD_INDEX, FILLFACTOR = 80
```

**Primer 29.**

```
CREATE TABLE [dbo].[USLOVNI](
      [Spred] [smallint] NULL,
      [UslPredmet] [smallint] NULL
) ON [PRIMARY]
go
ALTER TABLE [dbo].[USLOVNI] WITH CHECK ADD FOREIGN KEY([Spred])
REFERENCES [dbo].[PREDMETI] ([SPRED])
```

**Primer 30.**

```
CREATE SCHEMA predm_sch
CREATE VIEW predm_sch.uslovni_pogl
AS
SELECT Predmeti.Spred,Predmeti.Nazivp FROM dbo.Predmeti
JOIN dbo.Uslovni ON dbo.Predmeti.Spred = dbo.Uslovni.Spred
```

**Primer ***

```
CREATE UNIQUE CLUSTERED INDEX pogled_ind ON predm_sch.uslovni_pogl(Spred)
```

**Primer 31.**

```
ALTER VIEW predm_sch.uslovni_pogl
WITH SCHEMABINDING
AS
SELECT Predmeti.Spred,Predmeti.Nazivp FROM dbo.Predmeti
JOIN dbo.Uslovni ON dbo.Predmeti.Spred = dbo.Uslovni.Spred
```

**Primer 32.**

```
CREATE UNIQUE CLUSTERED INDEX pogled_ind ON predm_sch.uslovni_pogl(Spred)
```

**Primer **** `drop INDEX planst_ind  ON Planst.`

# OGRANIČENJA

### Primer 33.

```
CREATE TABLE Prodavci
    (prod_id        CHAR(4) NOT NULL PRIMARY KEY,
     prod_ime       VARCHAR(40),
     prod_adresa1   VARCHAR(40),
     prod_adresa2   VARCHAR(40),
     grad           VARCHAR(20),
     država               CHAR(2),
     poštanski_broj       CHAR(5),
     telefon              CHAR(12),
     prodaja_rep          INT);
```

```
drop table prodavci;
```

### Primer 34.

```
CREATE TABLE Prodavci
    (prod_id        CHAR(4)     NOT NULL,
     prod_ime       VARCHAR(40),
     prod_adresa1   VARCHAR(40),
     prod_adresa2   VARCHAR(40),
     grad           VARCHAR(20),
     država               CHAR(2),
     poštanski_broj       CHAR(5),
     telefon              CHAR(12),
     prodaja_rep          INT,
CONSTRAINT     pk_prod_id  PRIMARY KEY (prod_id));
```

```
drop table prodavci;
```

### Primer 35.

```
create table zaposleni (
       zap_id int not null primary key,
       ime varchar(40)
   );
CREATE TABLE Prodavci
    (prod_id             CHAR(4)    PRIMARY KEY,
     prod_ime            VARCHAR(40),
     prod_adresa1        VARCHAR(40),
     prod_adresa2        VARCHAR(40),
     grad                VARCHAR(20),
     država              CHAR(2),
     poštanski_broj      CHAR(5)    UNIQUE,
     telefon             CHAR(12),
     prodaja_rep         INT NOT NULL REFERENCES Zaposleni(zap_id));
```

```
drop table prodavci;
```

### Primer 36.

```
CREATE TABLE Prodavci
    (prod_id        CHAR(4)    NOT NULL,
     prod_ime       VARCHAR(40),
     prod_adresa1   VARCHAR(40),
     prod_adresa2   VARCHAR(40),
     grad           VARCHAR(20),
     država         CHAR(2)    ,
     poštanski_broj       CHAR(5)    ,
     telefon              CHAR(12)   ,
     prodaja_rep          INT        ,
```

```
CONSTRAINT pk_prod_id  PRIMARY KEY (prod_id),
CONSTRAINT fk_zap_id   FOREIGN KEY (prodaja_rep)
                      REFERENCES Zaposleni(zap_id),
CONSTRAINT unq_zip     UNIQUE (poštanski_broj) );
```

```
drop table prodavci;
```

**Primer 37.**

```
CREATE TABLE Prodavci
   (prod_id      CHAR(4)
      CONSTRAINT pk_prod_id PRIMARY KEY
      CONSTRAINT ck_prod_id CHECK
         (prod_id LIKE '[A-Z][A-Z][A-Z][A-Z]' OR
          prod_id LIKE '[A-Z][A-Z][0-9][0-9]'),
    prod_ime     VARCHAR(40),
    prod_adresa1 VARCHAR(40),
    prod_adresa2 VARCHAR(40),
    grad         VARCHAR(20),
    država       CHAR(2)
      CONSTRAINT def_st DEFAULT ('SR'),
    Poštanski_broj    CHAR(5)
      CONSTRAINT unq_prod_zip UNIQUE
      CONSTRAINT ck_prod_zip CHECK
         (Poštanski_broj LIKE '[0-9][0-9][0-9][0-9][0-9]'),
    telefon      CHAR(12),
    prodaja_rep   INT NOT NULL
   DEFAULT USER REFERENCES Zaposleni(zap_id))
```

```
drop table prodavci;
```

# POGLEDI

```
CREATE VIEW ime_pogleda {[(column [,...])] |
AS
select_naredba
```

Klauzula SELECT pogleda u SQL Serveru ne može:
- Da ima klauzule COMPUTE, COMPUTE BY, INTO ili ORDER BY (ORDER BY je omogućena ako se koristi SELECT TOP)
- Pozive privremene tabele
- Pozive promenljive tabele
- Pozive više od 1 024 kolona, uključujući one koje su pozvane podupitima

SQL Server omogućava više SELECT naredbi u pogledu, dok god su oni povezani klauzulama UNION ili UNION ALL. SQL Server takođe omogućava funkcije u naredbi SELECT pogleda. SQL Server pogled omogućava ažuriranje ako su sve stavke u sledećoj listi tačne:
- Naredba SELECT nema agregatne funkcije
- Naredba SELECT ne sadrži TOP, GROUP BY, DISTINCT ili UNION
- Naredba SELECT nema izvedene kolone
- Klauzula FROM u naredbi SELECT poziva najmanje jednu tabelu

SQL Server omogućava kreiranje indeksa na pogledima (vidi CREATE INDEX). Pri kreiranju jedinstvenog, klasterovanog indeksa na pogledu, prouzrokuje da SQL Server smesti fizičku kopiju pogleda na bazi podataka. Promene u tabelama baze se automatski ažuriraju u indeksiranom pogledu. Indeksirani pogledi troše više prostora na disku, ali obezbeđuju podizanje performansi. Indeksirani pogledi moraju da se izgrade korišćenjem klauzule SCHEMABINDING.
http://www.microsoft.com/technet/prodtechnol/sql/2005/impprfiv.mspx

Prema ANSI standardu preko pogleda može biti ažurirana bazna tabela(e) na kojima se zasniva pogled ako ispunjava sledeće uslove:
- Pogled ne sadrži operatore UNION, EXCEPT ili INTERSECT
- Definisana naredba SELECT ne sadrži klauzule GROUP BY ili HAVING
- Definisana naredba SELECT ne sadrži bilo kakvo pozivanje ne-ANSI pseudokolona kao što su ROWNUM or ROWGUIDCOL
- Definisana naredba SELECT ne sadrži klauzulu DISTINCT
- Pogled nije materijalizovan ()

Najznačajnije pravilo koga se treba setiti kada se ažurira bazna tabela preko pogleda je da sve kolone u tabeli koje su definisane kao NOT NULL moraju da prime ne nula vrednost kada dobijaju novu ili menjaju vrednost. Ovo može da se čini eksplicitno direktnim upisivanjem ili ažuriranjem not null vrednosti u koloni ili oslanjanjem na podrazumevanu vrednost. Osim toga, pogledi ne skidaju ograničenja na osnovnoj tabeli. Tako, vrednosti koje se upisuju ili ažutiraju u osnovnoj tabeli moraju da ispunjavaju sva ograničenja postavljena na njima sa jedinstvenim indeksima, primarnim ključevima, CHECK ograničenjima i td.

- ```
  CREATE VIEW Informatika AS
  SELECT Indeks, Upisan, Imes FROM Studenti
  WHERE Ssmer IN (SELECT Ssmer FROM Smer
  WHERE Nazivs IN('Informatika'));

  SELECT * FROM Informatika;
  ```

- ```
  CREATE VIEW dbo.prosecne_ocene
  WITH SCHEMABINDING
  AS
  SELECT indeks, upisan, round(avg(ocena*1.0),2) AS pros_ocena,
  count(*) AS broj_ocena FROM dbo.prijave
  GROUP BY indeks, upisan
  ```

Klauzula SCHEMABINDING povezuje šemu tabele (tabela) sa objektom, u ovom primeru sa pogledom. Posledica je da se ne može menjati definicija tabela koje učestvuju u pogledu na način koji bi imao uticaja na definiciju pogleda. Prvo mora da se promeni definicija pogleda (ili pogled izbriše), a zatim definicija tabela(e).

- ```
  CREATE VIEW pogled_stud AS
  SELECT Imes, Mesto, Datr, Ssmer FROM Studenti

  INSERT INTO pogled_stud VALUES ('Mihajlo','jagodina',null,1)
  UPDATE pogled_stud
  SET Ssmer = 2
  WHERE Imes = 'Stevan' AND Mesto = 'Paraćin'
  ```

**USER i ROLE - sigurnost**
  ```

## UPITI NAD BAZOM

### SELECT FROM

```
select distinct
select a as blabla
select *, null
nove kolone/atributi
```

### SELECT FROM WHERE

```
=, <, >, !=
is null
and, not, or
between
like
in
```

value [NOT] IN ({comp_value1, comp_value2 [,...] | subquery})

- SELECT * FROM Predmeti WHERE Spred NOT IN (1,2,6,7,14),

### SELECT FROM WHERE ORDER BY

### SELECT F-JA FROM WHERE

- Agregatne f-je
  **AVG i SUM**
  **COUNT**
  **MIN and MAX**

### SELECT FROM WHERE GROUP BY HAVING

### OSTALE F-JE

- Skalarne f-je

**CASE**

CASE *ulazna_vrednost*
WHEN *when_uslov* THEN *rezultujuća_vrednost*[...n]
[ELSE *else_rezultujuća_vrednost*]
END

- SELECT Indeks, Upisan, Spred, Ocena,
  položio = CASE Ocena
  WHEN 5 THEN 'nije položio'
  ELSE 'položio'
  END
  FROM Prijave

**CAST i CONVERT**

CAST(izraz AS tip_podataka [(length)])

- SELECT CAST(123.45 AS INT)
- SELECT CAST(123.45 AS DECIMAL(10,1))
- SELECT Spred,
  CAST(CAST(SUM(Ocena) AS DECIMAL(5,2)) / CAST(COUNT(*) as DECIMAL(5,2)) AS DECIMAL(5,2)) 'Srednja ocena iz predmeta'
  FROM Prijave WHERE Spred = 7
  GROUP BY Spred

| From | To | Behavior |
|---|---|---|
| numeric | numeric | Round |
| numeric | int | Truncate |
| numeric | money | Round |
| money | int | Round |
| money | numeric | Round |
| float | int | Truncate |
| float | numeric | Round |
| float | datetime | Round |
| datetime | int | Round |

- SELECT Indeks, Upisan, AVG(CONVERT(NUMERIC(4,2), Ocena)) AS 'Srednja ocena'
  FROM Prijave
  WHERE CONVERT(NUMERIC(5,2), Ocena)> 7.34
  GROUP BY Upisan, Indeks

- Numeričke skalarne funkcije

**ABS**

- SELECT ABS (-1) AS 'Apsolutna vrednost broja'

**DATALENGTH, LEN**

`datalength` - Returns the number of bytes used to represent any expression.
`len` - Returns the number of characters of the specified string expression, excluding trailing blanks.

- select len(cast('a' as nvarchar(5)))
- select datalength(cast('a' as nvarchar(5)))
- select datalength(cast('a' as varchar(5)))
- select datalength(cast(1 as int))
- select len(cast(1 as int))
- select ceiling(-1.2)

**CEIL**

- select ceiling(1.2)

**EXP, LOG, LOG10**

- SELECT EXP(1) AS 'Rezultat funkcije EXP'
- SELECT EXP( LOG(10)) as 'Rezultat EXP', LOG( EXP(10)) AS 'Rezultat LOG'
- SELECT LOG10(100)

**FLOOR**

- SELECT FLOOR (101.1) AS 'Rezultat funkcije FLOOR'
- SELECT FLOOR(-100.1)

% - deljenje po modulu

- SELECT SUM(Ocena) / COUNT(Ocena) AS 'Srednja ocena',
  SUM(Ocena) % COUNT(Ocena) AS 'Ostatak deljenja'FROM Prijave

**POWER**

- `SELECT POWER(10,3) as 'Stepen'`

**SQRT**

**Izdvajanje podataka iz datuma**

- `SELECT DATEPART(YEAR,'2013-07-02') AS 'Godina';`

`year, quarter, dayofyear, day, week, weekday, hour, minute, second, millisecond`

- `SELECT DATEPART(DAY, GETDATE())AS 'Dan u mesecu'`

**CHARINDEX**

- `SELECT CHARINDEX( 'de', 'abcdefg' )  Pozicija`

- Nizovne funkcije i operatori

**Operator povezivanja**

- ```
  SELECT Imes + ' iz ' + Mesto STUDENTI
  FROM STUDENTI
  WHERE Mesto IS NOT NULL ORDER BY Mesto;
  ```

**ROUND**

- `SELECT ROUND(12345.6789, 2)`

**LOWER i UPPER**

- `SELECT LOWER('ABcdE F1&') AS 'mala slova'`
- `SELECT UPPER('ABcdE F1&') AS 'VELIKA SLOVA'`

**SUBSTRING**

- ```
  SELECT Imes, SUBSTRING(Mesto,1,4) FROM Studenti WHERE
  SUBSTRING(Mesto,1,3) LIKE '%k%'
  ```

**LTRIM i RTRIM**

- `select LTRIM('  dsjh  ')`
- ```
  SELECT 'Student' + ' ' + RTRIM(CONVERT(CHAR(3),
  Prijave.Indeks)) + '/' +
  CONVERT(CHAR(4), Prijave.Upisan) + ' ima srednju ocenu ' +
  CONVERT(CHAR, AVG(CAST(Ocena AS NUMERIC(4,2))))
  FROM Prijave
  GROUP BY Prijave.Upisan, Prijave.Indeks
  ```

## POZIVANJE PROŠIRENIH IZVORA PODATAKA

- Operator `IN`
- Operator `EXISTS`

Operator EXIST je usmeren samo na određivanje da li pod upit vraća vrstu(e) ili ne. Ako vraća sračunava se u TRUE, inače u FALSE.

`WHERE [NOT] EXISTS (`*`subquery`*`)`

Upit

```
SELECT *
FROM Nastavnici  n
WHERE snast NOT IN (SELECT Snast
                    FROM Angazovanje a)
```

daje spisak nastavnika koji nisu angažovani.

Pokušaj dobijanja istih podataka korišćenjem operatora `EXISTS` sledećim upitom

```
SELECT *
FROM Nastavnici  n
WHERE NOT EXISTS (SELECT Snast FROM Angazovanje a)
```
```

neće biti uspešan, ali sa malom korekcijom hoće:

```
SELECT *
FROM Nastavnici  n
WHERE NOT EXISTS (SELECT snast
                  FROM Angazovanje a
                  WHERE n.Snast = a.Snast)
```

- Kvantifikovani operatori poređenja (quantified comparison predicate)
  `WHERE expression comparison {ALL | ANY | SOME} ( subquery )`
  - Spisak nastavnika kod kojih su polagali studenti sa smera 3

```
SELECT *
FROM Nastavnici
WHERE Snast = ANY (SELECT DISTINCT Snast
                   FROM Prijave, Studenti
                   WHERE Prijave.Indeks = Studenti.Indeks AND
                         Prijave.Upisan = Studenti.Upisan AND
                         Ssmer = 3);
```

  - Spisak studenata koji su prijavili bar jedan ispit
    Sledeći upit skoro pa daje tražene podatke

```
SELECT *
FROM Studenti
WHERE Studenti.Indeks = ANY (SELECT Prijave.Indeks
                             FROM Prijave, Studenti
                             WHERE Studenti.Indeks = Prijave.Indeks
                             AND Studenti.Upisan = Prijave.Upisan);
```

    **GDE JE GREŠKA?**

  - Spisak studenata čije je ime duže od prosečne dužine imena svih studenata

```
SELECT *
FROM Studenti
WHERE LEN(Imes) > ANY (SELECT AVG(LEN(Imes)) FROM Studenti);
```

  - Broj indeksa, godina upisa, ocena studenata čije su ocene manje od svih ocena koje je dao nastavnik sa ”s” u imenu

```
SELECT Indeks, Upisan, Ocena
FROM Prijave
WHERE Ocena < ALL (SELECT Ocena
                   FROM Prijave
                   WHERE Snast = ANY (SELECT Snast
                                      FROM Nastavnici
                                      WHERE Imen LIKE'%s%'))
```

## SKUPOVNI OPERATORI

**UNION, INTERSECT i EXCEPT**

```
<SELECT statement1>
UNION [ALL | DISTINCT]
<SELECT statement2>
```

- ```
  SELECT Indeks, Upisan, Imes, Ssmer
  FROM Studenti
  WHERE Ssmer  = 1 AND Imes LIKE '%n%'
  UNION
  SELECT Indeks, Upisan, Imes, Ssmer
  FROM Studenti
  WHERE Ssmer  = 2 AND  Imes LIKE '%n%'
  ORDER BY Imes
  ```

- ```
  SELECT Indeks, Upisan FROM Studenti
  EXCEPT
  SELECT Indeks, Upisan FROM Prijave;
  ```

# Upiti nad vise tabela

### SPAJANJA U KALUZULI WHERE

- ```
  SELECT n.Snast, Imen, Spred
  FROM Nastavnici AS n, Angazovanje AS a
  WHERE n.Snast = a.Snast
  ORDER BY Imen, Spred
  ```

### PODKLAUZULA JOIN

```
FROM table [AS alias] [ { INNER | { { LEFT | RIGHT | FULL } [ OUTER ] } }
    JOIN joined_table [AS alias]
   { ON join_condition1 [{AND|OR} join_condition2] [...] ]}
[...]
```

CROSS JOIN
Specificira kompletan Dekartov proizvod dveju tabela.
[INNER] JOIN
Specificira da neuparene vrste u bilo kojoj tabeli spajanja bi tzrebalo da budu odbačene. Ako tip spajanja nije eksplicitno definisan u ANSI stilu, ovo se podrazumeva.
LEFT [OUTER] JOIN
Specificira da će biti vraćeni svi rekordi iz tebele na levoj strani naredbe spajanja. Ako rekord vraćen iz leve tabele nema upareni rekord u tabeli na desnoj strani spajanja, one je još vraćen. Kolone iz desne tabele vraćaju vrednost NULL kada nema uparenu vrstu. To je dobra ideja za konfigurisanje (uobličavanje) svih spoljašnjih spajanja kao leva spoljašnja spajanja, radije nego mešanje levih i desnih spoljašnjih spajanja, gde god moguće radi konzistentnosti.
RIGHT [OUTER] JOIN
Specifies that all records be returned from the table on the right side of the join statement, even if the table on the left has no matching record. Columns from the left table return NULL values when there is no matching row.
FULL [OUTER] JOIN
Specificira da će biti vraćene sve vrste iz obeju tabela, nezavisno da li je vrsta iz jedne tabele uparena sa vrstom iz druge tabele. Bilo koje kolone koje nemaju vrednost u odgovarajućoj spojenoj tabeli dodeljuje im se NULL vrednost.
UNION JOIN
Specificira da će biti vraćene sve kolone obeju tabela i svaka vrsta iz obe tabele. Bilo koje kolone koje nemaju vrednost u odgovarajućoj spojenoj tabeli dodeljuje im se NULL vrednost.
ON join_condition
Spaja zajedno vrste iz tabele prikazane u kaluzuli FROM sa vrstama tabele koja je dekalrisana u kaluzuli JOIN. Mogu da postoje višestruke naredbe JOIN, sve bazirane na zajedničkom skupu vrednosti. Ove vrednosti su obično sadržane u kolonama sa istim imenom i tipom podataka koji se pojavljuju u obe tabele koje se spajaju.  Ove kolone, ili moguće jedna kolona iz svake tabele, se nazivaju ključ spajanja ili zajednički ključ.
```

Većinom ali ne uvek, ključ spajanja je primarni ključ u jednoj tabeli i spoljašnji ključ u drugoj tabeli. Spajanje može da se izvrši ukoliko se podaci u kolonama uparuju.

- Spisak nastavnika i sifara predmeta koje predaju

```
SELECT * FROM Nastavnici CROSS JOIN Predmeti
```

ili

```
SELECT * FROM Nastavnici, Predmeti
```

- Spisak nastavnika i sifara predmeta koje predaju

```
SELECT Angazovanje.Snast, Imen, Spred
FROM Angazovanje JOIN Nastavnici ON
     Angazovanje.Snast = Nastavnici.Snast
```

ili

```
SELECT a.Snast, Imen, Spred FROM Angazovanje AS a
JOIN Nastavnici AS n ON a.Snast = n.Snast
```

- Self-join

**Uraditi primer sa matičnom knjigom!**

```
SELECT Prvi.Imes, Prvi.Indeks, Prvi.Upisan, Prvi.Ssmer,
       Drugi.Imes, Drugi.Indeks, Drugi.Upisan, Drugi.Ssmer
FROM Studenti Drugi
JOIN Studenti Prvi ON Prvi.Indeks = Drugi.Indeks AND
      Prvi.Upisan > Drugi.Upisan
      ORDER BY Drugi.Indeks;
```

- Spisak studenata sa svim prijavama, ako su prijavljivali

```
SELECT Studenti.Indeks, Studenti.Upisan, Ssmer, Spred, Ocena
FROM Studenti LEFT OUTER JOIN Prijave ON
      Studenti.Indeks = Prijave.Indeks AND
      Studenti.Upisan = Prijave.Upisan
```

- Spisak studenata upisanih posle 2002 sa svim prijavama, ako su prijavljivali

```
SELECT Studenti.Indeks, Studenti.Upisan, Ssmer, Spred, Ocena
FROM Studenti LEFT OUTER JOIN Prijave ON
      Studenti.Indeks = Prijave.Indeks AND
      Studenti.Upisan = Prijave.Upisan
      AND Studenti.upisan > 2002
```

- Spisak imena nastavnika i naziva predmeta koje predaju

```
SELECT Imen, Nazivp
FROM Nastavnici JOIN Angazovanje ON
     Nastavnici.Snast = Angazovanje.Snast
     JOIN Predmeti ON Predmeti.Spred = Angazovanje.Spred
ORDER BY Imen
```

- Spisak imena studenata koji imaju bar jednu desetku

```
SELECT DISTINCT Imes
FROM Studenti JOIN Prijave ON
     Studenti.Indeks = Prijave.Indeks AND
```

```
        Studenti.Upisan = Prijave.Upisan
AND Ocena = 10
```

Uradite primer bez korišćenja `JOIN` klauzule

- Prikazati predmete i ocene studenata na smerovima Informatika i Profesor matematike i informatike i čije su ocene manje od 8

```
SELECT Imes, Nazivp, Ocena
FROM Prijave JOIN Studenti ON
        Studenti.Indeks = Prijave.Indeks AND
        Studenti.Upisan = Prijave.Upisan
     JOIN Predmeti ON
        Predmeti.Spred = Prijave.Spred
     JOIN Smer ON
        Studenti.Ssmer = Smer.Ssmer AND
        Nazivs IN ('Informatika',
                   'Profesor matematike i informatike') AND
        Ocena < 8
ORDER BY Imes
```

ili

```
SELECT Imes, Nazivp, Ocena
FROM Prijave, Studenti, Predmeti, Smer
WHERE   Studenti.Indeks = Prijave.Indeks AND
        Studenti.Upisan = Prijave.Upisan AND
        Predmeti.Spred = Prijave.Spred AND
        Studenti.Ssmer = Smer.Ssmer AND
        Nazivs IN( 'Informatika',
                   'Profesor matematike i informatike') AND
        Ocena < 8
ORDER BY Imes
```

**UPITI U SELECT DELU I UPITI U FROM DELU**

```
SELECT column1, column2, ... (scalar subquery)
FROM table1, ... (nested table subquery)
   AS subquery_table_name]
WHERE foo = (scalar subquery)
   OR foo IN (table subquery)
```

- 
```
SELECT DISTINCT Indeks, Upisan,
                    ( SELECT AVG(Ocena)
                      FROM Prijave p1
                      WHERE p1.Indeks = p2.Indeks AND
                             p1.Upisan = p2.Upisan) AS
                     'Prosečna ocena'
  FROM Prijave p2
```

- 
```
SELECT Nazivp, AVG(Ocena) AS 'Prosečna ocena'
FROM (SELECT Nazivp, Ocena
```

```
        FROM Prijave, Predmeti
      WHERE Prijave.Spred = Predmeti.Spred) AS Predmeti_ocene
GROUP BY Nazivp
```

- ```
  SELECT Indeks, Upisan, Mesto, Imes
  FROM Studenti s
  WHERE Indeks > (SELECT AVG(Indeks)
                  FROM Studenti
                  WHERE Mesto = s.Mesto)
  ```

## IZMENA PODATAKA U BAZI PODATAKA

INSERT

```
INSERT INTO [ONLY] {table_name | view_name} [(column1 [,...] )]
[OVERRIDE {SYSTEM | USER} VALUES]
{DEFAULT VALUES | VALUES (value1 [,...]) | SELECT_statement }
```

- ```
  INSERT INTO Studenti VALUES (1,2000,'Stevan',NULL,NULL,NULL)
  ```
- ```
  INSERT INTO Studenti VALUES (99,NULL,'Stevan',NULL,NULL,NULL)
  ```
- ```
  INSERT INTO studenti (Upisan,Imes,Indeks)
  VALUES (2000, 'Stevan', 1000)
  ```
- ```
  INSERT INTO studenti (indeks) VALUES (101)
  ```
- ```
  INSERT INTO Studenti (Indeks, Upisan, Imes, Mesto, Datr, Ssmer)
  SELECT Indeks, Upisan, Imes, Mesto, Datr, Ssmer
  FROM Kandidati WHERE Položen = 1
  ```

UPDATE

```
UPDATE [ONLY] {table_name | view_name}
SET {{column_name={DEFAULT|NULL| scalar_expression},
      column_name = {DEFAULT|NULL|scalar_expression}
      [,...] } | ROW = row_expression }
[ WHERE search_condition | WHERE CURRENT OF cursor_name ]
```

- ```
  UPDATE Prijave SET Ocena = Ocena + 1
  WHERE Indeks = 99 AND Upisan = 2000 AND Spred = 17
  ```

- ```
  UPDATE Prijave SET Ocena = 8
  WHERE Indeks = 99 AND
     Upisan = 2000 AND
     Spred = 17 AND
     Datump = '2007-03-24'
  ```

- ```
  UPDATE Studenti SET Mesto = (  SELECT Mesto
                                 FROM Studenti
                                 WHERE Imes = 'Sima')
    WHERE Imes = 'Stevan' AND Indeks = 99;
  ```
  ```

DELETE

```
DELETE FROM { table_name | ONLY (table_name) }
[{ WHERE search_condition | WHERE CURRENT OF cursor_name }]
```

- DELETE FROM STUDENTI WHERE Indeks = 40 AND Upisan = 2003;

**Primer 38.**

```
SELECT * FROM Nastavnici;
```

**Primer 39.**

```
SELECT Mesto FROM Studenti;
```

**Primer 40.**

```
SELECT DISTINCT Mesto FROM Studenti;
```

**Primer 41.**

```
SELECT Indeks, Upisan, Imes, Mesto FROM Studenti
WHERE Upisan = 2003
```

**Primer 42.**