Poslovno-proizvodni informacioni sistemi

Prof. dr Mirjana Misita

Osnovne definicije

- **Podatak** je iskaz definisan jednom prostom izjavnom rečenicom.
- **Informacija** je novi podatak koji poseduje neku relevantnu novinu, a koja je rezultat obrade poznatih podataka.
- **Entitet** je digitalna slika, opisana ograničenim brojem diskretnih vrednosti (atributima), nekog kontinualnog procesa koji se odvija u realnom svetu (nekog dela realnog sveta). Svojsta entiteta opisuju se atributima.

Primeri entiteta

Entiteti mogu npr. biti:

- Osoba
- Proizvod
- Institucija
- Proces

Za entitet Osoba, atributi bi bili:

- Ime
- Prezime
- Godina rođenja,
- Matični broj,
- Pol,
- itd.

Ograničenja atributa za enititet Osoba bi bila:

- Ime ne sme biti izostavljeno
- Matični broj mora imati 13 cifara
- Pol može niti M ili Ž.

- Podaci o entitetima čuvaju se u tabelama. Za svaki entitet pravi se posebna tabela. Entitet predstavlja grupu sličnih objekata (npr. svi studenti), a atributi opisuju njihove zajedničke osobine (npr. ime, godina rođenja).
- Svaki atribut ima jedinstveno ime po kome se razlikuje od drugih u istoj tabeli. Vrednosti tog atributa su uvek istog tipa (npr. brojevi, tekst...), a skup svih dozvoljenih vrednosti za jedan atribut naziva se domen tog atributa.
- Atribut mora imati jednostavnu vrednost — ne sme se sastojati od više delova (na primer, puna adresa nije poželjna kao jedan atribut ako sadrži ulicu, broj i grad u jednom).
- Ako neki atribut zahteva dodatne podatke da bi bio u potpunosti opisan, onda je bolje da se od njega napravi novi entitet.

Definicija baze podataka

- Definicija: **Baza podataka (DB) je kolekcija međusobno logički povezanih podataka koji imaju određeno značenje.**
  Ti podaci su organizovani tako da omogućavaju efikasno čuvanje, pretraživanje, ažuriranje i upravljanje informacijama koje se odnose na neku oblast ili temu.
- Definicija: **DBMS (sistem za upravljanje bazama podataka)** je softverski paket koji omogućava kreiranje, organizovanje, čuvanje, pretraživanje, izmenu i brisanje podataka u bazi podataka, uz kontrolu pristupa i zaštitu integriteta podataka.
- Drugim rečima, **DBMS je posrednik između korisnika i baze podataka** — omogućava da korisnik radi sa podacima bez potrebe da zna detalje o načinu fizičkog skladištenja.

Administrator baze podataka

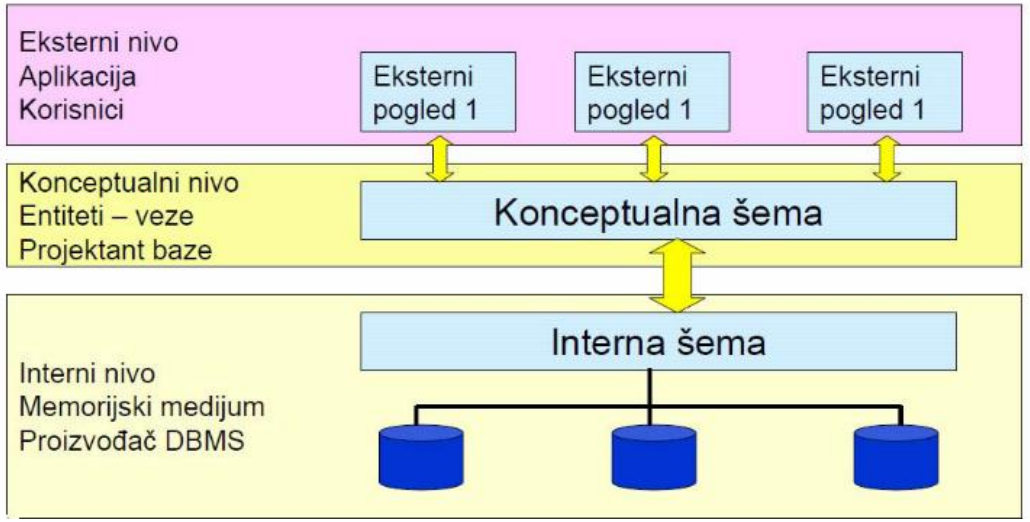
- Administrator baze podataka (DBA – Database Administrator) je osoba zadužena za instalaciju, konfiguraciju i održavanje baze podataka. Prilikom izrade baze podataka, osoba koja stvara bazu najčešće postaje njen DBA.
- Administrator ima najviši nivo korisničkih prava, što se tiče pristupa bazi i manipulisanja podacima. DBA dodaje ostale korisnike, u njegovoj je nadležnosti da određenim korisnicima dozvoli ili zabrani pristup pojedinim podacima itd. Isto tako DBA je zadužen za održavanje baze (backup).

Funkcije DBMS

- **Upravljanje podacima**
  Omogućava kreiranje, čuvanje, izmenu, brisanje i pretragu podataka u bazi.
- **Upravljanje strukturama baze**
  Omogućava definisanje tabela, veza između tabela, tipova podataka, ključeva itd.
- **Kontrola pristupa (bezbednost)**
  Određuje ko ima pravo da vidi, menja ili briše podatke — korisničke privilegije i autentifikacija.
- **Održavanje integriteta podataka**
  Održava tačnost i doslednost podataka (npr. kroz primarne i strane ključeve, ograničenja, pravila).
- **Upravljanje više korisnika (konkurentnost)**
  Dozvoljava da više korisnika istovremeno koristi bazu bez konflikata (zaključavanje, transakcije).
- **Oporavak od grešaka (recovery)**
  Omogućava povraćaj podataka u slučaju pada sistema, grešaka ili prekida u radu.
- **Rezervne kopije (backup)**
  Omogućava pravljenje sigurnosnih kopija podataka radi zaštite od gubitka.
- **Optimizacija upita i performansi**
  DBMS koristi mehanizme za efikasno izvršavanje upita i upravljanje resursima.

Troslojna arhitektura DBMS

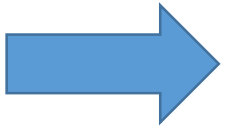
Eksterni nivo predstavlja pogled krajnjih korisnika i aplikacija na bazu podataka. Svaki korisnik ima svoj eksterni pogled, što znači da vidi samo deo podataka koji mu je potreban i dozvoljen. Na ovom nivou se definiše koje informacije su dostupne kome, čime se postiže kontrola pristupa i jednostavnije korišćenje baze.
Konceptualni nivo prikazuje celokupnu logičku strukturu baze podataka. Ovaj nivo je nezavistan od fizičkog skladištenja i sadrži sve entitete, njihove osobine (atribute) i međusobne veze.
Projektant baze na ovom nivou definiše konceptualnu šemu koja opisuje šta baza sadrži i kako su podaci povezani. Ovaj sloj je jedinstven za celu bazu i ne zavisi od pojedinačnih korisnika.



Interni nivo opisuje način na koji se podaci fizički čuvaju u računarskom sistemu. Na ovom nivou se nalazi interna šema, koja definiše strukture memorije, metode indeksiranja, organizaciju fajlova i druge tehničke detalje. Ovaj sloj je zadužen za efikasnost, prostor i performanse, i njime upravlja proizvođač DBMS-a.
Veze između ovih nivoa omogućavaju da se promene u jednom sloju (npr. u načinu skladištenja) ne moraju odražavati na drugim slojevima, čime se postiže **nezavisnost podataka**. Na taj način korisnici mogu da koriste bazu bez znanja o njenoj fizičkoj strukturi, a administratori mogu da unaprede performanse bez uticaja na korisničke aplikacije.

Modeli podatka

MODEL PODATAKA	OSNOVNA STRUKTURA
Relacioni	Relacija (tabela)
Mrežni	Graf
Hijerarhijski	Stablo
Objektno-orijentisani	Pokazivači (objekti)
Relacijsko-objektni	Relacije+objekti
NoSQL: Ključno-vrednosni	Parovi ključ–vrednost
NoSQL: Graf-model	Čvorovi i veze
NoSQL: Dokumenti i dr.	Dokumenti



Modeli podatka

Relacioni model koristi **relacije** (tabele) kao osnovnu strukturu. Podaci se čuvaju u tabelama koje se sastoje od redova i kolona, a veze između podataka ostvaruju se preko primarnih i stranih ključeva. Ovo je najčešće korišćen model u modernim bazama podataka kao što su MySQL, PostgreSQL i SQL Server.

Mrežni model koristi **graf** kao strukturu. U ovom modelu, podaci su predstavljeni kao čvorovi povezani vezama koje mogu biti više-veza (m:n odnosi). Ove veze se realizuju preko **pokazivača**, što omogućava kompleksne međusobne odnose između podataka. Mrežni model je bio popularan u ranijim generacijama DBMS-a, poput IDMS-a.

Hijerarhijski model koristi **stablo** kao osnovnu strukturu. Podaci su organizovani hijerarhijski – svaki roditeljski zapis može imati više „dece“, ali svako dete može imati samo jednog roditelja. Ova struktura je pogodna za podatke sa jasnom hijerarhijom, kao što su organizacione šeme ili sistemski fajlovi. Primer implementacije je IBM-ov IMS.

Objektno-orijentisani model koristi **pokazivače** i objekte. Podaci se čuvaju kao objekti koji sadrže i podatke (atribute) i ponašanje (metode), slično kao u objektno-orijentisanom programiranju. Ovaj model omogućava bolju integraciju sa programskim jezicima i koristi se u sistemima gde su podaci složeni i međusobno povezani, kao što su CAD sistemi ili multimedijalne aplikacije.

Relacijsko-objektni model predstavlja hibrid između relacionog i objektno-orijentisanog modela. Ovaj model zadržava tabelarnu strukturu relacionog modela, ali omogućava i uvođenje objekata, njihovih atributa i metoda. Na taj način se omogućava rad sa složenim podacima kao što su nizovi, ugnježđeni tipovi i korisnički definisane klase direktno unutar baze. Ovaj model omogućava bolje povezivanje aplikacione logike sa bazom i eliminiše potrebu za stalnim pretvaranjem između objekata i tabela. Primer sistema koji koristi ovaj model je **Oracle** sa podrškom za objektne tipove.

NoSQL modeli predstavljaju grupu nerelacionih modela podataka koji su razvijeni kao odgovor na potrebe za rad sa velikim količinama podataka, visokom skalabilnošću i fleksibilnošću. Za razliku od relacionih baza, NoSQL modeli nemaju strogu šemu (schema-less) i omogućavaju rad sa nestrukturisanim ili polustrukturisanim podacima. Postoji više podtipova NoSQL modela:

– **ključno-vrednosni** modeli čuvaju podatke kao parove ključ–vrednost (npr. Redis),

– **dokumentni** modeli čuvaju podatke kao JSON dokumente (npr. MongoDB),

– **graf modeli** čuvaju entitete kao čvorove i veze kao relacije (npr. Neo4j),

– **kolonijalni** modeli organizuju podatke po kolonama (npr. Cassandra).

NoSQL baze se često koriste u sistemima za analitiku, društvene mreže, real-time obradu i cloud aplikacije.

Relacioni model podataka

- **Relacioni model** je logički model podataka zasnovan na matematičkom pojmu **relacije** (tj. skupa uređenih vrednosti). U praksi, relacija se prikazuje kao **tabela** sastavljena od redova i kolona, gde:
- **Svaka relacija** predstavlja jednu tabelu.
- **Svaki red** predstavlja jedan zapis (instancu entiteta).
- **Svaka kolona (atribut)** predstavlja jedno svojstvo entiteta.
- Svi podaci u jednoj koloni su istog **tipa** (npr. tekst, broj, datum).
- Svaka tabela treba da ima **primarni ključ** – jedinstveni identifikator svakog reda.

Relaciona baza podataka se sastoji od vremenski promenljivih tabela (relacija) koje mogu biti bazne i izvedene.

- **Bazne tabele** se još nazivaju fizičke tabele zato što su zapisane u memoriji.
- **Izvedene** se izvode iz fizičkih upita.

U relacionim bazama postoji ne zavisnost aplikacije od podataka tako da se podaci mogu obrađivati pomoću raznih aplikacija što ih čini dostupnijim.

Struktura relacije (tabele)

ID	Ime	Prezime	Godina
1	Ana	Petrović	2021
2	Marko	Jovanović	2022
3	Petar	Todorović	2024
4	Jelena	Mitić	2022

U ovoj relaciji:
Naziv relacije je Student.
Atributi su: ID, Ime, Prezime, Godina.
Svaki red je jedan student (jedna instanca entiteta „Student“).
Primarni ključ je ID (jedinstveni identifikacioni broj ili ident broj).

Osnovne osobine relacionog modela

- Jedinstvena imena tabela i kolona
- Ne postoje duplikati redova
- Redosled redova i kolona nije bitan (matematički, skup)
- Vrednosti su atomične (ne mogu se dalje deliti)
- Atributi imaju definisane domene vrednosti (tipovi).

Svaka relacija treba da ima ključ. **Ključ minimalan skup jednog ili više atributa** koji **jedinstveno identifikuje svaki red** u relaciji (tabeli).

- Primarni ključ – jednoznačno identifikuje svaki red (npr. ID)
- Strani ključ – povezuje podatke između tabela (npr. StudentID u tabeli Ocene)
- Kandidat ključevi – svi potencijalni primarni ključevi

Pitanja iz ove lekcije:

1. Podatak/informacija
2. Entitet (i primer)
3. Atributi
4. DB/DBMS/DBA
5. Funkcije DMBS
6. Troslojna arihtektura DBMS
7. Vrste arhitektura DBMS
8. Vrste modela podataka
9. Relacioni model podataka
10. Osobine relacionog modela podataka

Relaciona algebra

Osnovni operatori relacione algebre, koji su ključni za razumevanje rada sa relacionim bazama podataka su:

- **Unija** – vraća sve redove iz obe relacije (bez duplikata),
- **Presek** – vraća samo zajedničke redove (može se izvesti iz unije i razlike).
- **Razlika (razlika skupova)** – vraća redove koji su u jednoj, ali ne i u drugoj relaciji,
- **Restrikcija (selekcija)** – filtrira redove na osnovu uslova,
- **Projekcija** – bira određene kolone,
- **Kartezijanski proizvod (proizvod)** – svaki red prve relacije se uparuje sa svakim redom druge,

Dodatni / izvedeni operatori:

- **Prirodno spajanje (natural join)** – spaja relacije po zajedničkim kolonama,
- **Deljenje (division)** – specifična operacija za univerzalne kvantifikatore,
- **Rename (promena imena)** – koristi se za dodeljivanje novog imena relaciji ili atributima.

Unija

- **Unija** je binarni operator relacione algebre koji kombinuje dve relacije (tabele) i vraća novu relaciju koja sadrži **sve jedinstvene torke (redove)** koje se nalaze **u bar jednoj** od te dve relacije.

Uslovi za primenu unije:

- Relacije moraju imati isti broj atributa.
- Odgovarajući atributi moraju biti definisani nad istim domenima (npr. oba ime: TEXT, oba id: INTEGER).

Primer unije

Tabela A

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
1772	Maksimović	Ilija	015 723 543

Tabela B

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
2345	Petrović	Dara	081 17318

Rezultat unije: A ∪ B

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
1772	Maksimović	Ilija	015 723 543
2345	Petrović	Dara	081 17318

Red sa šifrom **3244** se pojavljuje u obe tabele, ali u rezultatu unije se pojavljuje **samo jednom**.

Presek

- Presek dve relacije je nova relacija koja sadrži sve n-torke koje su zajedničke za obe relacije.
- **Presek** je binarni operator relacione algebre koji vraća **samo one redove** (torke) koji se nalaze **istovremeno u obe relacije**.

Uslovi za primenu preseka:

- Relacije moraju imati isti broj atributa.
- Odgovarajući atributi moraju biti definisani nad istim domenima.

Primer preseka

Tabela A

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
1772	Maksimović	Ilija	015 723 543

Tabela B

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
2345	Petrović	Dara	081 17318

Rezultat preseka: A ∩ B

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952

Razlika

- **Razlika** dveju kompatibilnih relacija je nova relacija koja ima iste atribute kao te relacije, a telo joj se sastoji od onih n-torki koje se nalaze u relaciji A, a ne nalaze se u relaciji B.
- U relacionoj algebri razlika se označava A/B.
- A-B ≠ B-A

Uslovi:

- Relacije moraju imati isti broj kolona,
- Kolone moraju imati isti redosled i biti definisane nad istim domenima.

Primer razlike

Tabela A

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
1772	Maksimović	Ilija	015 723 543

Tabela B

Šifra#	Prezime	Ime	Tel broj
3244	Aksentijević	Petar	071 334 952
2345	Petrović	Dara	081 17318

A – B

Šifra#	Prezime	Ime	Tel broj
1772	Maksimović	Ilija	015 723 543

B – A

Šifra#	Prezime	Ime	Tel broj
2345	Petrović	Dara	081 17318

Restrikcija

- **Restrikcija** (ili **selekcija**) je operator relacione algebre koji **filtrira redove** iz relacije tako što **zadržava samo one koji zadovoljavaju dati logički uslov**.

Uslovi za primenu restrikcije (selekcije):

- Relacija mora postojati i imati definisane kolone.
- Uslov mora biti logički izraz (npr. prosek > 8, godina = 3).
- Svi atributi koji se koriste u uslovu moraju postojati u relaciji.
- Tipovi podataka moraju biti kompatibilni (npr. broj se poredi s brojem, tekst s tekstom).
- Rezultat je nova relacija sa istim kolonama, ali manjim ili jednakim brojem redova.

Primer restrikcije:

Imamo tabelu **Student** sa kolonama:

ID	Ime	Godina	Prosek
101	Ana	1	8.2
102	Marko	3	9.1
103	Jelena	2	7.5
104	Nikola	3	6.9

Primer 1: studenti treće godine

Rezultat:

ID	Ime	Godina	Prosek
102	Marko	3	9.1
104	Nikola	3	6.9

Primer 2: studenti sa prosekom većim od 8

Rezultat:

ID	Ime	Godina	Prosek
101	Ana	1	8.2
102	Marko	3	9.1

Projekcija

- **Projekcija** je operator relacione algebre koji iz postojeće relacije (tabele) izdvaja **samo određene kolone (atribute)** koje nas interesuju, eliminišući sve ostale.

Uslovi za primenu projekcije:

- Svi navedeni atributi moraju postojati u relaciji.
- Atributi moraju biti navedeni po tačnom nazivu i redosledu.
- Rezultat sadrži samo odabrane kolone.
- Duplikati se automatski eliminišu.
- Tipovi i domeni se ne proveravaju jer se ne vrši poređenje.
- Projekcija ne zahteva tipološku kompatibilnost kao unija ili razlika, jer ne upoređuje podatke.

Primer projekcije

Polazna tabela: **Student**

ID	Ime	Smer	Godina	Prosek
101	Ana	Računarstvo	1	8.2
102	Marko	Menadžment	3	9.1
103	Jelena	Računarstvo	2	7.5
104	Ana	Računarstvo	1	8.2

Na primer iz tabele Student prikaži samo kolone Ime i Smer, a ostale kolone (ID, Godina, Prosek) zanemari.

Rezultat (posle uklanjanja duplikata):

Ime	Smer
Ana	Računarstvo
Marko	Menadžment
Jelena	Računarstvo

Red (Ana, Računarstvo) se pojavljuje dvaput u originalnoj tabeli, ali se prikazuje samo jednom jer projekcija automatski uklanja duplikate.

Kartezijanski proizvod (proizvod)

- **Kartezijanski proizvod** je binarni operator relacione algebre koji kombinuje **svaki red iz prve relacije sa svakim redom iz druge relacije**, dajući novu relaciju sa **sve mogućim kombinacijama redova**.

Uslovi za primenu kartezijanskog proizvoda:

•**Relacije ne moraju imati isti broj atributa.**

Svaka relacija može imati različit broj kolona.

•**Nema zahteva za istim domenima.**

Kolone ne moraju imati iste tipove podataka.

•**Kolone sa istim nazivima mogu izazvati dvosmislenost.**

Preporučuje se prethodno preimenovanje kolona (operator **rename**) ako postoji preklapanje

Pimer proizvoda

Tabela Student:

ID	Ime
1	Ana
2	Marko

Tabela Predmet:

Šifra	Naziv
P1	Matematika
P2	Informatika

Rezultat: Student × Predmet

ID	Ime	Šifra	Naziv
1	Ana	P1	Matematika
1	Ana	P2	Informatika
2	Marko	P1	Matematika
2	Marko	P2	Informatika

Tipovi veza između entiteta (relacija)

- **Veza 1 : 1** (jedan prema jedan)
- **Veza 1 : M** (jedan prema više)
- **Veza M : N** (više prema više)

Veza 1 : 1 (jedan prema jedan)

- Značenje: Svakom redu u jednoj tabeli odgovara tačno jedan red u drugoj tabeli, i obrnuto.

Primer: Tabela Zaposleni i tabela LičnaKarta

- Jedan zaposleni ima tačno jednu ličnu kartu.→
- Jedna lična karta pripada tačno jednom zaposlenom.

Tehnička realizacija:

- Primarni ključ iz jedne tabele može biti i strani ključ u drugoj.

Kada imamo vezu 1:1, možemo bezbedno spojiti dve tabele pomoću **prirodnog spajanja** (natural join), jer znamo da će **najviše jedan red** iz druge tabele odgovarati svakom redu iz prve

Veza 1 : M (jedan prema više)

- Značenje: Jednom redu u prvoj tabeli odgovara više redova u drugoj tabeli, ali svaki red u drugoj tabeli pripada samo jednom redu iz prve.

Primer:Tabela Profesor i tabela Predmet

- Jedan profesor predaje više predmeta.
- Svaki predmet predaje tačno jedan profesor.

Tehnička realizacija:

- Strani ključ se nalazi u tabeli sa "više" (npr. id_profesora u Predmet).

Veza 1 : M i restrikcija + spajanje. U vezi 1:M, često radimo spajanje (⋈) između glavne i zavisne tabele (npr. Profesor i Predmet), a zatim restrikciju (σ) da filtriramo ono što nas zanima.

Veza M : N (više prema više)

- Značenje: Jednom redu u prvoj tabeli može odgovarati više redova u drugoj i obrnuto.

Primer: Tabela Student i tabela Predmet

- Jedan student sluša više predmeta.
- Jedan predmet slušaju više studenata.

Tehnička realizacija:

- Potrebna je treća tabela (npr. Upis) sa stranim ključevima iz obe tabele:id_studenta, id_predmeta.

- Za M:N veze koristi se treća tabela (npr. Upis), i najčešće pravimo dvostruko spajanje

Veza	Tipični operatori relacione algebre
1 : 1	prirodno spajanje (⋈)
1 : M	spajanje (⋈), restrikcija (σ)
M : N	spajanje preko pomoćne tabele ili (× + σ)

SQL

- SQL - “ Structured Query Language“ strukturirani jezik za upite.
- razvijen je 1970.g. u IBM Research Laboratory u San Jose-u, California.
- 1981. IBM je predstavio prvi komercijalni SQL proizvod SQL/DS
- SQL je razvijen za rad sa relacionim bazama podataka za koje dr. Codd
- 1970 godine iznosi 12 Codd-ovih pravila

- SQL omogućava da kreiramo i menjamo strukturu baze podataka, dodamo prava korisniku za pristup bazama podataka ili tablelama, da dobijemo informacije od baze podataka i da menjamo sadržnaj baze podataka - odnosno imamo dve vrste funkcija:
- DDL (Data Definition Language) funkcije za definiciju podatka čiji je tipičan primer naredba *CREATE TABLE imeTabele ();*
- DML (Data Manipulation Language) funkcije za upravljanje podacima gdje kao primer možemo navesti osnovnu SQL naredbu *SELECT * FROM imeTabele*..

DDL- Data Definition Language

Za kreiranje baze koristi se naredba :

CREATE DATABASE imeBaze;

Primer pokazuje kreiranje (*CREATE*) i brisanje (*DROP*) novonastale baze podataka.

CREATE DATABASE proba;

DROP DATABASE proba;

isto važi i za tabele.

CREATE TABLE proba;

DROP TABLE proba;

- CREATE LOCATION CREATE SEQUENCE, CREATE VIEW itd.

CREATE

- Komanda CREATE se koristi za pravljenje novih objekata u bazi podataka, najčešće:
- tabela (CREATE TABLE),
- šema (CREATE SCHEMA),
- baza (CREATE DATABASE),
- pogled (CREATE VIEW),
- indeks (CREATE INDEX), itd.

CREATE

- Sintaksa za kreiranje tabele:

```
CREATE TABLE ime_tabele (
    naziv_kolone1 tip_podatka [OPCIJE],
    naziv_kolone2 tip_podatka [OPCIJE],
    ...
);
```

Primer:

```
CREATE TABLE korisnici (
    id SERIAL PRIMARY KEY,
    ime TEXT NOT NULL,
    prezime TEXT NOT NULL,
    email TEXT UNIQUE,
    datum_rodjenja DATE
);
```

DROP

Komanda DROP se koristi za trajno brisanje celih objekata iz baze podataka, uključujući:

- tabele (DROP TABLE)
- baze (DROP DATABASE)
- šeme (DROP SCHEMA)
- indekse (DROP INDEX)
- poglede (DROP VIEW) itd.

DROP

- DROP TABLE ime_tabele;
- DROP DATABASE ime_baze;
- DROP VIEW ime_pogleda;

- DROP TABLE IF EXISTS korisnici; koristi se da bi se izbegla greška ukoliko objekat ne postoji

ALTER

Komanda ALTER se koristi za izmenu postojeće strukture tabele u bazi podataka. Omogućava:

- dodavanje kolona,
- brisanje kolona,
- promenu tipa podataka,
- promenu imena kolona ili tabele, itd.

ALTER

- Dodavanje kolone:

ALTER TABLE ime_tabele ADD naziv_kolone tip_podatka;

- Brisanje kolone:

ALTER TABLE ime_tabele DROP COLUMN naziv_kolone;

- Izmena tipa podatka:

ALTER TABLE ime_tabele ALTER COLUMN naziv_kolone TYPE novi_tip;

- Promena imena kolone (PostgreSQL):

ALTER TABLE ime_tabele RENAME COLUMN staro_ime TO novo_ime;

- Promena imena tabele:s

ALTER TABLE staro_ime RENAME TO novo_ime;

Još komandi u DDL

Komanda **TRUNCATE** briše sve redove iz tabele, brzo i efikasno, ali bez mogućnosti WHERE uslova i često bez ROLLBACK-a (u zavisnosti od baze).

- TRUNCATE TABLE korisnici;
- Ova komanda briše sve podatke iz tabele korisnici, ali ne briše samu tabelu (za razliku od DROP).

Komanda **RENAME** se koristi za promenu imena tabele, kolone ili drugog objekta u bazi podataka.

- ALTER TABLE korisnici RENAME TO clanovi;

Komanda **COMMENT** se koristi za dodavanje opisa (komentara) na objekte baze podataka: tabele, kolone, indekse itd.

- COMMENT ON TABLE korisnici IS 'Tabela sa osnovnim podacima'

DML

- **DML** obuhvata komande koje se koriste za **rad sa podacima unutar postojećih tabela** – tj. za **ubacivanje, čitanje, izmenu i brisanje podataka**.
- Glavne DML komande:
- SELECT – dohvat (čitanje) podataka
- INSERT – ubacivanje novih redova
- UPDATE – izmena postojećih redova
- DELETE – brisanje redova

- SQL ne pravi razliku između malih i velikih slova, što znači da su sledeće dve naredbe jednake:

SELECT prezime FROM osoba where ime = 'pera'

ili

SELECT prezime FROM osoba WHERE ime = 'Pera'

- Radi lakšeg čitanja koda, preporučuje se da se ključne riječi (naredbe) pišu velikim slovima, svi ostali elementi malim slovima.
- U nekim bazama niz znakova (string) mora biti napisan kao što je u bazi. Znači u gornjim naredbama nije isto ako piše 'Pera' ili 'PERA'.

Pri kreiranju tabela određujemo nazive kolona, tip podatka koji će biti unet. Tipovi podataka su:

- Celobrojni:

- **bit** podatak koji je 1 ili 0

- **int** (**integer**) koji iznosi od –231 (-2,147,483,648) do 231-1

2,147,483,647) smešten u 4 byte-a

- **smallint** celi broj smešten u 2 byte-a; 215 (-32,768) do 215 - 1 (32,767)

- **tinyint** podatak od 0 - 255

- Decimalni:

- **decimal** ili **numeric** -(10)38 -1 do 1038 -1.

Primer je:

decimal(15, 3);. Prva cifra označava ukupan broj cifri, a druga broj decimalnih mesta iza decimalne tačke.

- Novac:

- **money** tip podatka je isti kao i decimal. Razlika je u zapisu.

-263 (-922,337,203,685,477.5808) do 263 - 1 (+922,337,203,685,477.5807)

- **smallmoney** 214,748.3648 do +214,748.3647

kod novčanog tipa podatka podaci se čuvaju sa četiri decimalna mesta.

- Datumi:

- **datetime** 1.avgust, 1753, do 31.decembar, 9999 uz tačnost od 3.33 milisekunde
- **smalldatetime** 1.avgust, 1753, do 31.decembar, 9999 uz tačnost od 1 minute

- Nizovi znakova:

- **char (character)** znakovni niz npr. char (9) u bazi će podatak zauzimati 9 znakova bez obzira na unešenu dužinu što znači da može doći do skraćivanja ili dopunjavanja. Maksimalno 8000 znakova.

- **nchar (**national char**)** upisuju se znakovi koji spadaju u

Unicode. Maksimalne dužine 4000 znakova.

- **text** unose se tekstualni podaci. Može sadržati 2,147,483,647

znakova.

- **varchar** promenjiva dužina (u bazi se čuva trenutna dužina podatka) ne Unicode znakova. Maksimalne dužine 8000 znakova.

- **nvarchar (**national char varying**)** promenjiva dužina Unicode znakova. Može sadržati 4000 znakova.

KREIRANJE TABELA

Maticni	Ime	prezime	ulica	mesto
0412974725028	Pera	Milić	Golsfortijeva 21	Novi Sad
2107979715020	Ivan	Matić	Cvijićeva 17	Niš
1503984725028	Marko	Jovanović	Vasina 5	Beograd

```
CREATE TABLE osoba
(
maticni NVARCHAR(15),
ime NVARCHAR(15) NOT NULL, prezime NVARCHAR(15) NOT NULL,
ulica NVARCHAR(25),
mesto NVARCHAR(15) DEFAULT 'Beograd'
PRIMARY KEY (maticni)
);
```

NAREDBA SELECT

Osnovna naredba u SQL-u je

SELECT izraz FROM imeTabele.

U prevodu

IZABERI izraz IZ ime Tabele.

U naredbi reč *izraz* zamenjujemo sa nazivima kolona koje želimo videti ili u slučaju da želimo videti sve sa *.

*SELECT * FROM imeTabele*

- ***Primer 1***

*SELECT * FROM osoba*

maticni	ime	prezime	ulica	mesto
0412974725028	Pera	Milić	Golsfortijeva 21	Novi Sad
2107979715020	Ivan	Matić	Cvijićeva 17	Niš
1503984725028	Marko	Jovanović	Vasina 5	Beograd

Primer 2

SELECT maticni, ime, prezime FROM osoba

maticni	ime	prezime
0412974725028	Pera	Milić
2107979715020	Ivan	Matić
1503984725028	Marko	Jovanović

- Možemo pored naziva kolone napisati novo ime kolone pod kojim ga želimo videti u izveštaju, ali da u tabeli ostaje sve po starom.

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

*SELECT 2 * plata, IDRadnika FROM TabelaPrimanjaRadnika*

Plata	**IDRadnika**
150000	010
130000	105
120000	152
120000	215
100000	244
90000	300
80000	335
64000	400
56000	441

Relacioni operatori

- U SQL-u postoji šest relacionih operatora i posle njihovog predstavljanja videćemo kako se koriste:

=	Jednako
<> ili !=	Različito
<	Manje
>	Veće
<=	Manje ili jednako
>=	Veće ili jednako

WHERE

- Da bi se prikazali samo oni redovi iz tabele koji zadovoljavaju određene kriterijume, koristi se *klauzula WHERE*. Ona se može najlakše razumeti ukoliko se pogleda nekoliko primera.
- Ukoliko želite da dobijete ID brojeve onih zaposlenih koji zarađuju preko 50.000, koristite sledeću naredbu:

```
SELECT IDRADNIKA
  FROM TABELAPRIMANJARADNIKA
  WHERE PLATA >= 50000;
```

- Obratite pažnju da se koristi znak >= (veće ili jednako), pošto smo želeli da izdvojimo one zaposlene koji zarađuju više od 50,000, ili jednako 50,000, i to prikazano zajedno. Kao rezultat dobijamo:
- IDRADNIKA

  010
  105
  152
  215
  244

- Isti tip operacije može se primeniti na tekstualne kolone:

```
SELECT IDRADNIKA
   FROM TABELAPRIMANJARADNIKA
   WHERE POLOŽAJ = 'rukovodilac';
```

- Ova naredba prikazuje ID brojeve svih rukovodilaca. Generalno, u slučaju tekstualnih kolona, koristite operatore jednako ili različito, i obavezno ceo tekst koji se pojavljuje u naredbi navedite unutar apostrofa (').

LOGIČKI OPERATORI AND, OR i NOT

- Dalji korak bi bio upotreba logičkih operatora tako da možemo još više proširiti uslove za pretraživanje.
- Logički operatori su *AND, OR* i *NOT.* Kod složenijih izraza bi trebalo voditi računa o prioritetima. Hijerarhija primjene operatora glasi:
- zagrada (),
- deljenje / i množenje * ,
- sabiranje + i oduzimanje - ,
- *NOT* (ne), *AND* (i), *OR* (ili).

..IS NULL

- Na početku smo opisali kolona ulica u tabeli Radnik je takva da nije nužno upisati podatak. Ipak poželjno je znati za koga nemamo podatke u tablici, a upravo to nam daje *IS NULL*.
- *SELECT ime, prezime FROM radnik WHERE ulica IS NULL*

- *SELECT ime, prezime FROM radnik WHERE ulica =’’ OR ulica IS NULL*

- *IS NULL* nije 0, nije prazan string – već bukvalno „nema podatka“).

ORDER BY

- Uz korišćenje ove naredbe možemo rezultat sortirati po nekom redosledu:

- *SELECT polozaj, plata, ime, prezime FROM radnik ORDER BY plata*

- *SELECT polozaj, plata, ime, prezime FROM radnik ORDER BY plata, prezime*

Ovom naredbom dobili bi smo rezultat sortiran od najniže ka najvišoj plati, a zatim po prezimenu

- ORDER BY ASC – za rastući niz
- ORDER BY DESC – za opadajući niz

- *SELECT polozaj, plata, ime, prezime FROM radnik ORDER BY plata DESC, prezime ASC*

DISTINCT, ALL

- Da bi smo videli sve kategorije polozaja u tabeli radnik uneli bi

SELECT polozaj FROM radnik

- Pri tome neke kategorije polozaja se ponavljaju, npr. Imamo više administratora
- Gornja naredba je skraćeni oblik:

SELECT ALL polozaj FROM radnik

- ..i daje iste rezultate

- Da bi smo videli u rezultatima samo kategorije u koloni polozaj potrebno je koristiti ključnu reč DISTINCT

- *SELECT DISTINCT polozaj FROM radnik*

FUNKCIJE SAKUPLJANJA

U funkcije sakupljanja spadaju

- *SUM* (suma svih vrednosti),
- *AVG* (prosečna vrijednost),
- *COUNT* (broj redova dobijenih rezultata),
- *MAX* (maksimalna vriednost u izrazu),
- *MIN* (minimalna vriednost u izrazu),
- *STDEV* (standardna devijacija),
- *VAR* (varijansa).

- SELECT SUM(plata) *FROM radnik*
- Rezultat je suma svih plata radnika koji se nalaze u datoj tabeli
- SELECT SUM(plata), AVG(plata), COUNT(plata), MAX(plata), MIN(plata) *FROM radnik*
- Rezultat je suma svih plata, prosečna plata, broj plata koje su unete u tabelu, minimalna plata i maksimalna plata

MATEMATIČKE FUNKCIJE

- Matematičke funkcije su:
- *ABS* (apsolutna vrjednost),
- *ACOS* (arkus kosinus),
- *ASIN* (arkus sinus),
- *ATAN* (arkus tangens),
- *COS* (kosinus),
- *COT* (kotangens),
- *LOG* (logaritam baze 2),
- *LOG10* (logaritam baze 10),
- *PI* (3,14),
- *POW*(stepenovanje),
- *RADIANS* (pretvara stepene u radijane),
- *RAND* (generator slučajnih brojeva),
- *ROUND* (zaokruživanje decimalnih brojeva na zadatu preciznost),
- *SIN* (sinus),
- *SQRT* (drugi koren),

- SELECT SUM(plata), AVG(plata), COUNT(plata), MAX(plata), MIN(plata), SQRT(plata) *FROM radnik*

- *Osim prethodno izračunatih funkcija sakljupljanja prikazuje i kvadratni koren ukupnih plata*

- SELECT POW(AVG(plata), 2) *FROM radnik*
- *Rezultat je kvadrat prosečne plate*

Spajanje tabela