Programski Sistemi

Baze podataka

podsetnik

Baza podataka

- Baza je zbirka podataka organizovanih na specifičan način
- Postoje različiti tipovi baza:
  - Relacione
  - Objektno orijentisane
  - Distribuirane
  - NoSQL
- Nas zanimaju isključivo relacione baze, u kojima su podaci organizovani u obliku tabela povezanih relacijama

SQL

- Structured Query Language
- Jezik koji se koristi za rad sa relacionim bazama podataka; kreiranje tabela, vršenje upita, unos, brisanje i ažuriranje podataka
- Aktuelna standardizovana verzija je SQL-99, a verzija SQL-92 je univerzalno prihvaćena
- Što bolje savladate SQL, bićete uspešniji u izvlačenju podataka iz baze

DBMS

- Database Management System
- Aplikacija koja se koristi za upravljanje bazom podataka
- Najpoznatija besplatna: MySQL
- Komercijalne: Oracle, IBM DB2, Microsoft SQLServer, PostgreSQL
- Baza podataka generalno nije prenosiva između različitih DBMS sistema (za to ponekad postoje posebni alati)

Nezavisnost podataka

- Sistemi za upravljanje bazama podataka nisu vezani za određeni programski jezik, niti su relacione baze specifične za određenu aplikaciju
- Taj važan princip zove se nezavisnost podataka
- Drugim rečima, podaci traju duže od bilo koje aplikacije

Relaciona baza

- Sastoji se od tabela, koje imaju zapise (redove) i polja (atribute, kolone)
- Svaka kolona ima svoj tip (int, date, char itd.); raspoloživi tipovi kolona zavise od konkretnog DBMS sistema
- Redosled zapisa nije bitan
- Struktura tabela i veze između njih predstavljaju se E-R (entity-relationship) dijagramom

Ključevi tabela

- Relacije između tabela se opisuju uz pomoć ključeva
- Tabela može da ima polje koje jedinstveno određuje zapis (red) u tabeli; to polje se zove primarni ključ (PRIMARY KEY)
- Primarni ključ može da bude i kombinacija kolona; tada se zove kompozitni ključ

Ključevi tabela

- Ako nema pogodnog kandidata za primarni ključ, najbolje je uvesti surogat-ključ (npr. redni broj zapisa)
  - Većina DBMS sistema za to nudi opciju auto-increment
- Kada se glavni ključ neke tabele koristi kao polje u drugoj tabeli, to polje se zove strani (spoljni) ključ (FOREIGN KEY)

Integritet baze podataka

- Podaci uneti u bazu moraju biti tačni, validni i konzistentni
- Proveru integriteta obavlja DBMS
- Dva su najvažnija ograničenja za integritet (integrity constraints):
  - Integritet primarnog ključa
  - Referencijalni integritet

Integritet primarnog ključa

- Vrednost null znači odsustvo informacija za polje nekog zapisa
- Vrednost polja koje čine primarni ključ ne može biti null i mora biti jedinstvena

Nije prihvatljivo!

BrojIndeksa	Ime	Prezime	DatumRođenja
2012201063	Vladica	Kostić	03-22-1988
NULL	Ksenija	Veselinović	04-13-1989
2010201071	NULL	NULL	08-04-1985
2012202095	Zorica	Lazić	NULL
2012201063	Ivan	Branković	NULL

Može biti prihvatljivo ako atribut nije neophodan

Referencijalni integritet

- Odnosi se na spoljne ključeve (foreign key)
- Spoljni ključ može biti null i može da ima ponovljene vrednosti u zapisima, ali:
- Vrednost spoljašnjeg ključa mora da postoji u zapisima tabele na koju se odnosi
- Tabele u kojima ovo nije ispunjeno narušavaju referencijalni integritet baze

Referencijalni integritet

Student	Ocena	Predmet
2012201063	10	01
2010200716	8	04
2012202095	9	01
2011201820	10	03
2010200199	6	05

?

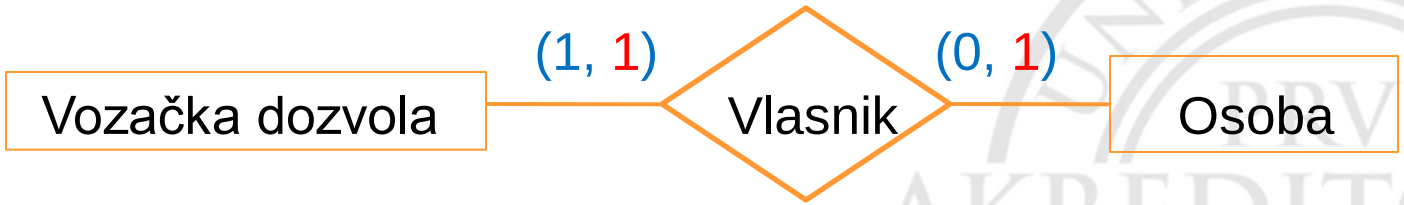
Kod	Naziv	Predavač
01	OOP	Irina Branović
02	Baze podataka	Mladen Veinović
03	Programski jezici	Dejan Živković
04	Aplikativni softver	Violeta Tomašević

Kardinalnost relacija

- One-to-one
- One-to-many
  - Modelira se tabelama povezanim pomoću spoljašnjeg ključa
- Many-to-many
  - Modelira se pomoćnom tabelom koja sadrži informacije isključivo o relacijama između dve tabele

One-to-one

- Osoba može da ima samo jednu vozačku dozvolu (ili da je nema)
- Vlasnik vozačke dozvole je jedna osoba
- Podaci o vlasniku i dozvoli su obično u istoj tabeli

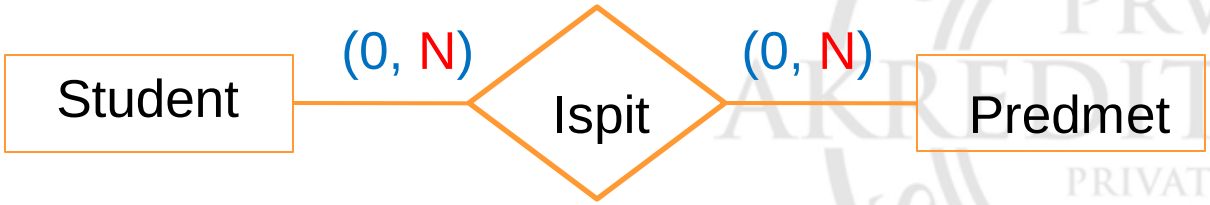


Many-to-one

- Svaka osoba ima samo jedan grad u kome je prijavljena da živi (prebivalište)
- Grad može da ima puno rezidenata, ili nijednog

Many-to-many

- Student je izlazio na ispit iz proizvoljnog broja predmeta (ili nijednog)
- Ispit iz određenog predmeta je polagao proizvoljan broj studenata (ili nijedan)



Primer many-to-many relacije

Kod	Opis	Stanje	Cena
A001	Ekser	234000	100
A804	Šraf	102400	200
P0101	Šrafciger	200	550
P0230	Čekić	600	7

Proizvod

Primarni ključ

Kod	Količina	Datum	Klijent
A001	1000	02-02-2013	0034
P0101	30	11-04-2013	0034
P0230	7000	01-03-2013	0012
P0101	2	07-08-2013	0002

Spoljni ključ

Isporuka

Spoljni ključ

Primarni ključ

Klijent	Ime	Telefon
0101	Zorica Lazić	011-234523
0034	Nikola Majstorović	065-456372
0012	Aleksandar Živković	023-786531
0002	Sanja Stjepanović	061-223312

Klijent

Join (ukrštanje)

- Join je akcija koja se sprovodi nad više tabela, a daje rezultat u obliku tabele
- Join je ono što bazu čini relacionom
- Postoji nekoliko vrsta ukrštanja:
  - Cross join
  - Left i right join (odnosi se na smer tabela u SQL naredbi; retko se koristi right jer programeri razmišljaju sleva udesno i tabelu koja upravlja upitom stavljaju na početak)
  - Inner i outer join

Primer ukrštanja (join)

- Tabele koje se ukrštaju

a
1
2
3
4

Tabela A

b
3
4
5
6

Tabela B

Cross join

- Dekartov proizvod kolona

a	b
1	3
2	3
3	3
4	3
1	4
2	4
3	4
4	4

a	b
1	5
2	5
3	5
4	5
1	6
2	6
3	6
4	6

Inner i outer join

- Ako zamislimo zapise iz dve tabele kao skupove:
  - Inner join daje presek skupova; left inner join je najviše korišćen i najlakši za razumevanje
  - Outer join daje uniju skupova

a	b
3	3
4	4

inner join

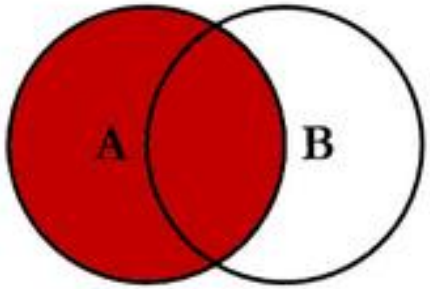
a	b
1	NULL
2	NULL
3	3
4	4
NULL	5
NULL	6

full outer join

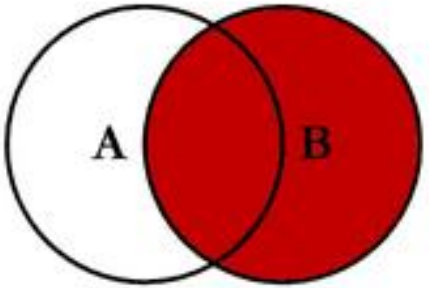
a	b
1	NULL
2	NULL
3	3
4	4

left outer join

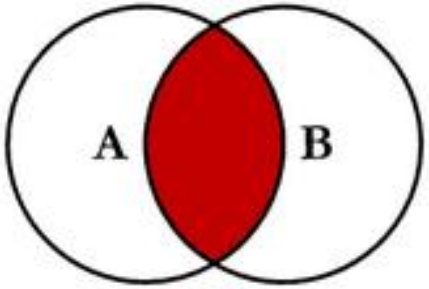
SQL JOINS



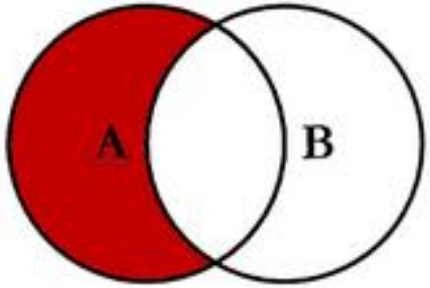

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



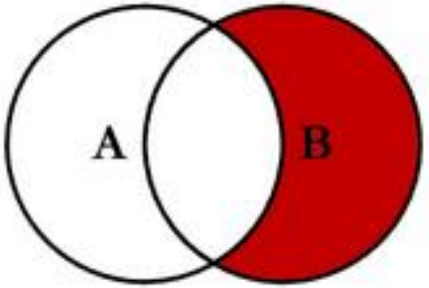

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



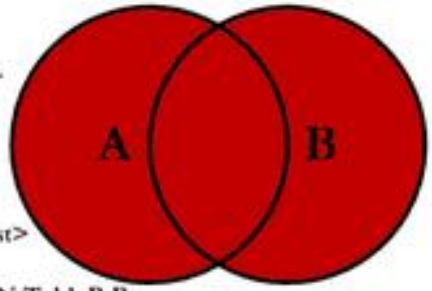

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



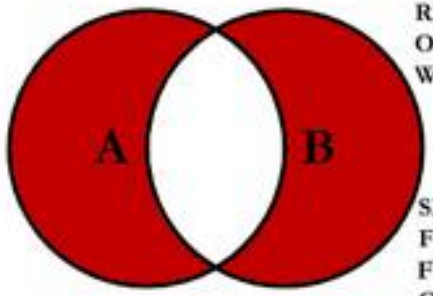

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Normalizacija

- Proces u kome se baza podataka projektuje tako da se ukloni redundansa, tj. tako da se poboljša organizacija tabela
- Da li će se i kako baza normalizovati dolazi sa iskustvom; to je kompromis između efikasnosti održavanja podataka i brzine izvršavanja upita
- Normalizacija se obavlja poštovanjem pravila (zovu se normalne forme)

Prva normalna forma (1NF)

- Prepoznati i eliminisati atribute koje se ponavljaju (tj. polja koja se pojavljuju sa različitim vrednostima za istu vrednost primarnog ključa)
- Organizovati ta polja kao posebnu tabelu, zajedno sa kopijom ključa iz početne tabele
- Nijedan entitet ne treba da ima polja iz drugog entiteta sa kojim je povezan

Primer normalizacije

Tabela koja nije normalizovana

IDProjekat	ImeProjekta	Vodja	Budzet	IDZaposleni	ImeZaposleni	IDOdeljenje	ImeOdeljenje	CenaPoSatu
PC010	Penzije	Ivan Branković	245000	10001	Marko Cicvarić	L004	IT	18
PC010	Penzije	Ivan Branković	245000	10030	Predrag Davidović	L001	Administracija	12.5
PC010	Penzije	Ivan Branković	245000	21010	Stana Bićanin	L004	IT	16
PC045	Plate	Nikola Rančić	174000	10010	Bojan Madov	L004	IT	18
PC045	Plate	Nikola Rančić	174000	10001	Marko Cicvarić	L004	IT	18
PC045	Plate	Nikola Rančić	174000	31002	Mladen Trampić	L008	HR	11
PC045	Plate	Nikola Rančić	174000	13210	Saša Rusmir	L005	Marketing	14
PC064	HR	Aleksa Jonić	122500	31002	Mladen Trampić	L008	HR	15
PC064	HR	Aleksa Jonić	122500	21010	Stana Bićanin	L004	IT	18
PC064	HR	Aleksa Jonić	122500	10034	Vladica Kostić	L009	Prodaja	15

- Polja *IDZaposleni*, *ImeZaposleni*, *IDOdeljenje*, *ImeOdeljenje* i *CenaPoSatu* se ponavljaju za istu vrednost primarnog ključa

Primer normalizacije: 1NF

IDProjekat	ImeProjekta	Vodja	Budzet
PC010	Penzije	Ivan Branković	245000
PC045	Plate	Nikola Rančić	174000
PC064	HR	Aleksa Jonić	122500

IDProjekat	IDZaposleni	ImeZaposleni	IDOdeljenje	ImeOdeljenje	CenaPoSatu
PC010	10001	Marko Cicvarić	L004	IT	18
PC010	10030	Predrag Davidović	L001	Administracija	12.5
PC010	21010	Stana Bićanin	L004	IT	16
PC045	10010	Bojan Madov	L004	IT	18
PC045	10001	Marko Cicvarić	L004	IT	18
PC045	31002	Mladen Trampić	L008	HR	11
PC045	13210	Saša Rusmir	L005	Marketing	14
PC064	31002	Mladen Trampić	L008	HR	15
PC064	21010	Stana Bićanin	L004	IT	18
PC064	10034	Vladica Kostić	L009	Prodaja	15

Kompozitni primarni ključ

Druga normalna forma (2NF)

- Tabela koja se posmatra mora već da bude u 1NF
- Sve kolone u tabeli treba da zavise samo od primarnog ključa
- Tabele sa “prostim” (ne-kompozitnim) primarnim ključem ili bez ključa su automatski u 2NF
- Treba napraviti posebne tabele za skupove vrednosti koji se pojavljuju u više zapisa
- Te tabele treba povezati stranim ključem

Primer normalizacije: 2NF

- Tabela projekata je već u 2NF
- Polja *ImeZaposleni*, *IDOdeljenje* i *ImeOdeljenje* zavise samo od *IDZaposleni* i zato treba da budu premešteni u novu tabelu čiji je ključ *IDZaposleni*
- *CenaPoSatu* zavisi i od *IDProjekat* i od *IDZaposleni* jer zaposleni može da bude plaćen različito po satu u zavisnosti od projekta na kome radi. Zato ovo polje treba da ostane u originalnoj tabeli.

Primer normalizacije:2NF

IDProjekat	IDZaposleni	CenaPoSatu
PC010	10001	18
PC010	10030	12.5
PC010	21010	16
PC045	10010	18
PC045	10001	18
PC045	31002	11
PC045	13210	14
PC064	31002	15
PC064	21010	18
PC064	10034	15

IDZaposleni	ImeZaposleni	IDOdeljenje	ImeOdeljenje
10001	Marko Cicvarić	L004	IT
10030	Predrag Davidović	L001	Administracija
21010	Stana Bićanin	L004	IT
10010	Bojan Madov	L004	IT
10001	Marko Cicvarić	L004	IT
31002	Mladen Trampić	L008	HR
13210	Saša Rusmir	L005	Marketing
31002	Mladen Trampić	L008	HR
21010	Stana Bićanin	L004	IT
10034	Vladica Kostić	L009	Prodaja

Treća normalna forma (3NF)

- Treba eliminisati iz tabela sva polja koja ne zavise od primarnog ključa
- Drugim rečima, tabela treba da sadrži informacije samo o jednoj stvari
- U opštem slučaju, kad god sadržaj grupe polja može da se primeni na nekoliko zapisa u tabeli, ta polja treba organizovati u posebnu tabelu

Primer normalizacije: 3NF

- Tabela projekata iz 2NF automatski prelazi u 3NF jer ima samo jedan atribut koji nije ključ.
- Polje *ImeOdeljenje* više zavisi od *IDOdeljenje* nego od *IDZaposleni* i zato treba da bude premešteno u novu tabelu. *IDOdeljenje* je primarni ključ nove tabele i strani ključ u tabeli *Zaposleni*.

Primer normalizacije: 3NF

IDZaposleni	ImeZaposleni	IDOdeljenje
10001	Marko Cicvarić	L004
10030	Predrag Davidović	L001
21010	Stana Bićanin	L004
10010	Bojan Madov	L004
10001	Marko Cicvarić	L004
31002	Mladen Trampić	L008
13210	Saša Rusmir	L005
31002	Mladen Trampić	L008
21010	Stana Bićanin	L004
10034	Vladica Kostić	L009

IDOdeljenje	ImeOdeljenje
L004	IT
L001	Administracija
L008	HR
L005	Marketing
L009	Prodaja

Indeks (1)

- Struktura podataka koja ubrzava pronalaženje podataka u tabeli, bez pretraživanja svakog zapisa
- Indeks je kopija dela tabele, obično strukturirana u specifičnom obliku (binarno stablo, heš tabela) i može se napraviti nad bilo kojim poljem tabele
- U nekim bazama, indeksi se čuvaju nezavisno od tabela

Indeks (2)

- Sve relacione baze imaju neku vrstu optimizacije izvršavanja SQL-a; to se obično zove *Optimizer*
- Indeksi se mogu praviti tako da budu unique i non-unique
- Mogu se praviti i nad više polja; tada se zovu kompozitni indeksi

Pogled (view)

- Za razliku od tabela koje se čuvaju u bazi, pogled nije deo fizičke šeme; to je virtuelna tabela koja se dinamički pravi kada se to zatraži
- Pogled je zapravo definicija upita koji se čuva u bazi; rezultatima upita pristupa se kao da je u pitanju stvarna, a ne virtuelna tabela
- Pogledi zauzimaju veoma malo mesta, jer baza sadrži samo njihovu definiciju, a ne i podatke koje prezentuju

Pogled (2)

- Treba biti oprezan sa filtriranjem rezultata pogleda, npr. kada želimo da prikažemo samo mali deo zapisa velike tabele
- Filtriranje treba da se obavi u upitu, a ne nad rezultatima koje daje pogled, jer se u tom slučaju filtriranje obavlja kada je upit u bazi već izvršen
- Pogledi se koriste uglavnom tokom razvoja, jer na duži rok mogu da degradiraju performanse

Stored procedure

- Stored procedura je niz SQL naredbi snimljenih u bazi; njih mogu da koriste svi SQL korisnici ili programeri
- Sačuvane procedure se koriste zbog efikasnosti i pogodnosti
- Način na koji se stored procedura čuva u bazi zavisi od konkretnog DBMS sistema
  - Oracle: PL/SQL
  - Microsoft SQL Server: Transact-SQL

Transakcije

- Skup SQL naredbi se može grupisati u transakciju
- Ako sve prođu dobro, za transakciju se izvršava *commit*; ako se desi greška, izvršava se *rollback*, tj. poništava se dejstvo i onih komandi koje su dobro prošle pre nego što se desila greška

Trigeri (okidači)

- Programiran skup naredbi koje se često izvode, a pokreću se pre ili posle nekog događaja u samoj bazi
- Trigeri pokreću npr. stored procedure
- Ne smeju da sadrže komande rollback ili commit
- Najveća opasnost kod rada sa trigerima je rekurzivno pozivanje

MySQL tipovi podataka

Tip	Veličina (bajt)	Opis
TINYINT	1	Ceo broj (neoznačen 0-255, označen -127 – 128)
SMALLINT	2	Ceo broj(neoznačen 0-65535, označen -32768-32767)
MEDIUMINT	3	Ceo broj (neoznačen 0-16777215, označen -8388608-8388607)
INT	4	Ceo broj (neoznačen 0-429467295, označen 2147483648-2147483647)
BIGINT	8	Ceo broj (neoznačen 0-18446744, označen -9223372036854775808-9223372036854775807)
FLOAT	4	Decimalni broj u opsegu +/-3.402823466E38 do +/11.175494351E-38
DOUBLE PRECISION	8	Decimalni broj u opsegu +/- -1.7976931348623157E308 do +/-2.2250738585072014E-308
DECIMAL (length, decimal)	length	Decimalni broj opsega istog kao double ali se čuva kao char

MySQL tipovi podataka

Tip	Veličina (bajt)	Opis
TINYBLOB	length+1	Binarno polje max dužine 255 znakova
TEXT	length+1	64kb teksta
BLOB	length+1	64kb podataka
MEDIUMTEXT	length+3	16Mb teksta
MEDIUMBLOB	length+3	16Mb podataka
LONGTEXT	length+4	4GB teksta
LONGBLOB	length+4	4GB podataka

MySQL tipovi podataka

Tip	Veličina (bajt)	Opis
TIMESTAMP	4	YYYYMMDDHHMMSS ili YYMMDDHHMMSS ili YYYYMMDD, YYMMDD. Vrednost Timestamp se menja kad god se promeni vrednost zapisa. Vrednost NULL postavlja vrednost na trenutno vreme.
DATE	3	YYYY-MM-DD
TIME	3	HH:MM:DD
DATETIME	8	YYYY-MM-DD HH:MM:SS
YEAR	8	YYYY or YY
CHAR(length)	length	String fiksne dužine (length); ako je uneta vrednost kraća, dopunjava se prazninama do potrebne dužine.
VARCHAR(length)	length	String fiksne dužine (length, maksimum 255) kome se praznine na kraju odsecaju.
TINYTEXT	length+1	Tekstualno polje max dužine 255 znakova.

SQL podsetnik (1)

- Standard propisuje da SQL nije case-sensitive (mada se SQL naredbe obično pišu velikim slovima)
- Ipak, u nekim konfiguracijama nazivi tabela i kolona su case-sensitive (na primer, u DBMS koji rade pod OS koji su case-sensitive, kao što je Linux)
- Svaki DBMS ima svoje tipove podataka i pomalo različitu SQL sintaksu

SQL podsetnik (1)

- Kreiranje tabele
  - CREATE TABLE student (id SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY, BrojIndeksa CHAR(15), Ime CHAR(20), Prezime CHAR(20), DatumRodjenja DATE, Telefon CHAR(20), Email CHAR(20));
- Brisanje tabele
  - DROP TABLE student;
- Menjanje tabele
  - ALTER TABLE student ADD Age SMALLINT;

SQL podsetnik (2)

- Umetanje zapisa
  - INSERT INTO student (FirstName, LastName, BirthDate) VALUES (‘Ivan','‘Branković','1990-09-18');
- Menjanje zapisa
  - UPDATE student SET Age ='23' WHERE FirstName=‘Ivan;
- Brisanje zapisa
  - DELETE FROM student WHERE Age=23;
  - DELETE FROM student;

SQL podsetnik (3)

- Upit (query)
  - SELECT * FROM student;
  - SELECT * FROM student WHERE id > 1;
  - SELECT * FROM student WHERE email IS NOT NULL;
  - SELECT * FROM student ORDER BY LastName;
  - SELECT * FROM student WHERE BirthDate > '1989-03-06';
  - SELECT FirstName, LastName FROM student WHERE LastName LIKE 'C%';
  - SELECT * FROM names WHERE LastName IN (‘Branković',‘Jonić');

SQL podsetnik (4)

- Join
  - SELECT FirstName, LastName, Email FROM student, email WHERE student. id = email.student_id;

  Isto što i:
  - SELECT * FROM student LEFT JOIN email USING (student_id);