BAZE PODATAKA · MATERIJAL ZA VEŽBE

Uvod u SUBP i ER model

Sistem za upravljanje bazama podataka (DBMS) i model entiteti-veze

Deo I — Uvod u SUBP · Istorija SUBP-a · Arhitekture SUBP-a · Fajl sistem naspram SUBP-a

Deo II — Uvod u ER model · Strukturna ograničenja veza · Generalizacija, specijalizacija i agregacija · Zadaci za vežbu

Deo III — Relacioni model · Relaciona algebra · Kodova pravila · Ključevi · Prevođenje ER modela · Projektovanje šeme · Zadaci

Deo IV — Uvod u MySQL · Instalacija na Windows · DDL naredbe i tipovi podataka · Upravljanje korisnicima · Naredba SELECT · Aliasi, agregatne funkcije i pogledi

Sadržaj

Deo I — Osnove sistema za upravljanje bazama podataka 6

1. Uvod u SUBP 6

1.1 Šta je SUBP? 6

1.2 Ključne karakteristike SUBP-a 7

1.3 Problemi tradicionalnih sistema zasnovanih na datotekama 7

1.4 Komponente SUBP aplikacija 7

1.5 Tipovi SUBP-a 9

1.6 Jezici baza podataka 10

1.7 Primene SUBP-a 12

2. Istorija SUBP-a 13

2.1 Kratak pregled 13

2.2 Evolucija tehnika skladištenja i obrade podataka 13

3. Tipovi arhitektura SUBP-a 15

3.1 Jednoslojna arhitektura (1-Tier) 15

3.2 Dvoslojna arhitektura (2-Tier) 15

3.3 Troslojna arhitektura (3-Tier) 16

3.4 Uporedni pregled arhitektura 17

4. Razlika između fajl sistema i SUBP-a 18

4.1 Fajl sistem (File System) 18

4.2 SUBP 18

4.3 Uporedna tabela 19

4.4 Zaključak 20

Deo II — ER model (Model entiteti-veze) 21

5. Uvod u ER model 21

5.1 Šta je ER model? 21

5.2 ER model u procesu projektovanja baze podataka 21

5.3 Svrha i upotreba ER dijagrama 21

5.4 Simboli ER modela 22

5.5 Entiteti 22

5.6 Atributi 24

5.7 Tip veze i skup veza 25

5.8 Stepen veze 26

5.9 Kardinalnost 27

5.10 Ograničenje učešća (participacije) 29

5.11 Kako nacrtati ER dijagram – koraci 30

6. Strukturna ograničenja veza u ER modelu 31

6.1 Pojam strukturnih ograničenja....31
6.2 Kardinalni odnosi....31
6.3 Tipovi kardinalnosti....31
6.4 Ograničenja učešća....32
6.5 Strukturna ograničenja i (min, max) notacija....32
7. Generalizacija, specijalizacija i agregacija....34
7.1 Generalizacija....34
7.2 Specijalizacija....34
7.3 Nasleđivanje....35
7.4 Agregacija....36
7.5 Predstavljanje agregacije relacionom šemom....36
8. Zadaci za vežbu....37
8.1 Zadatak 1 – Gradska biblioteka....37
8.2 Zadatak 2 – Bolnica....38
8.3 Zadatak 3 – Fakultet (min, max)....40
8.4 Zadatak 4 – Aviokompanija....40
8.5 Zadatak 5 – Konsultantska firma....42
Deo III — Relacioni model....43
9. Uvod u relacioni model....43
9.1 Šta je relacioni model?....43
9.2 Terminologija....43
9.3 Najpoznatiji RDBMS sistemi....44
10. Relaciona algebra....45
11. Kodova pravila....46
12. Ključevi u relacionom modelu....48
12.1 Kandidat za ključ (Candidate Key)....48
12.2 Superključ (Super Key)....49
12.3 Alternativni ključ (Alternate Key)....49
12.4 Strani (spoljni) ključ (Foreign Key)....49
12.5 Delimični ključ (Partial Key)....50
12.6 Primarni ključ (Primary Key)....50
12.7 Sekundarni ključ (Secondary Key)....50
12.8 Jedinstveni ključ (Unique Key)....50
12.9 Složeni ključ (Composite Key)....50
12.10 Surogat (veštački) ključ (Surrogate Key)....51
13. Prevođenje ER modela u relacioni model....52
13.1 Slučaj 1: veza 1:1, jedan entitet ima potpuno učešće....52
13.2 Slučaj 2: veza 1:1, oba entiteta imaju delimično učešće....53
13.3 Slučaj 3: veza N:1 (1:N)....53
13.4 Slučaj 4: veza M:N....54

13.5 Slučaj 5: veza sa slabim entitetom 54
13.6 Pregled pravila prevođenja 55
14. Strategije projektovanja šeme 56
14.1 Četiri strategije 56
14.2 Koraci za kreiranje baze podataka 56
14.3 Pristupi projektovanju šeme 57
14.4 Ograničenja u SUBP-u 57
15. Zadaci: prevođenje ER modela u relacioni model 58
15.1 Zadatak 1 – Biblioteka u relacionom modelu 58
15.2 Zadatak 2 – Bolnica u relacionom modelu 58
15.3 Zadatak 3 – Fakultet u relacionom modelu 59
15.4 Zadatak 4 – Aviokompanija u relacionom modelu 60
15.5 Zadatak 5 – Konsultantska firma u relacionom modelu 61
Deo IV — MySQL u praksi 63
16. Uvod u MySQL 63
16.1 Šta je MySQL? 63
16.2 MySQL i SQL nisu isto 63
16.3 Kako MySQL obrađuje upit 64
16.4 Ključne mogućnosti 64
16.5 Gde se koristi 64
17. Instalacija MySQL-a na Windows 66
17.1 Sistemski zahtevi 66
17.2 Preuzimanje instalera 66
17.3 Instalacija komponenti 67
17.4 Konfiguracija servera 68
17.5 Dodavanje MySQL-a u PATH (opciono) 69
17.6 Provera instalacije 70
18. DDL naredbe i tipovi podataka 71
18.1 Kako se čitaju šabloni 71
18.2 Rad sa bazama 71
18.3 Kreiranje tabele (CREATE TABLE) 71
18.4 Tipovi podataka 72
18.5 Ograničenja (constraints) 74
18.6 Izmena tabele (ALTER TABLE) 76
18.7 Indeksi (CREATE INDEX) 76
18.8 Brisanje tabele (DROP i TRUNCATE) 76
18.9 Pregled DDL naredbi 77
18.10 Zadaci: DDL i tipovi podataka 77
19. Upravljanje korisnicima u MySQL-u 79
19.1 Kreiranje korisnika (CREATE USER) 79

19.2 Pregled prava (SHOW GRANTS) 79
19.3 Dodela i oduzimanje prava (GRANT, REVOKE) 79
19.4 Promena lozinke 80
19.5 Brisanje korisnika (DROP USER) 80
19.6 Ko je prijavljen: USER() i srodne funkcije 80
19.7 Pregled naredbi 81
19.8 Zadaci: upravljanje korisnicima 81
20. Naredba SELECT: zadaci nad bazom w3schools 83
20.1 Baza za vežbe 83
20.2 Šablon naredbe SELECT 83
20.3 Zadaci: SELECT 84
21. Aliasi, agregatne funkcije i pogledi: zadaci 86
21.1 Podsetnik: šabloni 86
21.2 Zadaci 86

DEO I

Osnove sistema za upravljanje bazama podataka

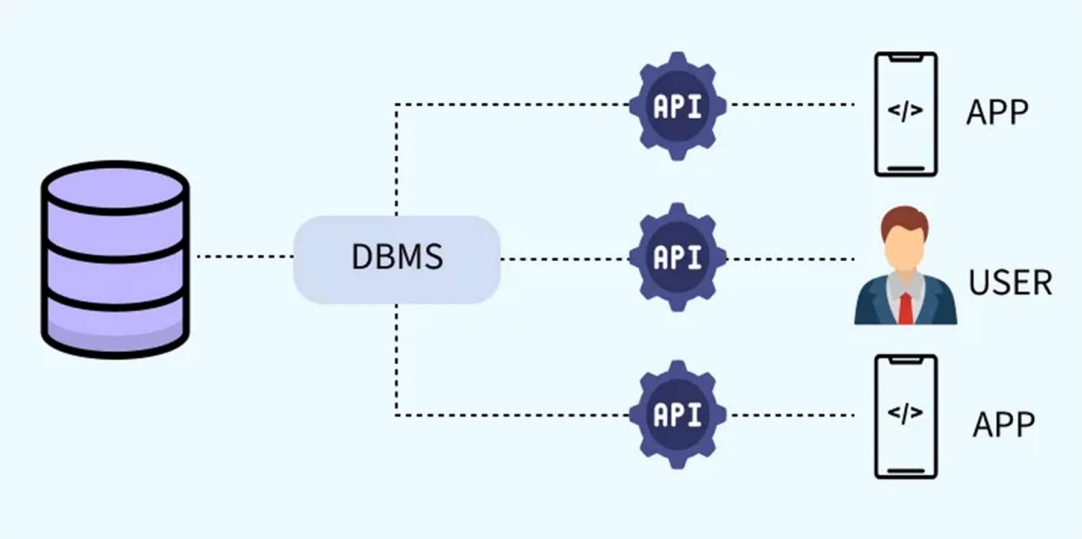
1. Uvod u SUBP

1.1 Šta je SUBP?

> DEFINICIJA
> **SUBP** (Sistem za upravljanje bazama podataka; *engl. DBMS – Database Management System*) jeste softverski sistem koji efikasno upravlja, skladišti i preuzima podatke u strukturiranom formatu. SUBP deluje kao most između centralne baze podataka i višestrukih klijenata, uključujući aplikacije i korisnike.

Osnovne uloge SUBP-a:

- Povezuje centralnu bazu podataka sa više klijenata (aplikacija i korisnika).
- Omogućava korisnicima da efikasno kreiraju, ažuriraju i pretražuju baze podataka.
- Osigurava integritet, konzistentnost i bezbednost podataka pri radu sa više korisnika i aplikacija.
- Smanjuje redundansu (dupliranje) i nekonzistentnost podataka putem centralizovane kontrole.
- Podržava istovremeni pristup podacima, upravljanje transakcijama i automatsko pravljenje rezervnih kopija (bekap).
- Koristi API interfejse za obradu zahteva za podacima, čime osigurava bezbedan i efikasan pristup.



Slika 1. *SUBP kao posrednik između baze podataka, aplikacija i korisnika*

1.2 Ključne karakteristike SUBP-a

SUBP je softverski sistem dizajniran za upravljanje, skladištenje, preuzimanje i rad sa podacima u bazama podataka. On služi kao interfejs između baza podataka i krajnjih korisnika i omogućava korisnicima i programerima da kreiraju, menjaju i uklanjaju baze podataka, kao i da organizuju okruženje u kome se podaci čuvaju u tabelama, zapisima i poljima.

SUBP podržava skladištenje, preuzimanje i ažuriranje podataka, kao i upravljanje samom bazom. Obično komunicira sa bazom pomoću jezika SQL (*Structured Query Language*) i prvenstveno se koristi za organizovano upravljanje velikim količinama podataka, uz očuvanje bezbednosti, konzistentnosti i integriteta.

- **Nezavisnost podataka** – izmene u strukturama podataka ne utiču na aplikativne slojeve.
- **Apstrakcija podataka** – pojednostavljuje rad sa bazom skrivajući složenost skladištenja od korisnika.
- **Kontrola istovremenog pristupa (konkurentnost)** – više korisnika istovremeno i bez konflikata pristupa podacima.

1.3 Problemi tradicionalnih sistema zasnovanih na datotekama

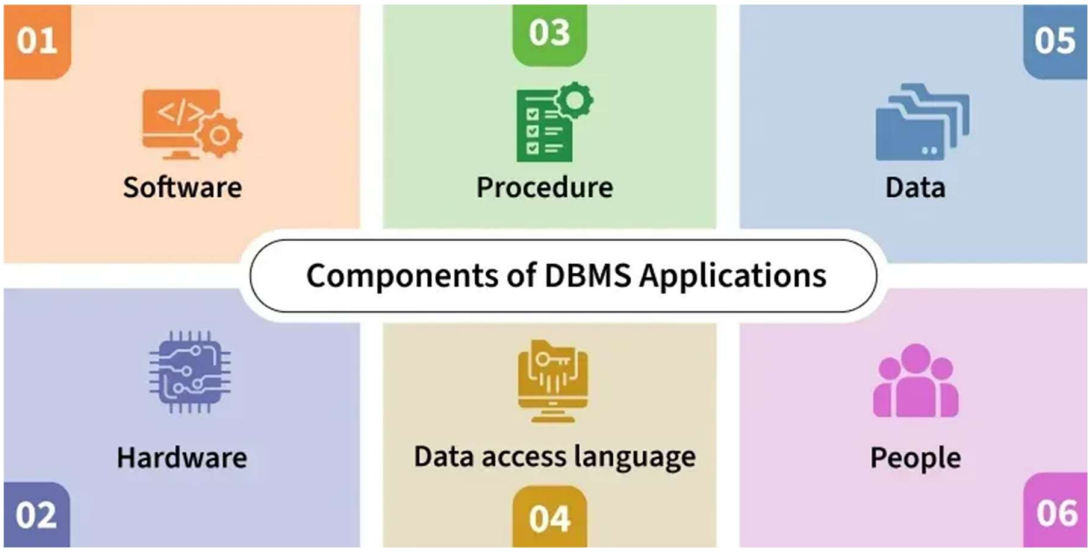
Pre uvođenja modernih SUBP-a, podacima se upravljalo pomoću osnovnih fajl sistema na hard diskovima. Iako je ovaj pristup omogućavao korisnicima da skladište, preuzimaju i ažuriraju datoteke po potrebi, nosio je sa sobom brojne izazove:

- **Redundansa (dupliranje) podataka:** duplirani unosi u različitim datotekama.
- **Nekonzistentnost:** kontradiktorne ili zastarele informacije.
- **Otežan pristup:** neophodna je bila ručna pretraga datoteka.
- **Loša bezbednost:** nedostatak kontrole nad pristupom podacima.
- **Nedostatak višekorisničke podrške:** bez podrške za zajednički rad.
- **Odsustvo bekapa i oporavka (Backup/Recovery):** gubitak podataka je često bio trajan.

> 💡 Primer: univerzitetski sistem zasnovan na datotekama, koji podatke čuva u odvojenim fajlovima (npr. *Nastava*, *Rezultati*, *Studentska služba*), često se suočavao sa svim navedenim problemima.

1.4 Komponente SUBP aplikacija

Svaka aplikacija zasnovana na SUBP-u sastoji se od šest ključnih komponenti koje rade zajedno kako bi se podacima upravljalo na efikasan način.



Slika 2. *Šest komponenti SUBP aplikacije*

1. Hardver (Hardware)

- Fizički uređaji kao što su serveri, diskovi i ulazno-izlazni uređaji (tastatura, monitor, štampač).
- Skladišti i obrađuje podatke; deluje kao interfejs između ulaza iz stvarnog sveta i digitalnih sistema.
- **Primeri:** hard disk računara, RAM memorija, mrežni uređaji koji se koriste za operacije SUBP-a.

2. Softver (Software)

- Sam SUBP softver, kao što su MySQL, Oracle, PostgreSQL.
- Obuhvata motor baze podataka (*database engine*), operativni sistem, mrežni softver i aplikativne alate.
- Prevodi jezike za pristup bazi podataka u konkretne operacije.

3. Podaci (Data)

- Sirove činjenice uskladištene u strukturiranim ili nestrukturiranim formatima.
- **Operativni podaci:** stvarni korisnički podaci (npr. ime, godine).
- **Metapodaci (Metadata):** podaci o podacima (npr. vreme skladištenja, veličina, tip podataka).
- Osnovni razlog postojanja SUBP-a – efikasno upravljanje i skladištenje podataka.

4. Procedure (Procedures)

- Uputstva i pravila za efikasno korišćenje SUBP-a.
- Pokrivaju podešavanje, prijavljivanje/odjavljivanje, validaciju podataka, bekap, kontrolu pristupa i generisanje izveštaja.
- Pomažu u obezbeđivanju doslednog i bezbednog korišćenja sistema.

5. Jezik za pristup bazi podataka (Database Access Language)

- Koristi se za interakciju sa bazom podataka (kreiranje, čitanje, ažuriranje, brisanje podataka).

- **Primeri:** SQL, MS Access, Oracle PL/SQL.
- **DDL** (*Data Definition Language*) – CREATE, ALTER, DROP.
- **DML** (*Data Manipulation Language*) – INSERT, UPDATE, DELETE.

6. Ljudi / Korisnici (People)

Korisnici koji stupaju u interakciju sa SUBP-om na različitim nivoima:

- **Administratori baza podataka (DBA):** upravljaju bezbednošću, performansama i pristupom korisnika.
- **Programeri (Developers):** izrađuju aplikacije koje koriste bazu podataka.
- **Krajnji korisnici (End Users):** koriste aplikacije za pristup bazi (npr. studenti, zaposleni).

1.5 Tipovi SUBP-a

Postoji nekoliko tipova sistema za upravljanje bazama podataka, od kojih je svaki prilagođen različitim strukturama podataka, zahtevima skalabilnosti i potrebama aplikacija. Najčešći tipovi su:

1. Relacioni SUBP (RDBMS)

- Organizuje podatke u tabele (relacije) sastavljene od redova i kolona.
- Koristi primarne ključeve za jedinstvenu identifikaciju redova i strane ključeve za uspostavljanje veza između tabela.
- Upiti se pišu u SQL-u, što omogućava efikasno upravljanje i preuzimanje podataka.
- **Primeri:** MySQL, Oracle, Microsoft SQL Server, PostgreSQL.

2. NoSQL SUBP

- Dizajnirani za rad sa velikim količinama podataka (*Big Data*); pružaju visoke performanse tamo gde relacioni model može biti ograničavajući.
- Skladište podatke u nerelacionim formatima: parovi ključ-vrednost, dokumenti, grafovi ili kolone.
- Fleksibilni modeli omogućavaju brzo skaliranje i odgovaraju nestrukturiranim ili polustrukturiranim podacima.
- **Primeri:** MongoDB, Cassandra, DynamoDB, Redis.

3. Objektno-orijentisani SUBP (OODBMS)

- Integriše koncepte objektno-orijentisanog programiranja; podaci se skladište kao objekti.
- Podržava složene tipove podataka i relacije – pogodan za napredno modelovanje i simulacije iz stvarnog sveta.
- **Primeri:** ObjectDB, db4o.

4. Hijerarhijska baza podataka

- Organizuje podatke u strukturu nalik stablu: svaki zapis (čvor) ima jednog roditelja i može imati više dece.

- Model je sličan fajl sistemu sa fasciklama i podfasciklama.
- Efikasna za podatke sa jasnom hijerarhijom (organizacione šeme, direktorijumi); navigacija je brza i predvidljiva.
- Nedostaje joj fleksibilnost i teško se koristi za složene veze „više prema više".
- **Primer:** IBM Information Management System (IMS).

5. Mrežna baza podataka

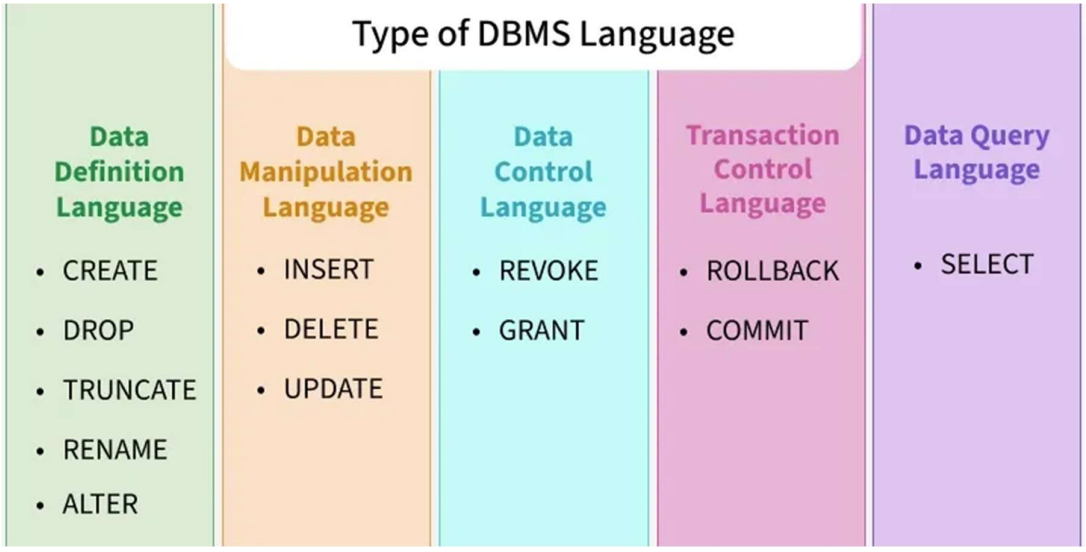
- Koristi model nalik grafu, čime omogućava složenije veze između entiteta.
- Za razliku od hijerarhijskog modela, dete može imati više roditelja – moguće su veze „više prema više".
- Podaci se predstavljaju zapisima i skupovima (*sets*), gde skupovi definišu veze.
- **Primeri:** Integrated Data Store (IDS), TurboIMAGE.

6. Baza podataka u oblaku (Cloud-Based Database)

- Smeštena na platformama kao što su AWS, Azure ili Google Cloud.
- Nudi skalabilnost po zahtevu, visoku dostupnost, automatski bekap i pristup sa daljine.
- Može biti relaciona (SQL) ili nerelaciona (NoSQL); održava je provajder, čime se smanjuju administrativni troškovi.
- **Primeri:** Amazon RDS (SQL), MongoDB Atlas (NoSQL), Google BigQuery.

1.6 Jezici baza podataka

Jezici baza podataka su specijalizovani skupovi naredbi koji se koriste za definisanje, manipulaciju i kontrolu podataka unutar baze. Svaki tip jezika ima posebnu ulogu u upravljanju bazom podataka.



Slika 3. *Podela jezika baza podataka*

1. DDL (Data Definition Language) – jezik za definisanje podataka

Bavi se shemama i opisima baze podataka, odnosno time kako podaci treba da budu smešteni u bazi.

- **CREATE:** kreira bazu i njene objekte (tabele, indekse, poglede, skladištene procedure, funkcije, okidače).
- **ALTER:** menja strukturu postojeće baze podataka.
- **DROP:** briše objekte iz baze podataka.
- **TRUNCATE:** uklanja sve zapise iz tabele i oslobađa prostor alociran za njih.
- **COMMENT:** dodaje komentare u rečnik podataka (*data dictionary*).
- **RENAME:** preimenuje objekat.

2. DML (Data Manipulation Language) – jezik za manipulaciju podacima

Omogućava dodavanje, ažuriranje i brisanje podataka uskladištenih u bazi.

- **INSERT:** ubacuje podatke u tabelu.
- **UPDATE:** ažurira postojeće podatke u tabeli.
- **DELETE:** briše zapise iz tabele.
- **MERGE:** operacija „UPSERT" (ubacivanje ili ažuriranje).
- **CALL:** poziva PL/SQL ili Java podprogram.
- **EXPLAIN PLAN:** prikazuje putanju pristupa podacima.
- **LOCK TABLE:** kontrola istovremenog pristupa (konkurentnosti).

3. DCL (Data Control Language) – jezik za kontrolu podataka

Upravlja pristupnim dozvolama, odnosno time ko može obavljati određene akcije u bazi.

- **GRANT:** dodeljuje prava korisniku (npr. SELECT, INSERT).
- **REVOKE:** oduzima prethodno dodeljene dozvole.

4. TCL (Transaction Control Language) – jezik za upravljanje transakcijama

Nadgleda transakcije kako bi se održale konzistentnost, pouzdanost i atomarnost.

- **COMMIT:** trajno čuva sve izmene napravljene tokom transakcije.
- **ROLLBACK:** poništava izmene napravljene tokom transakcije.
- **SAVEPOINT:** postavlja tačku unutar transakcije na koju se kasnije može vratiti.

5. DQL (Data Query Language) – jezik za upite nad podacima

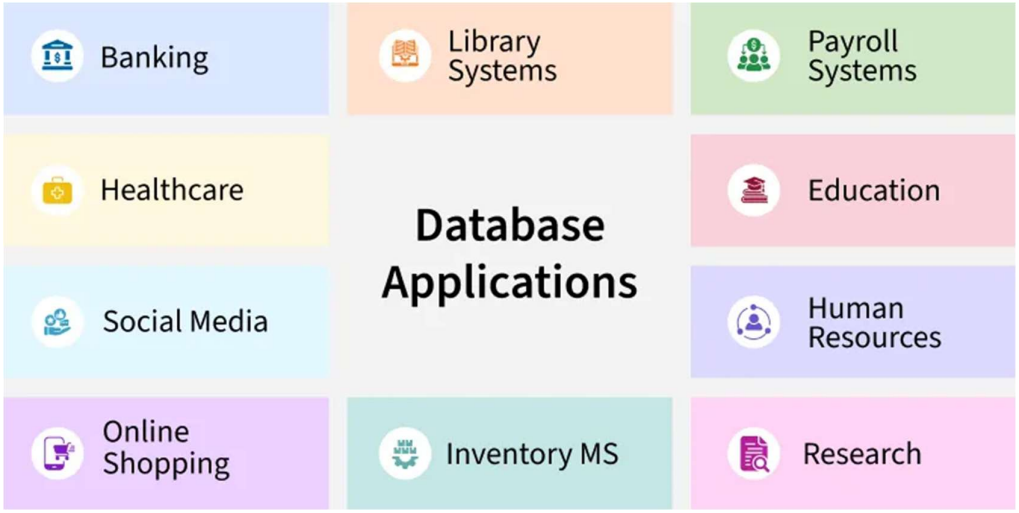
Podskup SQL-a koji služi za preuzimanje podataka bez njihovog menjanja. Glavna naredba je **SELECT**, koja korisniku vraća tačno one informacije koje traži.

Tabela 1. *Pregled jezika baza podataka*

Jezik	Namena	Naredbe
DDL	Definisanje strukture	CREATE, ALTER, DROP, TRUNCATE, COMMENT, RENAME

Jezik	Namena	Naredbe
DML	Rad sa podacima	INSERT, UPDATE, DELETE, MERGE, CALL, LOCK TABLE
DCL	Prava pristupa	GRANT, REVOKE
TCL	Transakcije	COMMIT, ROLLBACK, SAVEPOINT
DQL	Upiti	SELECT

1.7 Primene SUBP-a



Slika 4. *Oblasti primene SUBP-a*

- **Bankarstvo:** upravlja računima i transakcijama.
- **E-trgovina:** prati proizvode, porudžbine i kupce.
- **Zdravstvo:** skladišti kartone pacijenata i dijagnoze.
- **Obrazovanje:** upravlja ocenama studenata i rasporedima.
- **Društvene mreže:** upravljaju korisničkim profilima i interakcijama.
- **Nauka o podacima (Data Science):** podržava analitiku i predviđanja.

2. Istorija SUBP-a

2.1 Kratak pregled

Prvi sistemi za upravljanje bazama podataka nastali su 1960-ih godina kako bi se izašlo na kraj sa složenim poslovnim podacima. Među njima su *Integrated Data Store* (IDS) Čarlsa Bahmana (*Charles Bachman*) i IBM-ov *Information Management System* (IMS). Baze su najpre bile organizovane u strukture nalik stablu, koristeći hijerarhijski i mrežni model.

Edgar F. Kod (*Edgar F. Codd*) je 1970-ih popularizovao relacioni model, koji podatke organizuje u tabele (relacije), a za upite koristi SQL. Iz toga su nastali savremeni sistemi poput Oracle-a i MySQL-a, koji se i danas razvijaju i integrišu nove tehnologije, kao što su NoSQL baze za rad sa nestrukturiranim podacima.

Tabela 2. *Vremenska linija razvoja baza podataka*

Period	Ključna novina
1950-te i rane 1960-te	Magnetne trake i bušene kartice; sekvencijalna obrada (npr. obračun plata)
Kasne 1960-te i 1970-te	Hard diskovi i direktan pristup; mrežne i hijerarhijske baze; Kodov relacioni model (1970)
1980-te	System R, SQL/DS, Ingres, DB2, Oracle – relacione baze postaju dominantne
Rane 1990-te	Podrška odlučivanju, paralelne baze, objektno-relaciona podrška
1990-te	World Wide Web – visoke stope transakcija, dostupnost 24x7, veb-interfejsi
2000-te	XML/XQuery, open-source baze (PostgreSQL, MySQL), column-store i distribuirani sistemi

2.2 Evolucija tehnika skladištenja i obrade podataka

1950-te i rane 1960-te

Za skladištenje podataka razvijene su magnetne trake. Zadaci obrade podataka, kao što je obračun plata, automatizovani su, pri čemu su podaci čuvani na trakama. Obrada se sastojala od čitanja podataka sa jedne ili više traka i upisivanja na novu traku. Podaci su se mogli unositi i putem špila bušenih kartica, a izlaz se slao na štampače.

Kasne 1960-te i 1970-te

Masovna upotreba hard diskova krajem 1960-ih uveliko je promenila obradu podataka, jer su diskovi omogućili direktan pristup. Pozicija podataka na disku postala je nevažna, pošto se bilo kojoj lokaciji moglo pristupiti za svega nekoliko desetina milisekundi. Sa diskovima su nastale mrežne i hijerarhijske baze podataka koje su omogućavale skladištenje struktura poput lista i stabala. Prekretnički Kodov rad iz 1970. godine definisao je relacioni model i neproceduralne načine postavljanja upita, čime su rođene relacione baze podataka.

1980-te

Iako akademski zanimljiv, relacioni model u početku nije korišćen u praksi zbog slabijih performansi u odnosu na postojeće mrežne i hijerarhijske baze. To se promenilo sa projektom *System R* u okviru *IBM Research*-a, koji je razvio tehnike za izgradnju efikasnog relacionog sistema (pregledi: Astrahan i saradnici [1976], Čamberlin i saradnici [1981]). Prototip *System R* doveo je do IBM-ovog prvog komercijalnog relacionog proizvoda, *SQL/DS*. Istovremeno je na Univerzitetu Kalifornije u Berkliju razvijan sistem *Ingres*, koji je doveo do istoimenog komercijalnog proizvoda.

Prvi komercijalni relacioni sistemi – IBM DB2, Oracle, Ingres i DEC Rdb – unapredili su tehnike efikasne obrade deklarativnih upita. Do početka 1980-ih relacione baze postale su konkurentne i po performansama, a zbog jednostavnosti korišćenja na kraju su zamenile mrežne i hijerarhijske baze. Programeri starijih sistema morali su da se bave niskonivojskim detaljima i da pišu upite proceduralno, vodeći računa o efikasnosti; u relacionoj bazi većinu tih zadataka obavlja sama baza, pa programer radi na logičkom nivou.

Rane 1990-te

Jezik SQL je prvobitno dizajniran za aplikacije za podršku odlučivanju, koje intenzivno postavljaju upite, dok su okosnicu baza 1980-ih činile aplikacije za obradu transakcija, sa intenzivnim ažuriranjima. Podrška odlučivanju ponovo je postala glavna oblast primene, a alati za analizu velikih količina podataka zabeležili su veliki rast. Mnogi proizvođači predstavili su paralelne baze podataka i počeli da dodaju objektno-relacionu podršku.

1990-te

Glavni događaj decenije bio je eksplozivan rast Svetske mreže (*World Wide Web*). Baze podataka su se primenjivale obimnije nego ikada ranije i morale su da podrže veoma visoke stope obrade transakcija, izuzetnu pouzdanost i dostupnost 24x7 (bez zastoja zbog planiranog održavanja), kao i veb-interfejse za pristup podacima.

2000-te

Prva polovina 2000-ih donela je XML i prateći jezik za upite *XQuery*. Iako se XML široko koristi za razmenu podataka i skladištenje složenih tipova, relacione baze i dalje čine jezgro velike većine aplikacija. Razvijene su tehnike automatskog upravljanja (*autonomic computing / auto-admin*) radi smanjenja napora oko administracije, a značajno je porasla upotreba baza otvorenog koda, posebno *PostgreSQL*-a i *MySQL*-a.

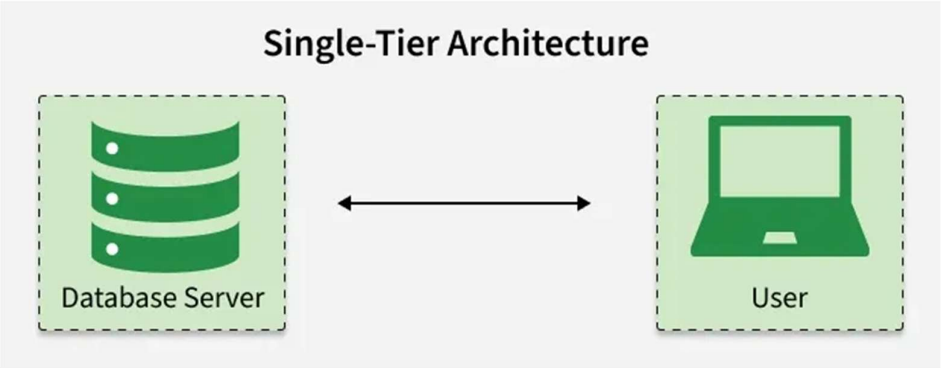
Kasniji deo decenije doneo je specijalizovane baze za analizu podataka, posebno *column-store* baze (svaku kolonu tabele čuvaju kao poseban niz) i visoko paralelne sisteme za analizu veoma velikih skupova podataka. Za potrebe velikih veb-sajtova (Amazon, Facebook, Google, Microsoft, Yahoo!) izgrađeno je nekoliko distribuiranih sistema za skladištenje, od kojih se neki danas nude kao veb-usluge.

3. Tipovi arhitektura SUBP-a

Arhitektura SUBP-a definiše način na koji korisnici stupaju u interakciju sa bazom podataka kako bi čitali, upisivali ili ažurirali informacije. Dobro dizajnirana arhitektura i shema (nacrt koji prikazuje tabele, polja i veze) obezbeđuju konzistentnost podataka, bolje performanse i bezbednost.

3.1 Jednoslojna arhitektura (1-Tier)

Korisnik radi direktno sa bazom podataka na istom sistemu: klijent, server i baza nalaze se unutar jedne aplikacije. Korisnik otvara aplikaciju, radi sa podacima i obavlja zadatke bez posebnog servera ili mrežne konekcije.



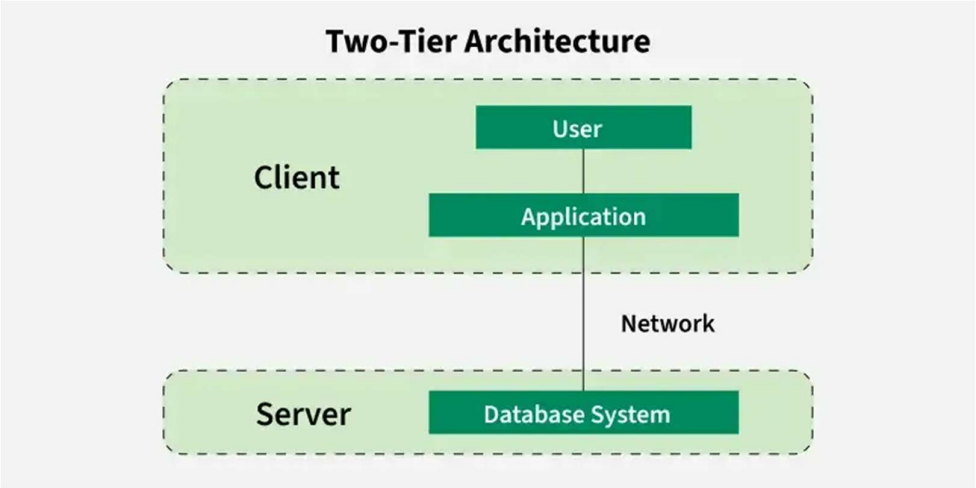
Slika 5. *Jednoslojna arhitektura*

Uobičajen primer je *MS Access*: korisnički interfejs, poslovna logika i skladištenje podataka nalaze se na istom uređaju. Ovo okruženje je jednostavno i idealno za lične ili samostalne (*standalone*) aplikacije i projekte manjeg obima.

Prednosti	Nedostaci
• **Jednostavna arhitektura:** dovoljan je jedan računar. • **Isplativost:** nije potreban dodatni hardver. • **Laka implementacija:** pogodna za manje projekte.	• **Samo jedan korisnik:** nije namenjena timskom radu. • **Slaba bezbednost:** pristup mašini znači pristup i podacima i aplikaciji. • **Nema centralizovane kontrole:** podaci su lokalni, bekap na više uređaja je otežan. • **Otežano deljenje podataka:** sve je na jednom računaru.

3.2 Dvoslojna arhitektura (2-Tier)

Slična je osnovnom klijent-server modelu: aplikacija na klijentu direktno komunicira sa bazom na serveru, putem API interfejsa kao što su **ODBC** i **JDBC**. Server obrađuje upite i upravlja transakcijama, dok klijent pokreće korisnički interfejs i aplikativne programe.



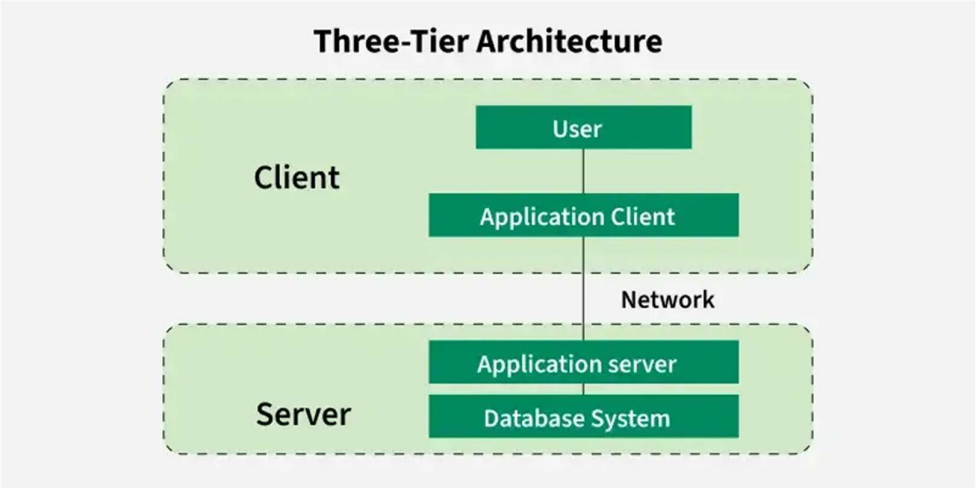
Slika 6. *Dvoslojna arhitektura*

> 💡 Primer: sistem za upravljanje školskom bibliotekom. **Klijentski sloj (sloj 1)** je desktop aplikacija za pretragu i izdavanje knjiga i proveru rokova. **Sloj baze (sloj 2)** je server koji čuva podatke o knjigama, članovima i transakcijama. Klijent šalje zahtev (npr. pretragu knjige), server ga obrađuje i vraća rezultat.

Prednosti	Nedostaci
• **Lak pristup:** brže preuzimanje podataka. • **Niski troškovi:** jeftinija od troslojne i višeslojne. • **Jednostavna implementacija.** • **Jednostavnost:** samo dve komponente.	• **Ograničena skalabilnost:** server se preopterećuje sa rastom broja korisnika. • **Bezbednosni problemi:** klijenti se direktno povezuju na bazu. • **Čvrsta povezanost (tight coupling):** izmena baze često zahteva izmenu klijenta. • **Otežano održavanje** i dodavanje funkcionalnosti.

3.3 Troslojna arhitektura (3-Tier)

Između klijenta i servera baze postoji dodatni sloj – **aplikativni server**. Klijent ne komunicira direktno sa bazom, već sa aplikativnim serverom, koji dalje komunicira sa bazom. Ovaj sloj je posrednik za razmenu delimično obrađenih podataka i tipičan je za velike veb-aplikacije.



Slika 7. *Troslojna arhitektura*

> 💡 Primer: online prodavnica. **Korisnik** pretražuje proizvod i dodaje ga u korpu; **aplikativni sloj** proverava stanje, računa cenu i primenjuje popuste; **baza podataka** čuva proizvode, korpu i istoriju porudžbina.

Prednosti	Nedostaci
• **Veća skalabilnost:** distribuirani aplikativni serveri, bez pojedinačne veze svakog klijenta sa bazom. • **Integritet podataka:** srednji sloj sprečava oštećenje podataka. • **Bezbednost:** klijent nema direktan pristup bazi.	• **Veća složenost:** dodatna komunikaciona tačka. • **Otežana interakcija** zbog srednjeg sloja. • **Sporiji odziv:** zahtev prolazi kroz dodatni sloj. • **Veći troškovi:** više hardvera, softvera i stručnog kadra.

3.4 Uporedni pregled arhitektura

Tabela 3. *Poređenje arhitektura SUBP-a*

Arhitektura	Kako komunicira	Primer	Pogodna za
1-Tier	Sve na jednom računaru	MS Access	Lične, male projekte
2-Tier	Klijent ↔ server baze (ODBC/JDBC)	Školska biblioteka	Manje organizacije
3-Tier	Klijent ↔ aplikativni server ↔ baza	Online prodavnica	Velike veb-aplikacije

4. Razlika između fajl sistema i SUBP-a

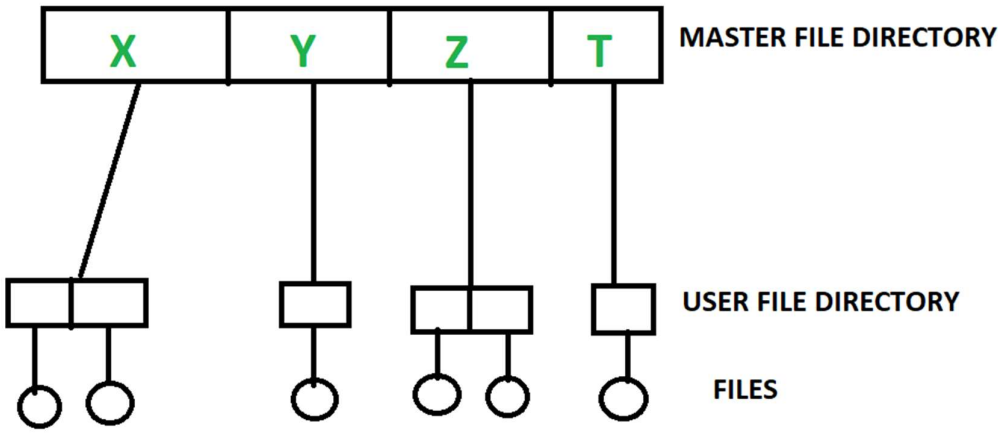
Fajl sistem i SUBP su dve vrste sistema za upravljanje podacima sa različitim namenama i karakteristikama. **Fajl sistem** organizuje datoteke u grupe i fascikle i skladišti ih na uređaj, omogućavajući čitanje, upis i brisanje. **SUBP** je složenija softverska aplikacija zadužena za upravljanje velikim količinama strukturiranih podataka, sa upitima, indeksiranjem, transakcijama i integritetom podataka.

Fajl sistem dobro služi aplikacijama kod kojih se podaci čuvaju jednostavno i ne zahtevaju veliku organizaciju, dok je SUBP prikladniji kada podaci moraju biti organizovani, optimizovani i zaštićeni.

4.1 Fajl sistem (File System)

Fajl sistem je način organizovanja datoteka na medijumu za skladištenje, kao što je hard disk. Datoteke su grupisane u direktorijume, koji dalje sadrže druge fascikle i datoteke. Fajl sistem obavlja osnovne operacije: upravljanje i imenovanje datoteka, pravila pristupa itd.

Primeri: *NTFS* (New Technology File System), *EXT* (Extended File System).

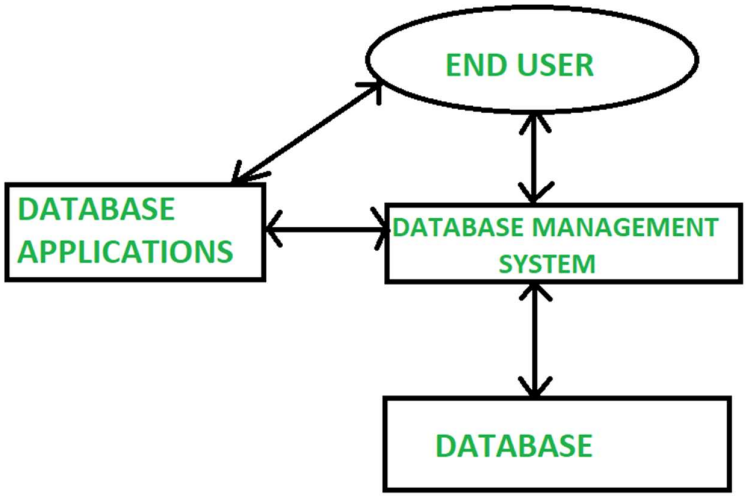


Slika 8. *Hijerarhijska organizacija fajl sistema*

4.2 SUBP

SUBP je softver koji upravlja kolekcijom povezanih podataka i efikasno ih skladišti i preuzima. Pruža bezbednosne mere protiv neovlašćenog pristupa, preuzimanje podataka pomoću SQL upita i relacione algebre, kao i mehanizme za oporavak i bekap.

Primeri: *Oracle*, *MySQL*, *MS SQL Server*.



Slika 9. *Mesto SUBP-a između korisnika, aplikacija i baze*

4.3 Uporedna tabela

Tabela 4. *Fajl sistem naspram SUBP-a*

Kriterijum	Fajl sistem	SUBP
Struktura	Način organizovanja datoteka na medijumu za skladištenje.	Softver za upravljanje bazom podataka.
Redundansa podataka	Duplirani podaci mogu biti prisutni.	Nema nepotrebnog dupliranja podataka.
Bekap i oporavak	Nema ugrađen mehanizam za bekap i oporavak.	Ugrađeni alati za bekap i oporavak.
Obrada upita	Ne postoji efikasna obrada upita.	Efikasna obrada upita.
Konzistentnost	Manja konzistentnost podataka.	Veća konzistentnost, zahvaljujući normalizaciji.
Složenost	Manje složen.	Složeniji za rukovanje.
Bezbednost	Manja bezbednost.	Više bezbednosnih mehanizama.
Cena	Jeftiniji.	Srazmerno skuplji.
Nezavisnost podataka	Ne postoji.	Postoji: logička i fizička nezavisnost.
Korisnički pristup	Jedan korisnik u datom trenutku.	Više korisnika istovremeno.
Procedure	Nije potrebno pisanje procedura.	Korisnik piše procedure/upite za rad sa bazom.
Deljenje podataka	Podaci razbacani po fajlovima – deljenje je teško.	Centralizovano – deljenje je lako.
Apstrakcija podataka	Otkriva detalje skladištenja i predstavljanja.	Skriva interne detalje baze.
Ograničenja integriteta	Teško se implementiraju.	Lako se implementiraju.

Kriterijum	Fajl sistem	SUBP
Atributi za pristup	Potrebni su naziv i lokacija datoteke.	Nisu potrebni.
Primeri	COBOL, C++ (rad sa fajlovima)	Oracle, SQL Server

4.4 Zaključak

Glavna razlika je u načinu organizovanja i upravljanja podacima. Fajl sistemi upravljaju datotekama i direktorijumima i pružaju osnovne operacije (kreiranje, brisanje, preimenovanje, pristup), obično u hijerarhijskoj strukturi. Jednostavni su i efikasni, ali ne mogu da upravljaju složenim vezama između podataka niti da obezbede konzistentnost.

SUBP obezbeđuje centralizovano i organizovano skladištenje podataka kojima može pristupati i menjati ih više korisnika ili aplikacija. Nudi validaciju podataka, indeksiranje, transakcije, kontrolu konkurentnosti, bekap i oporavak, a integritet čuva ograničenjima kao što su primarni ključevi, strani ključevi i tipovi podataka.

> **UKRATKO**
> **Fajl sistem** je pogodan za male količine nestrukturiranih podataka i osnovne operacije nad datotekama. **SUBP** je namenjen velikim količinama strukturiranih podataka i složenim slučajevima sa mnogo zapisa koje treba skladištiti, pretraživati i ažurirati, uz integritet, bezbednost i performanse.

DEO II

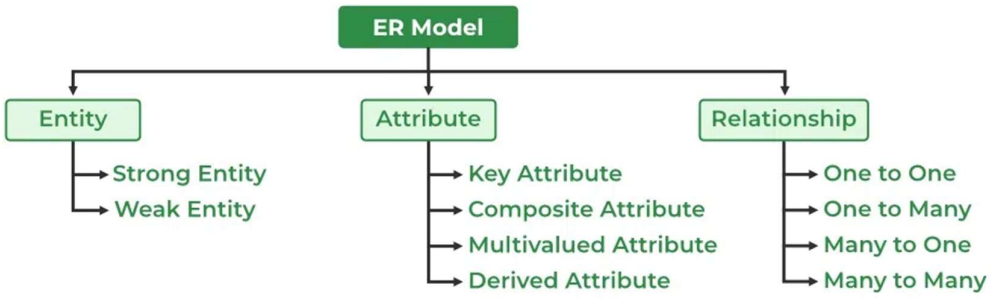
ER model (Model entiteti-veze)

5. Uvod u ER model

5.1 Šta je ER model?

> DEFINICIJA
> **Model entiteti-veze** (*ER model – Entity-Relationship Model*) jeste konceptualni model za projektovanje baza podataka. On predstavlja logičku strukturu baze, uključujući entitete, njihove atribute i veze (relacije) između njih. Grafički prikaz ovog modela naziva se **ER dijagram** (*ERD – Entity-Relationship Diagram*).

- **Entitet (Entity):** objekat koji se skladišti kao podatak. *Npr. Student, Predmet, Kompanija.*
- **Atribut (Attribute):** svojstvo koje opisuje entitet. *Npr. BrojIndeksa, NazivPredmeta, EmailZaposlenog.*
- **Veza / Relacija (Relationship):** povezanost između entiteta. *Npr. Student sluša Predmet.*



Slika 10. *Osnovni elementi ER modela*

5.2 ER model u procesu projektovanja baze podataka

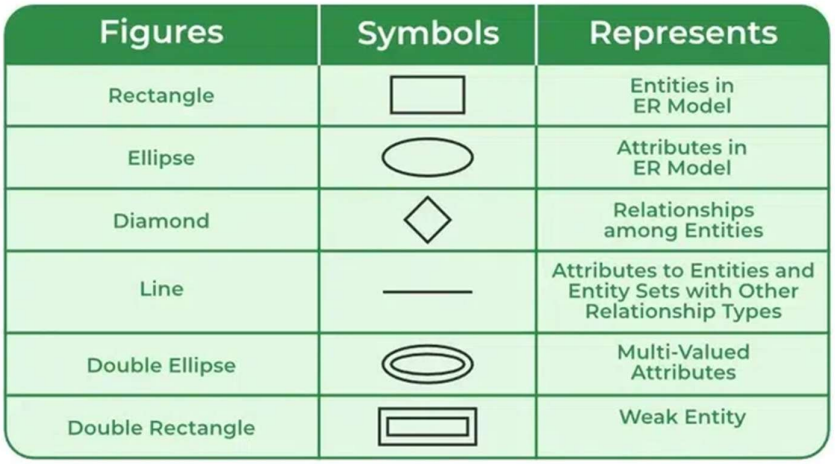
1. **Prikupljanje zahteva** (funkcionalnih i zahteva za podacima) postavljanjem pitanja korisnicima baze.
2. **Kreiranje konceptualnog (logičkog) projekta** baze podataka – ovde se koristi ER model, najčešći grafički prikaz konceptualnog projekta.
3. **Fizičko projektovanje** (npr. indeksiranje) i **spoljašnje projektovanje** (npr. pogledi).

5.3 Svrha i upotreba ER dijagrama

- ER dijagrami se lako prevode u relacije (tabele).
- Služe za modelovanje objekata iz stvarnog sveta.

- Ne zahtevaju tehničko poznavanje SUBP-a koji se koristi u pozadini.
- Vizuelno prikazuju podatke i njihove veze, pa složene sisteme čine razumljivijim.

5.4 Simboli ER modela



Slika 11. *Simboli koji se koriste u ER dijagramu*

Tabela 5. *Pregled simbola*

Simbol	Značenje
Pravougaonik	Jaki entitet
Dvostruki pravougaonik	Slabi entitet
Elipsa (oval)	Atribut
Podvučena elipsa	Ključni atribut
Dvostruka elipsa	Viševrednosni atribut
Isprekidana elipsa	Izvedeni atribut
Romb	Veza (relacija)
Dvostruki romb	Identifikaciona veza (slabog entiteta)
Linija / dvostruka linija	Povezivanje / potpuno (totalno) učešće

5.5 Entiteti

Entitet je objekat, koncept ili stvar iz stvarnog sveta o kojoj se podaci čuvaju u bazi. On je gradivni blok baze, a u relacionoj bazi entitete predstavljaju tabele.

- **Objekti iz stvarnog sveta:** Osoba, Automobil, Zaposleni…
- **Koncepti:** Predmet, Događaj, Rezervacija…
- **Stvari:** Proizvod, Dokument, Uređaj…

Tip entiteta definiše strukturu entiteta, dok pojedinačne **instance** tog tipa predstavljaju konkretne entitete.

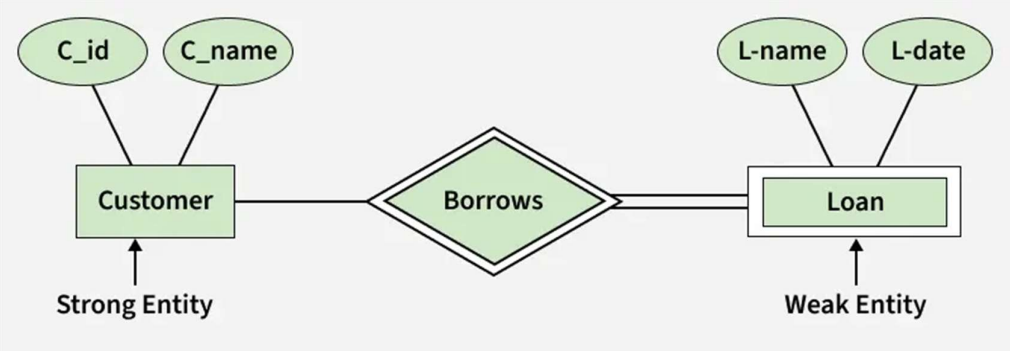
Jaki entitet (Strong Entity)

Ima ključni atribut koji jedinstveno identifikuje svaku instancu i ne zavisi ni od jednog drugog entiteta. Poseduje primarni ključ i u ER dijagramu se predstavlja **pravougaonikom**.

Slabi entitet (Weak Entity)

Ne može se jedinstveno identifikovati samo pomoću sopstvenih atributa, već zavisi od jakog (identifikacionog) entiteta. Predstavlja se **dvostrukim pravougaonikom**, a njegovo učešće u vezi je uvek potpuno. Veza između slabog entiteta i njegovog identifikacionog entiteta naziva se **identifikaciona veza** i crta se **dvostrukim rombom**.

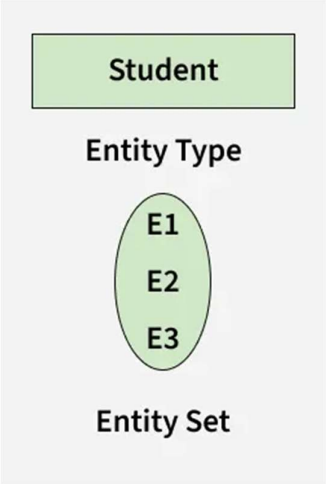
> 💡 Primer: kompanija čuva podatke o izdržavanim članovima porodice (*Dependents*) zaposlenog. Izdržavani član ne može postojati u sistemu bez zaposlenog, pa je on **slabi entitet**, a *Zaposleni* je njegov **identifikacioni (jaki) entitet**.



Slika 12. *Jaki entitet (Customer), slabi entitet (Loan) i identifikaciona veza*

Skup entiteta (Entity Set)

Entitet je pojedinačni objekat datog tipa, a kolekcija svih entiteta tog tipa naziva se **skup entiteta**. Na primer, *E1* je entitet tipa „Student”, a svi studenti zajedno čine skup entiteta.

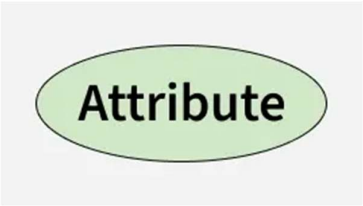


Slika 13. *Tip entiteta i skup entiteta*

U ER dijagramu predstavljamo skupove entiteta, ali ne i pojedinačne entitete: entitet je kao jedan red u tabeli, a ER dijagram prikazuje strukturu i veze, ne konkretne podatke.

5.6 Atributi

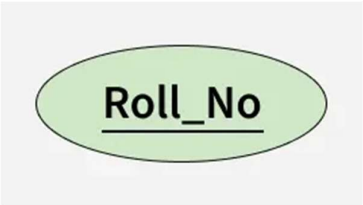
Atributi su svojstva koja opisuju tip entiteta. Za entitet *Student* to su, na primer, *BrojIndeksa*, *Ime*, *DatumRođenja*, *Godine*, *Adresa* i *BrojTelefona*. U ER dijagramu atribut se predstavlja **elipsom (ovalom)**.



Slika 14. *Atribut*

1. Ključni atribut (Key Attribute)

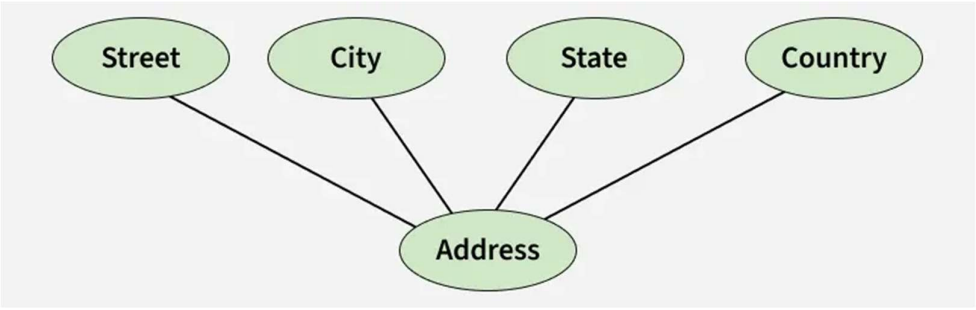
Jedinstveno identifikuje svaki entitet u skupu entiteta (npr. *BrojIndeksa* je jedinstven za svakog studenta). Crta se kao **elipsa sa podvučenim tekstom**.



Slika 15. *Ključni atribut*

2. Složeni atribut (Composite Attribute)

Sastoji se od više drugih atributa (npr. *Adresa* = *Ulica*, *Grad*, *Region*, *Država*). Crta se kao **elipsa iz koje se granaju druge elipse**.



Slika 16. *Složeni atribut Adresa*

3. Viševrednosni atribut (Multivalued Attribute)

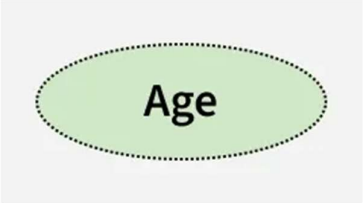
Može imati više vrednosti za jedan entitet (npr. *BrojTelefona* – student može imati više brojeva). Crta se kao **dvostruka elipsa**.



Slika 17. *Viševrednosni atribut*

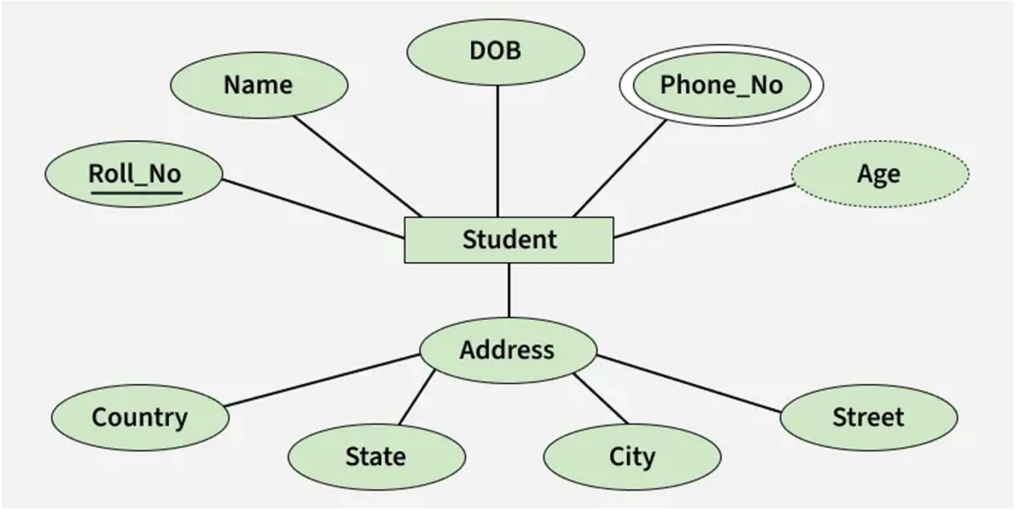
4. Izvedeni atribut (Derived Attribute)

Može se izračunati iz drugih atributa (npr. *Godine* iz *DatumRođenja*). Crta se kao **isprekidana elipsa**.



Slika 18. *Izvedeni atribut*

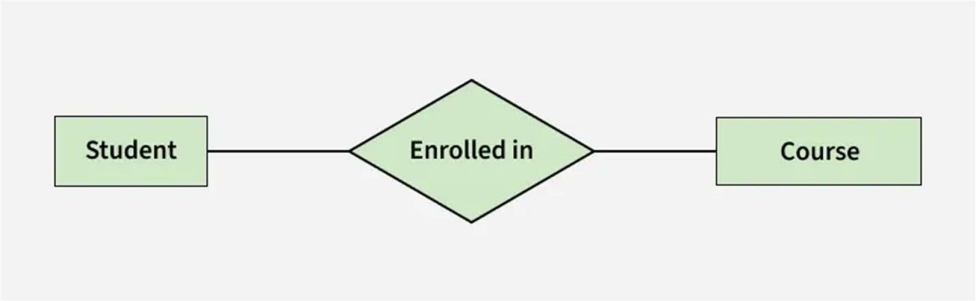
Kompletan tip entiteta **Student** sa svim vrstama atributa:



Slika 19. *Entitet Student sa svim tipovima atributa*

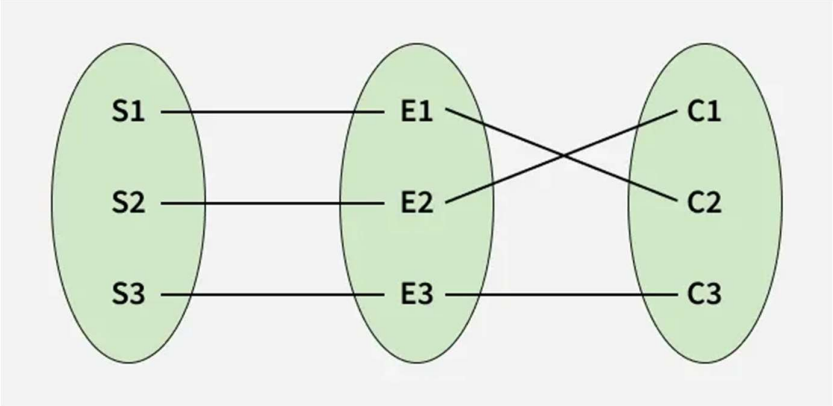
5.7 Tip veze i skup veza

Tip veze predstavlja povezanost između tipova entiteta. Na primer, *„Upisan na"* (Enrolled in) je tip veze između entiteta *Student* i *Predmet*. U ER dijagramu tip veze se predstavlja **rombom** povezanim linijama sa entitetima.



Slika 20. *Tip veze „Upisan na”*

Skup veza istog tipa naziva se **skup veza** (*relationship set*). Na slici je *S1* upisan na *C2*, *S2* na *C1*, a *S3* na *C3*.



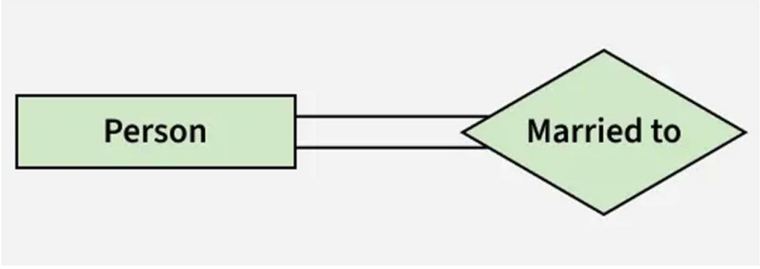
Slika 21. *Skup veza „Upisan na”*

5.8 Stepen veze

Broj različitih skupova entiteta koji učestvuju u vezi naziva se **stepen veze**.

1. Unarna (rekurzivna) veza

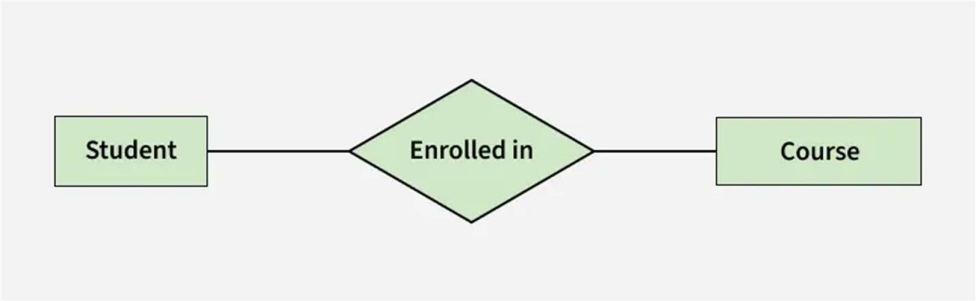
U vezi učestvuje samo **jedan** skup entiteta. Npr. osoba je u braku sa osobom (oba učesnika iz skupa *Osoba*).



Slika 22. *Unarna veza „U braku sa”*

2. Binarna veza

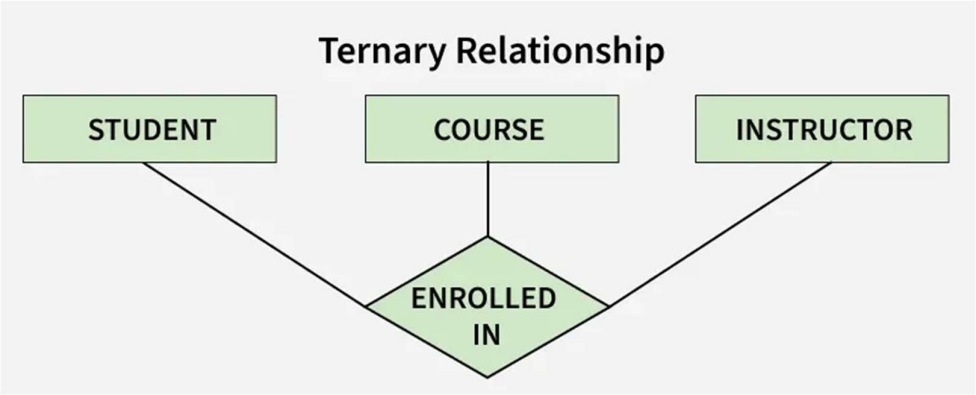
U vezi učestvuju **dva** skupa entiteta. Npr. *Student* je upisan na *Predmet*.



Slika 23. *Binarna veza*

3. Ternarna veza

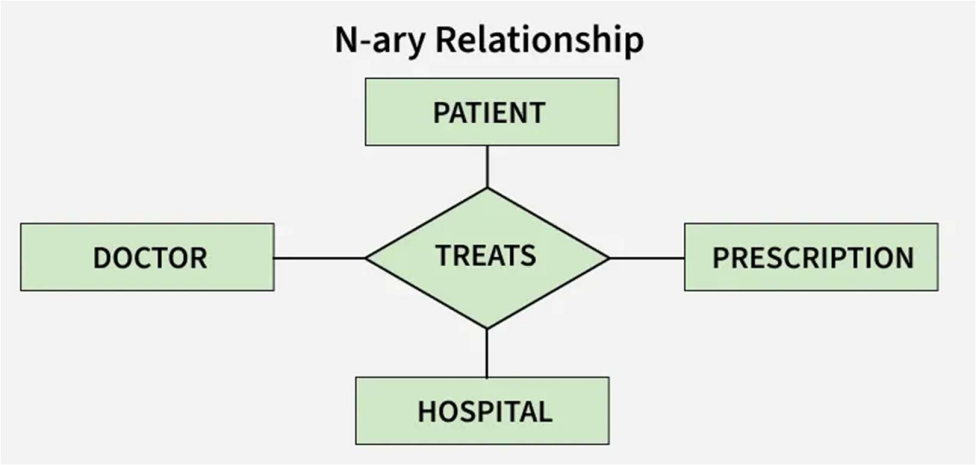
U vezi učestvuju **tri** skupa entiteta.



Slika 24. *Ternarna veza*

4. N-arna veza

U vezi učestvuje **n** skupova entiteta.



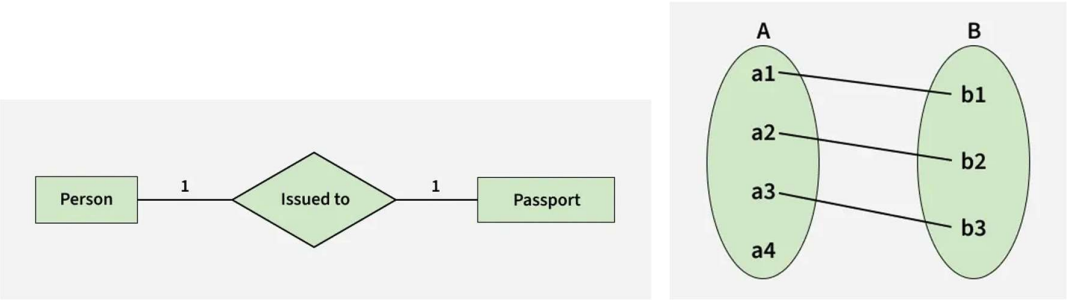
Slika 25. *N-arna veza*

5.9 Kardinalnost

Maksimalan broj puta koliko entitet iz nekog skupa može učestvovati u skupu veza naziva se **kardinalnost**.

1. Jedan-prema-jedan (1:1)

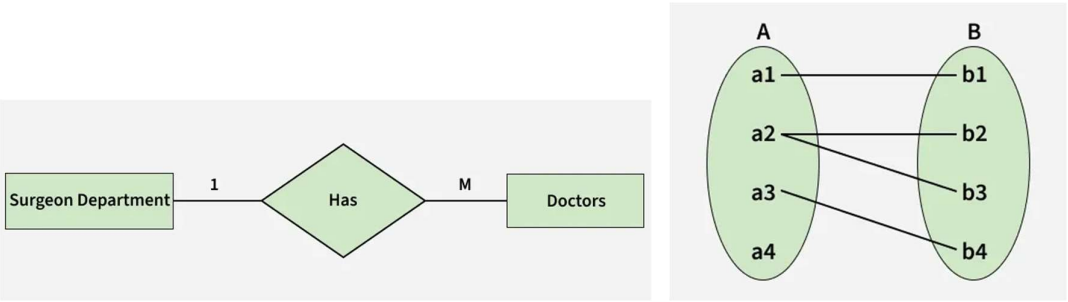
Svaki entitet iz oba skupa učestvuje najviše jednom. Npr. osobi se izdaje samo jedan pasoš, a pasoš pripada samo jednoj osobi.



Slika 26. *Veza 1:1 – ER prikaz i prikaz pomoću skupova*

2. Jedan-prema-više (1:N)

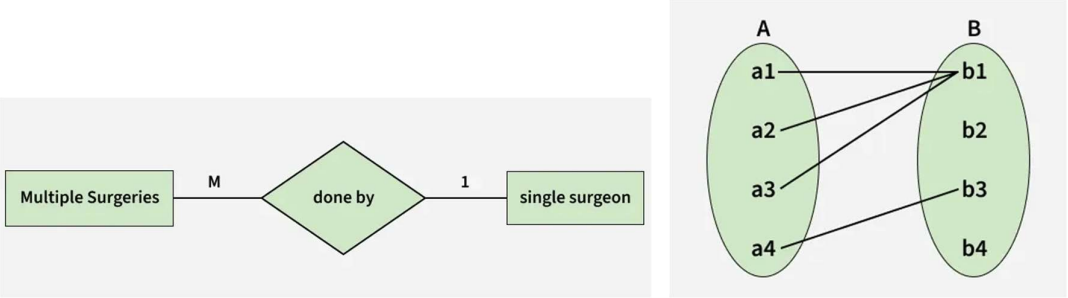
Jedan entitet može biti povezan sa više entiteta. Npr. odeljenje hirurgije može imati više lekara, a lekar pripada jednom odeljenju.



Slika 27. *Veza 1:N – ER prikaz i prikaz pomoću skupova*

3. Više-prema-jedan (N:1)

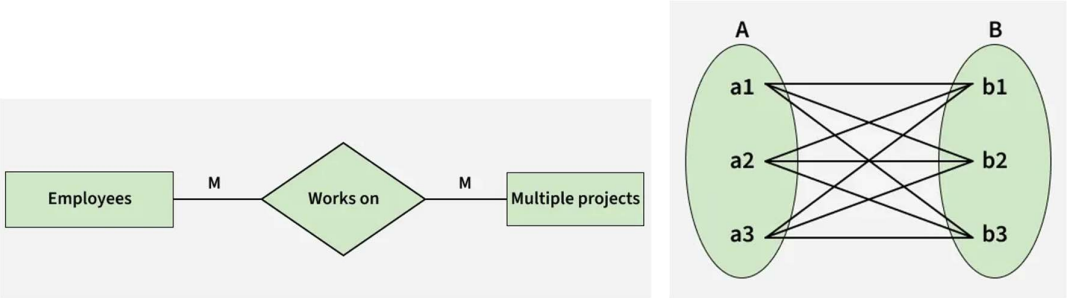
Entiteti iz jednog skupa učestvuju najviše jednom, a entiteti iz drugog više puta. Npr. hirurg može izvesti više operacija, ali svaku operaciju izvodi samo jedan hirurg.



Slika 28. *Veza N:1 – ER prikaz i prikaz pomoću skupova*

4. Više-prema-više (M:N)

Entiteti iz oba skupa mogu učestvovati više puta. Npr. zaposleni može raditi na više projekata, a na projektu radi više zaposlenih.

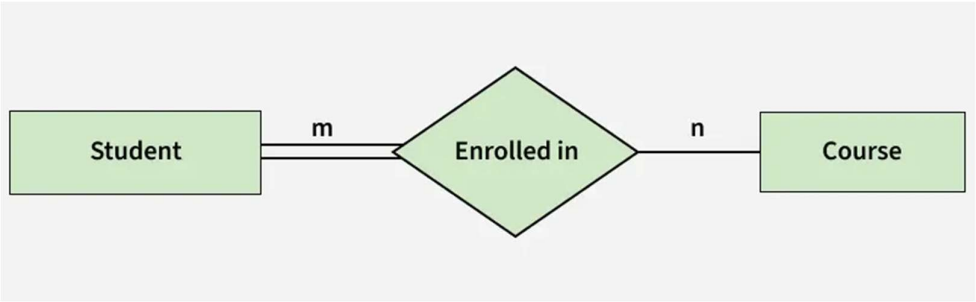


Slika 29. *Veza M:N – ER prikaz i prikaz pomoću skupova*

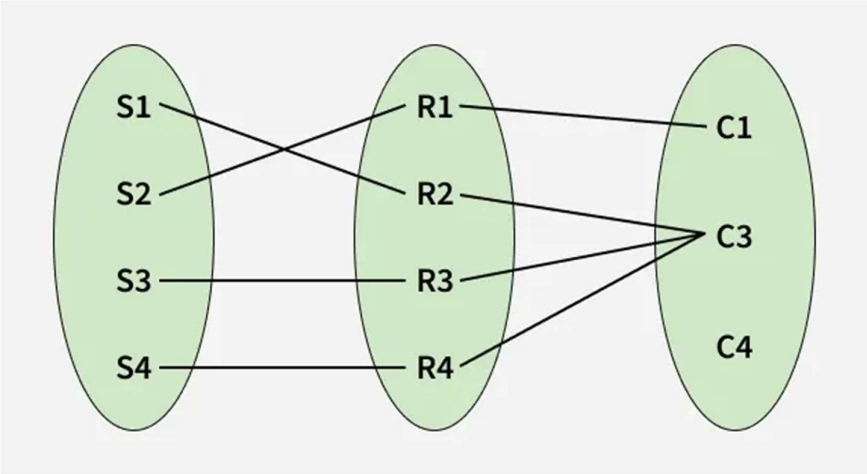
U prikazu pomoću skupova, *a1* je povezan sa *b1*, *b2* i *b3*, a *b3* sa *a1*, *a2* i *a3* – to je veza **više-prema-više**.

5.10 Ograničenje učešća (participacije)

- **Potpuno (totalno) učešće:** svaki entitet iz skupa mora učestvovati u vezi. Ako svaki student mora biti upisan na bar jedan predmet, učešće studenata je potpuno. Crta se **dvostrukom linijom**.
- **Delimično (parcijalno) učešće:** entitet može, ali ne mora učestvovati u vezi. Ako postoje predmeti na koje niko nije upisan, učešće predmeta je delimično. Crta se **jednostrukom linijom**.



Slika 30. *Student – potpuno učešće, Predmet – delimično učešće*



Slika 31. *Isto pomoću skupova: predmet C4 ne učestvuje u vezi*

5.11 Kako nacrtati ER dijagram – koraci

1. **Identifikujte entitete** i predstavite ih pravougaonicima.
2. **Identifikujte veze** između entiteta i predstavite ih rombovima. Veze se ne povezuju direktno jedna sa drugom.
3. **Dodajte atribute** (elipse) i povežite ih sa odgovarajućim entitetom.
4. **Definišite primarne ključeve** – podvučeni atributi koji jedinstveno identifikuju instancu.
5. **Uklonite redundanse** – nepotrebne ili ponovljene entitete i veze.
6. **Proverite jasnoću** – dijagram treba jasno da prenosi veze između entiteta.

6. Strukturna ograničenja veza u ER modelu

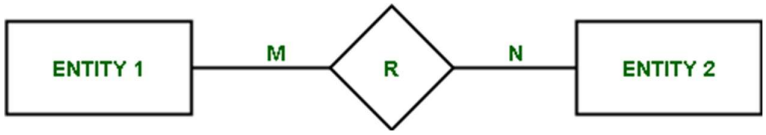
6.1 Pojam strukturnih ograničenja

Strukturna ograničenja određuju način na koji entiteti učestvuju u vezama i daju osnovne smernice za projektovanje interakcija između entiteta. Dva osnovna tipa su **kardinalnost** (broj instanci u vezi) i **učešće** (da li u vezi učestvuju sve ili samo neke instance). Kardinalnost je *1:1*, *1:N* ili *M:N*, a učešće *potpuno* ili *delimično*.

Ova ograničenja obezbeđuju da model podataka verno odražava stvarno stanje, pa je njihovo razumevanje ključno za projektovanje stabilnih i funkcionalnih baza.

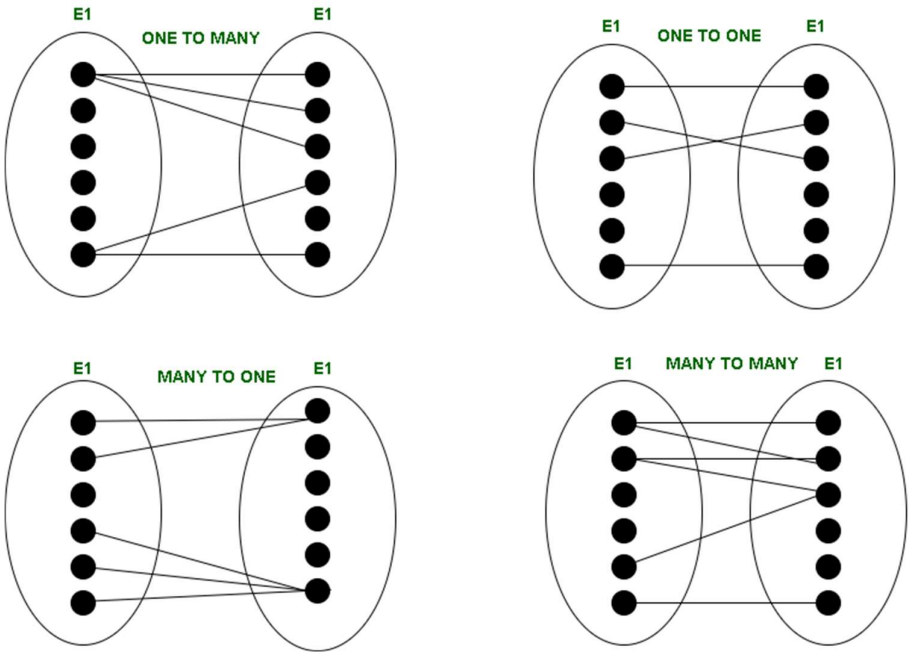
6.2 Kardinalni odnosi

Entiteti se označavaju pravougaonikom, a veze rombom. Brojevi iznad linija (npr. **M** i **N**) nazivaju se **kardinalni odnosi** i predstavljaju maksimalan broj entiteta koji mogu biti povezani vezom *R*.



Slika 32. *Kardinalni odnos M:N*

6.3 Tipovi kardinalnosti



Slika 33. *Četiri tipa kardinalnosti*

Tabela 6. *Tipovi kardinalnosti*

Tip	Opis
1:1	Svaki entitet u oba skupa učestvuje najviše jednom.

Tip	Opis
1:N	Entiteti prvog skupa učestvuju najviše jednom, a drugog više puta.
N:1	Entiteti prvog skupa učestvuju više puta, a drugog najviše jednom.
M:N	Entiteti iz oba skupa učestvuju više puta.

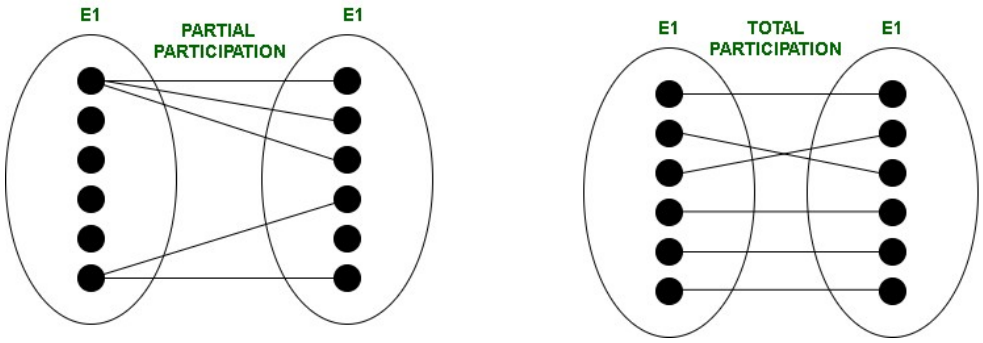
6.4 Ograničenja učešća

Potpuno (totalno) učešće

Svaka instanca entiteta mora učestvovati u bar jednoj instanci veze. Npr. ako svaki student mora biti upisan na bar jedan predmet, ili ako svaki profesor predaje bar jedan predmet. Nijedan entitet ne može postojati bez učešća u vezi.

Delimično (parcijalno) učešće

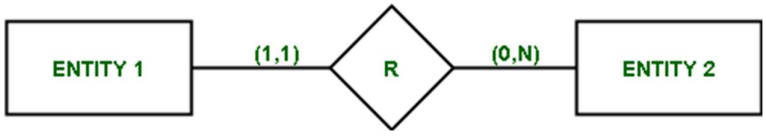
Neki entiteti iz skupa ne učestvuju u vezi; crta se jednostrukom linijom. Npr. ako neki profesori ne predaju nijedan predmet, njihovo učešće u vezi *„Predaje"* je delimično. Slično, knjiga u biblioteci može, ali ne mora biti pozajmljena, pa je učešće *Knjige* u vezi sa *Članom* delimično.



Slika 34. *Delimično i potpuno učešće prikazano skupovima*

6.5 Strukturna ograničenja i (min, max) notacija

Kada se kardinalnost i ograničenja učešća razmatraju zajedno, oni čine **strukturna ograničenja** (strukturna svojstva). To su restrikcije koje se moraju nametnuti nad podacima da bi baza bila usaglašena sa zahtevima.



Slika 35. *(min, max) notacija na liniji veze*

Strukturna ograničenja se zapisuju **(min, max) notacijom** – parom brojeva **(m, n)** na liniji između entiteta i veze, gde je **m** minimalan, a **n** maksimalan broj učešća entiteta u vezi:

- **m = 0** → učešće je **delimično**.
- **m ≥ 1** → učešće je **potpuno**.

> 💡 Broj puta koliko entitet učestvuje u vezi jednak je broju pojavljivanja tog entiteta u torkama skupa veza.

7. Generalizacija, specijalizacija i agregacija

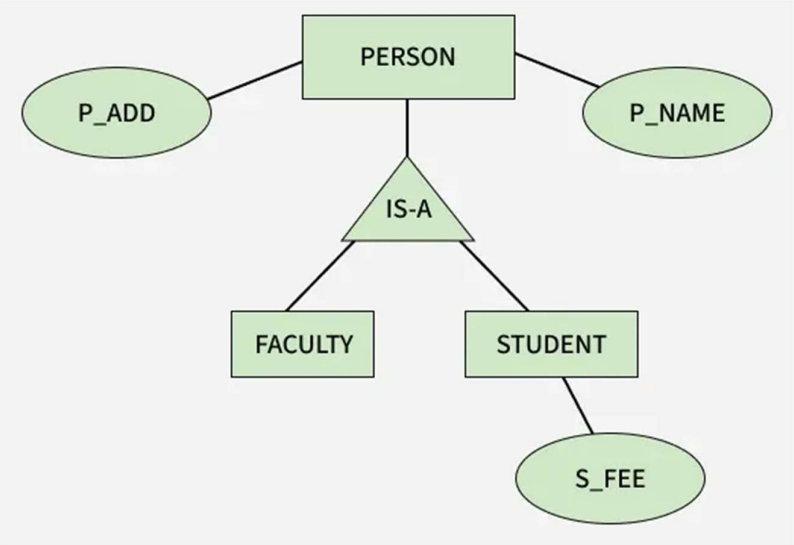
Korišćenje ER modela za veće skupove podataka unosi veliku složenost. Da bi se ona smanjila, uvedeni su koncepti **generalizacije**, **specijalizacije** i **agregacije**. Oni služe za **apstrakciju podataka** – skrivanje detalja o skupu objekata.

7.1 Generalizacija

PRISTUP ODOZDO NAGORE

Generalizacija je proces izdvajanja zajedničkih svojstava iz više skupova entiteta i kreiranja generalizovanog entiteta višeg nivoa. Dva ili više entiteta sa zajedničkim atributima spajaju se u entitet višeg nivoa.

💡 Primer: *STUDENT* i *FACULTY* (nastavno osoblje) generalizuju se u *PERSON* (Osoba). Zajednički atributi *P_NAME* i *P_ADD* prelaze u *PERSON*, a specifični atribut *S_FEE* (školarina) ostaje u *STUDENT*.



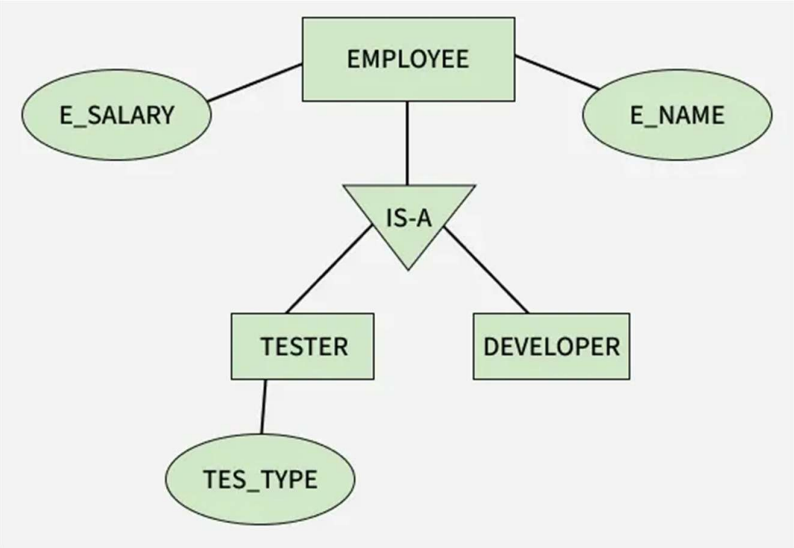
Slika 36. *Generalizacija: STUDENT i FACULTY → PERSON*

7.2 Specijalizacija

PRISTUP ODOZGO NADOLE

Specijalizacija je proces u kom se entitet višeg nivoa deli na dva ili više pod-entiteta nižeg nivoa na osnovu svojih karakteristika.

💡 Primer: *EMPLOYEE* (Zaposleni) se specijalizuje u *DEVELOPER* i *TESTER*. Zajednički atributi *E_NAME* i *E_SALARY* ostaju u *EMPLOYEE*, a specifičan atribut *TES_TYPE* (tip testiranja) pripada entitetu *TESTER*.

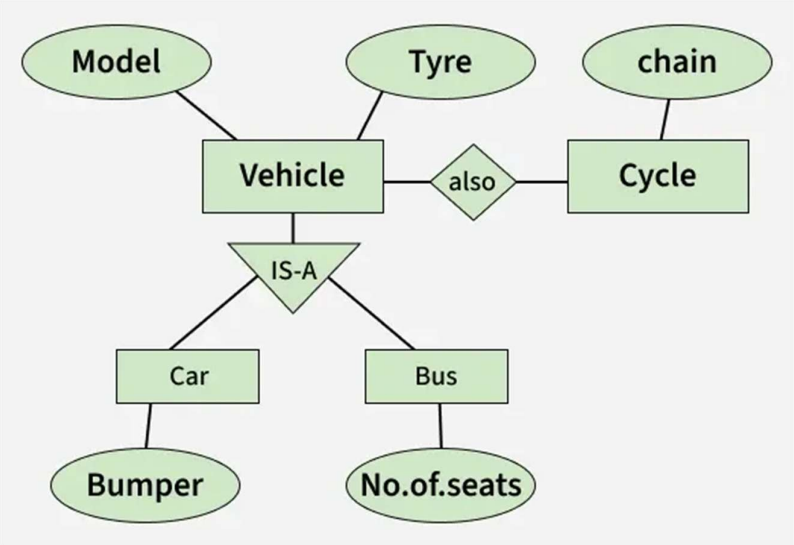


Slika 37. *Specijalizacija: EMPLOYEE → TESTER, DEVELOPER*

7.3 Nasleđivanje

Nasleđivanje je važna osobina generalizacije i specijalizacije: pod-entiteti ponovo koriste svojstva roditeljskog entiteta.

- **Nasleđivanje atributa:** entiteti nižeg nivoa nasleđuju atribute entiteta višeg nivoa, ali ne i obrnuto. Npr. *Car* je podklasa *Vehicle* i nasleđuje njegove atribute.
- **Nasleđivanje veza:** pod-entiteti nasleđuju i veze roditeljskog entiteta.
- **Redefinisanje (overriding):** pod-entiteti mogu redefinisati postojeće ili dodati sopstvene atribute i ponašanja.
- **Nasleđivanje učešća:** ograničenja učešća prenose se sa superklase na podklasu, pa podklase poštuju ista pravila učešća.



Slika 38. *Nasleđivanje: Car i Bus nasleđuju atribute entiteta Vehicle*

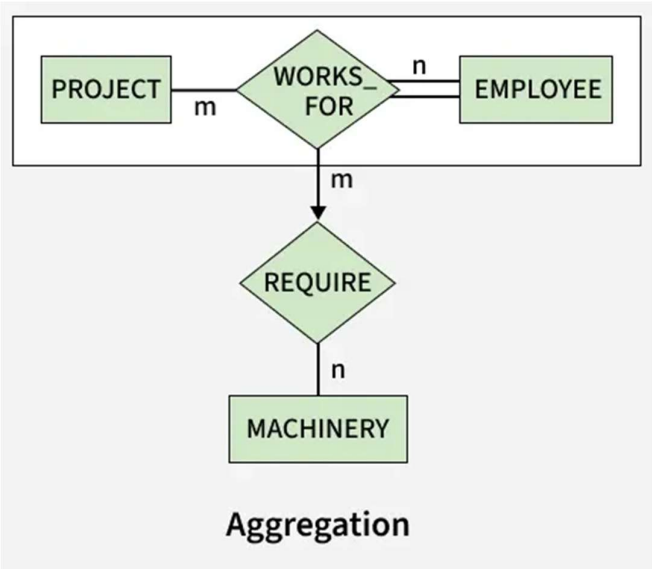
💡 Na dijagramu *Vehicle* ima vezu sa entitetom *Cycle*. Nasleđivanje učešća prenosi samo ograničenja učešća na podklase, a ne same konkretne veze između entiteta.

7.4 Agregacija

ER dijagram standardno ne može da prikaže vezu između entiteta i druge veze. U takvim slučajevima se veza, zajedno sa entitetima koji u njoj učestvuju, **agregira u entitet višeg nivoa**.

DEFINICIJA

Agregacija je proces apstrakcije kojim se veze predstavljaju kao skupovi entiteta višeg nivoa.



Slika 39. *Agregacija: (EMPLOYEE – WORKS_FOR – PROJECT) REQUIRE MACHINERY*

💡 Primer: zaposlenom koji radi na projektu potrebna je određena mašinerija. Potrebna je veza *REQUIRE* između veze *WORKS_FOR* i entiteta *MACHINERY*. Agregacijom se *WORKS_FOR* zajedno sa *EMPLOYEE* i *PROJECT* posmatra kao jedan entitet višeg nivoa, a zatim se *REQUIRE* povezuje sa *MACHINERY*.

7.5 Predstavljanje agregacije relacionom šemom

1. **Šema za agregiranu vezu** – veza se tretira kao skup entiteta; sadrži primarne ključeve entiteta koji u njoj učestvuju i sve opisne atribute te veze.
2. **Šema za vezu višeg nivoa** – sadrži primarni ključ šeme agregirane veze, primarni ključ povezanog entiteta i sve dodatne opisne atribute.

8. Zadaci za vežbu

Pet zadataka je poređano od lakšeg ka težem, i svaki uvodi bar jedan novi element ER modela. Za svaki zadatak nacrtajte ER dijagram: označite ključeve, vrste atributa, kardinalnost i učešće. Kod zadatka 3 koristite isključivo (min, max) notaciju. Rešenja su data posle svakog zadatka.

Kod IS-A trougla oznaka **d** znači disjunktnu, a **o** preklapajuću specijalizaciju; dvostruka linija do trougla znači potpunu specijalizaciju. Isprekidano podvučen atribut je delimični ključ slabog entiteta.

Tabela 7. *Koje elemente ER modela pokriva koji zadatak*

Element ER modela	Z1	Z2	Z3	Z4	Z5
Ključni i obični atributi	✓	✓	✓	✓	✓
Složeni atribut	✓				✓
Viševrednosni atribut	✓	✓		✓	✓
Izvedeni atribut	✓			✓	
Atribut na vezi	✓	✓	✓	✓	✓
Veza 1:1		✓			
Veza 1:N / N:1	✓	✓		✓	✓
Veza M:N	✓	✓	✓	✓	✓
Slabi entitet		✓			
Rekurzivna veza sa ulogama		✓	✓		
Ternarna veza			✓		
(min, max) notacija			✓		
Specijalizacija d / o				✓	
Agregacija					✓

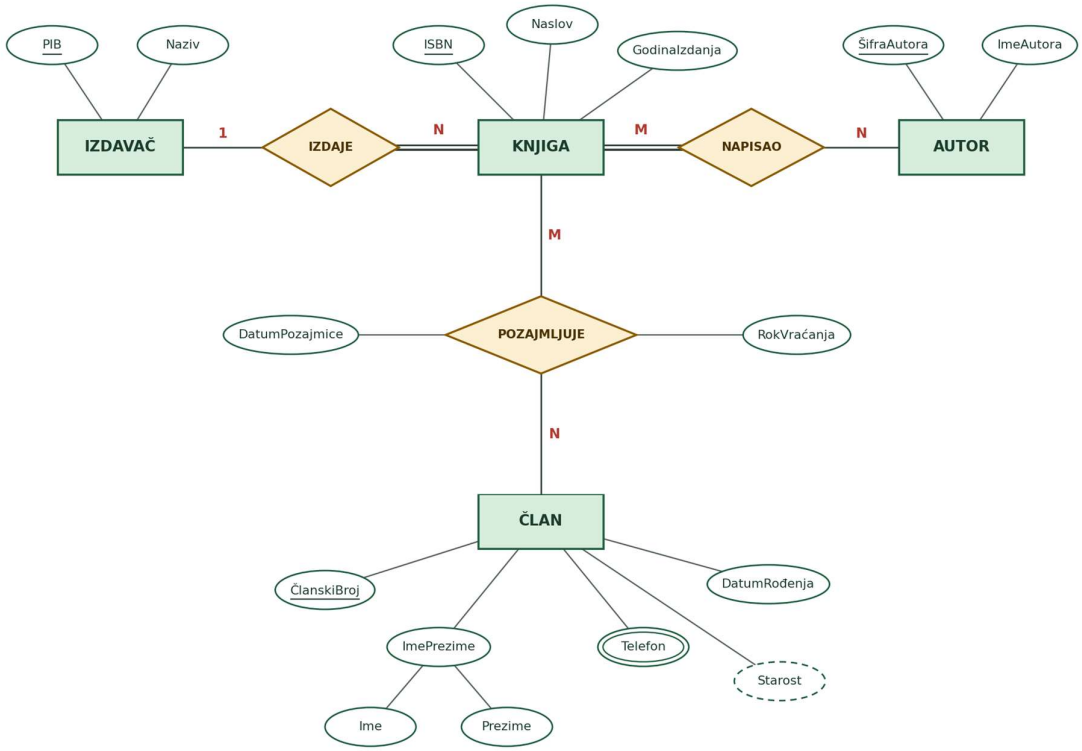
8.1 Zadatak 1 – Gradska biblioteka

Biblioteka vodi evidenciju o knjigama, autorima, izdavačima i članovima. Za svaku knjigu pamte se ISBN (jedinstven), naslov i godina izdanja. Svaku knjigu izdao je tačno jedan izdavač, a izdavač može izdati više knjiga; u bazi mogu postojati i izdavači čije knjige biblioteka još nema. Za izdavača se pamte PIB i naziv.

Knjigu je napisao jedan ili više autora, a autor može napisati više knjiga; u bazi mogu biti i autori bez ijedne knjige u biblioteci. Za autora se pamte šifra i ime. Za člana se pamte članski broj, ime i prezime (posebno ime, posebno prezime), jedan ili više brojeva telefona, datum rođenja i starost, koja se izračunava iz datuma rođenja. Član može pozajmiti više knjiga, a knjigu tokom vremena pozajmljuje više članova; za svaku pozajmicu pamte se datum pozajmice i rok vraćanja.

Rešenje

Rešenje 1 – Gradska biblioteka



Slika 40. *Rešenje zadatka 1 – ER dijagram gradske biblioteke*

- **IZDAJE (1:N):** knjigu izdaje tačno jedan izdavač, pa je 1 pored IZDAVAČ. Knjiga ima potpuno učešće (dvostruka linija), izdavač delimično, jer može postojati bez knjiga.
- **NAPISAO (M:N):** knjiga mora imati bar jednog autora (potpuno), autor ne mora imati knjigu u biblioteci (delimično).
- **POZAJMLJUJE (M:N):** DatumPozajmice i RokVraćanja su atributi veze, jer pripadaju paru član–knjiga, a ne knjizi ili članu posebno. Oba učešća su delimična.
- **ČLAN:** ČlanskiBroj je ključ, ImePrezime je složen (Ime, Prezime), Telefon viševrednosni (dvostruka elipsa), Starost izvedena iz DatumRođenja (isprekidana elipsa).

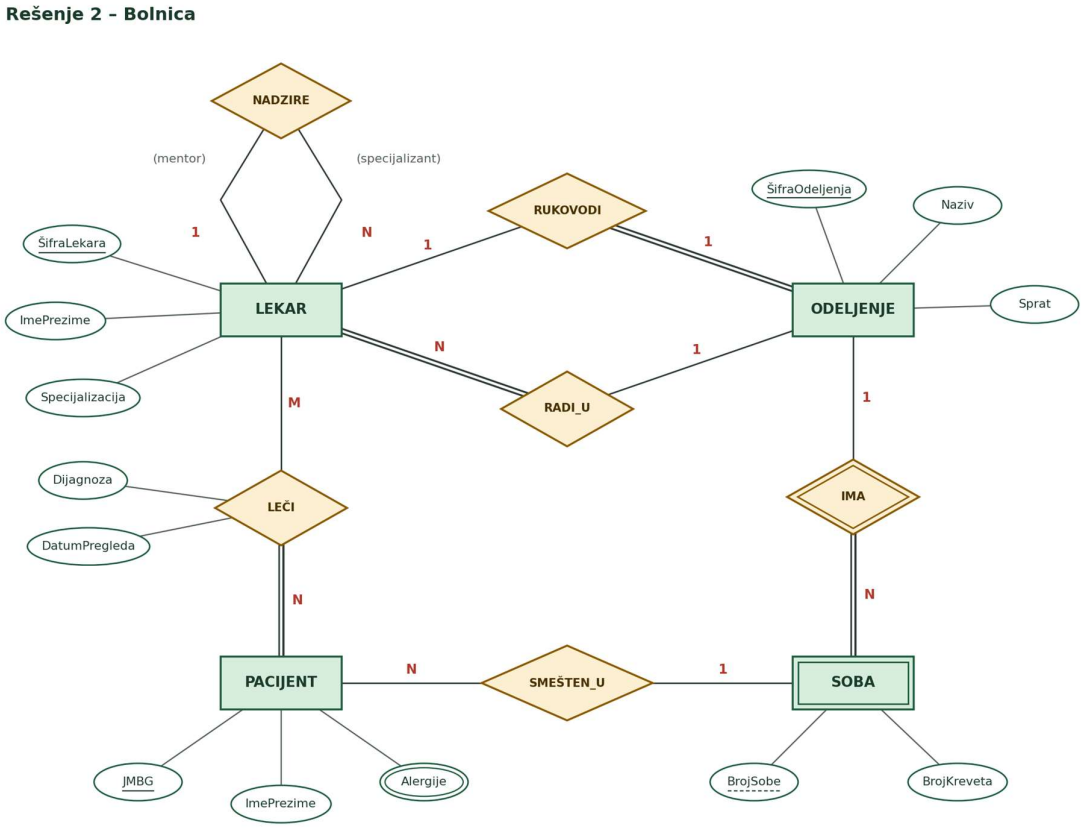
> 💡 Pitanje za diskusiju: kako bi se model promenio da biblioteka ima više primeraka iste knjige? Uvodi se slabi entitet PRIMERAK (delimični ključ InventarniBroj) vezan za KNJIGA, a pozajmljuje se primerak, ne knjiga.

8.2 Zadatak 2 – Bolnica

Bolnica ima odeljenja (šifra, naziv, sprat). Svaki lekar (šifra, ime i prezime, specijalizacija) radi na tačno jednom odeljenju, a na odeljenju radi više lekara. Svakim odeljenjem rukovodi tačno jedan lekar, a lekar rukovodi najviše jednim odeljenjem. Iskusniji lekar može biti mentor više specijalizanata, a svaki specijalizant ima najviše jednog mentora.

Odeljenje ima sobe. Broj sobe je jedinstven samo unutar odeljenja (soba 12 postoji na više odeljenja), a pamti se i broj kreveta. Za pacijenta se pamte JMBG, ime i prezime i alergije (može ih biti više). Pacijent je smešten u najviše jednu sobu, a u sobi može biti više pacijenata. Svakog pacijenta leči bar jedan lekar, a lekar leči više pacijenata; za svako lečenje beleže se dijagnoza i datum pregleda.

Rešenje



Slika 41. *Rešenje zadatka 2 – ER dijagram bolnice*

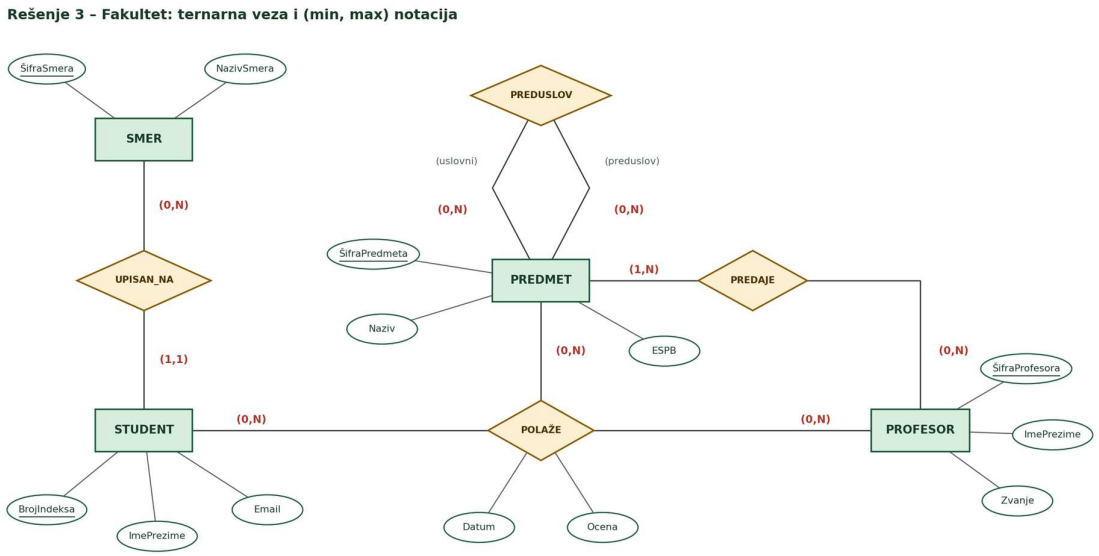
- **SOBA je slabi entitet:** BrojSobe nije jedinstven sam po sebi, pa je delimični ključ (isprekidano podvučen). Identifikaciona veza IMA je dvostruki romb, a SOBA ima potpuno učešće. Puni ključ sobe je ŠifraOdeljenja + BrojSobe.
- **Dve veze između istih entiteta:** RADI_U (N:1, lekar potpuno) i RUKOVODI (1:1, odeljenje potpuno jer svako odeljenje ima šefa, lekar delimično jer većina lekara nije šef).
- **NADZIRE je rekurzivna 1:N veza** na entitetu LEKAR; uloge mentor i specijalizant upisane su na linijama, da se zna koja strana je 1, a koja N.
- **LEČI (M:N)** nosi Dijagnoza i DatumPregleda; pacijent ima potpuno učešće jer ga leči bar jedan lekar. Alergije su viševrednosni atribut.
- **SMEŠTEN_U (N:1):** pacijent je u najviše jednoj sobi, pa je učešće delimično (ambulantni pacijent nema sobu).

8.3 Zadatak 3 – Fakultet (min, max)

Svaki student (broj indeksa, ime i prezime, email) upisan je na tačno jedan smer (šifra, naziv); na smeru može biti nijedan ili više studenata. Predmet (šifra, naziv, ESPB) predaje bar jedan profesor, a profesor (šifra, ime i prezime, zvanje) može predavati nijedan ili više predmeta.

Evidentira se polaganje ispita: student polaže određeni predmet kod određenog profesora, pri čemu se pamte datum i ocena. Predmet može imati više predmeta kao preduslov i sam može biti preduslov za više drugih predmeta.

Rešenje



Slika 42. *Rešenje zadatka 3 – ER dijagram fakulteta u (min, max) notaciji*

- **Čitanje (min, max):** par pored entiteta kaže koliko puta TAJ entitet učestvuje u vezi. STUDENT (1,1) u UPISAN_NA znači tačno jedan smer; SMER (0,N) znači nijedan ili više studenata.
- **PREDAJE:** PREDMET (1,N) jer ga predaje bar jedan profesor; PROFESOR (0,N) jer ne mora ništa da predaje.
- **POLAŽE je ternarna veza.** Datum i Ocena pripadaju kombinaciji student–predmet–profesor. Tri binarne veze ne bi mogle da kažu kod kog profesora je student položio koji predmet.
- **PREDUSLOV je rekurzivna M:N veza** na entitetu PREDMET, sa ulogama uslovni i preduslov; obe strane (0,N).

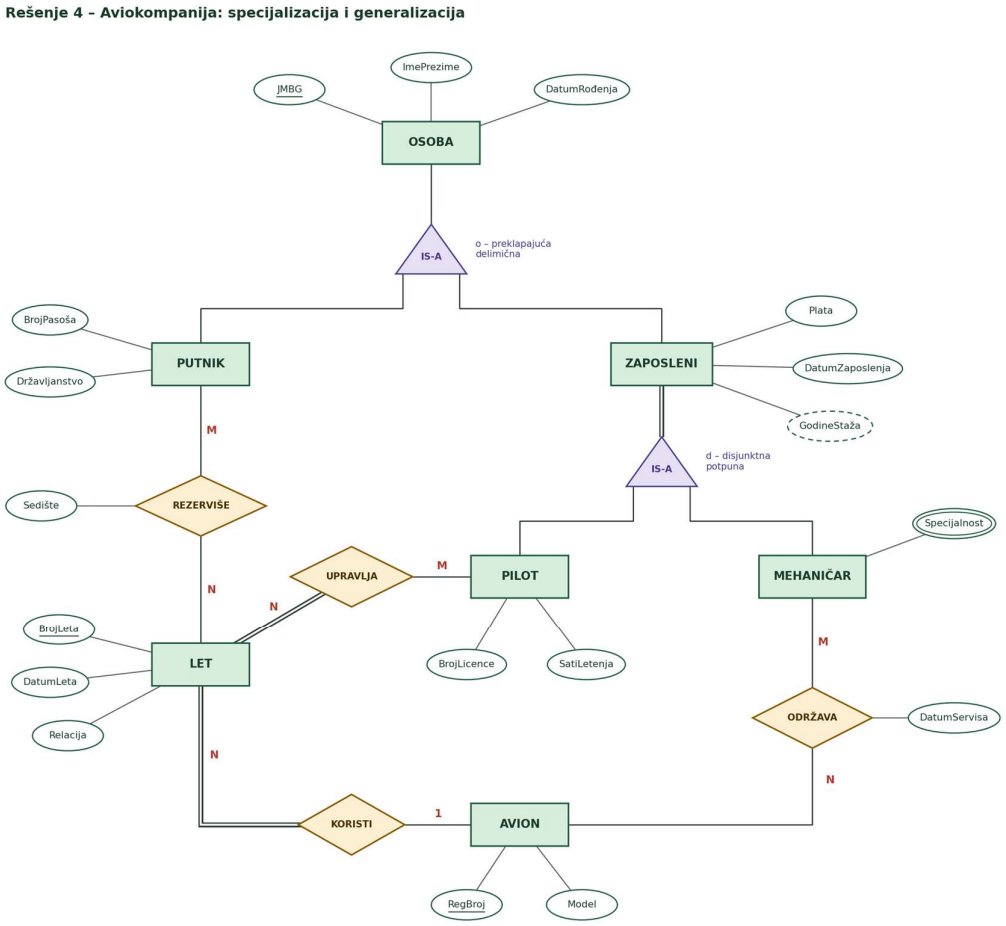
> 💡 Česta greška: (1,1) upisan pored SMER umesto pored STUDENT, kao u notaciji sa odnosima 1:N. U (min, max) notaciji oznake stoje na suprotnoj strani veze.

8.4 Zadatak 4 – Aviokompanija

O svakoj osobi pamte se JMBG, ime i prezime i datum rođenja. Osoba može biti putnik, zaposleni ili oboje (zaposleni može leteti kao putnik), a u bazi mogu postojati i osobe koje nisu ni jedno ni drugo. Za putnika se pamte broj pasoša i državljanstvo, a za zaposlenog plata, datum zaposlenja i godine staža, koje se izračunavaju.

Svaki zaposleni je ili pilot ili mehaničar, ne može biti oba. Pilot ima broj licence i sate letenja, a mehaničar jednu ili više specijalnosti. Svakim letom (broj leta, datum, relacija) upravlja jedan ili više pilota, a pilot upravlja mnogim letovima. Svaki let koristi tačno jedan avion (registarski broj, model), a avion se koristi za više letova. Mehaničari održavaju avione (jedan mehaničar više aviona, jedan avion više mehaničara), uz datum servisa. Putnik rezerviše letove, a za rezervaciju se pamti sedište.

Rešenje



Slika 43. *Rešenje zadatka 4 – ER dijagram aviokompanije sa IS-A hijerarhijom*

- **Prvi IS-A (OSOBA → PUTNIK, ZAPOSLENI): o, delimična.** Preklapajuća, jer zaposleni može biti i putnik; delimična, jer postoje osobe koje nisu ni jedno ni drugo (jednostruka linija do trougla).
- **Drugi IS-A (ZAPOSLENI → PILOT, MEHANIČAR): d, potpuna.** Disjunktna, jer ne može oba; potpuna, jer svaki zaposleni mora biti jedan od njih (dvostruka linija).
- **Nasleđivanje:** PILOT i MEHANIČAR nemaju svoj ključ, nasleđuju JMBG od OSOBA preko ZAPOSLENI, kao i Platu i DatumZaposlenja. GodineStaža je izvedeni, Specijalnost viševrednosni atribut.
- **Veze sa podtipovima:** UPRAVLJA vezuje PILOT (a ne ZAPOSLENI) za LET, jer mehaničari ne upravljaju letovima; ODRŽAVA vezuje MEHANIČAR za AVION. LET ima potpuno učešće u UPRAVLJA i KORISTI.

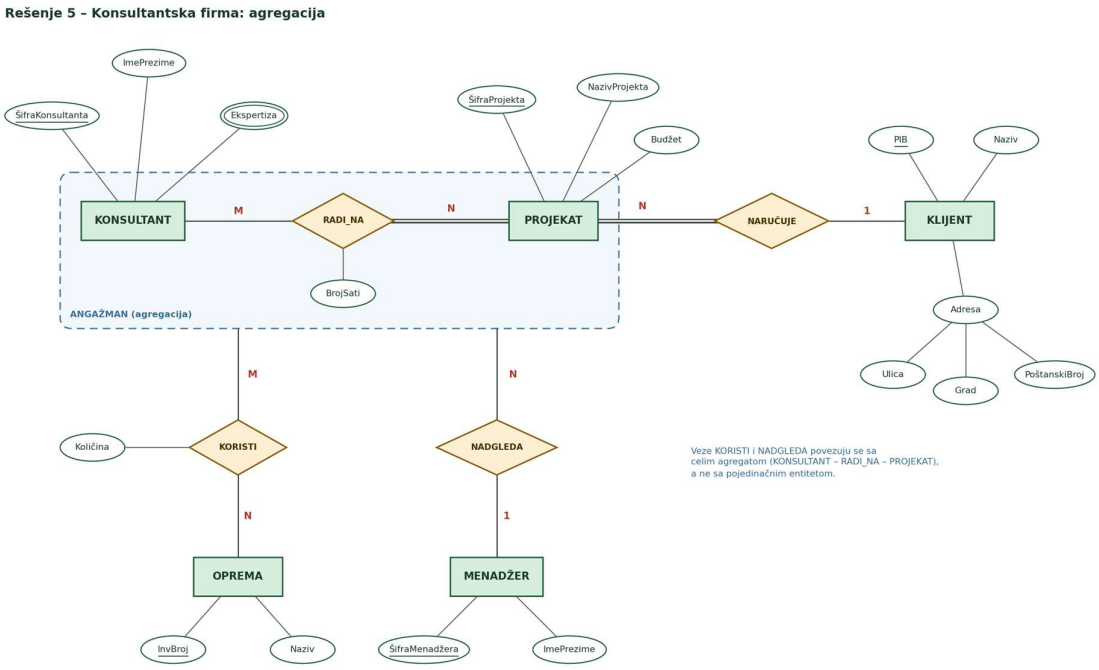
- **REZERVIŠE (M:N)** povezuje PUTNIK sa LET, a Sedište je atribut veze.

8.5 Zadatak 5 – Konsultantska firma

Konsultant (šifra, ime i prezime, jedna ili više ekspertiza) radi na projektima (šifra, naziv, budžet). Na projektu radi više konsultanata, svaki projekat ima bar jednog konsultanta, a za svako angažovanje pamti se broj sati. Svaki projekat naručuje tačno jedan klijent (PIB, naziv, adresa sa ulicom, gradom i poštanskim brojem); klijent može naručiti više projekata.

Konsultant na određenom projektu koristi opremu (inventarski broj, naziv), pri čemu se pamti količina; ista oprema može se koristiti na više angažmana. Svaki angažman (konsultant na projektu) nadgleda tačno jedan menadžer (šifra, ime i prezime), a menadžer nadgleda više angažmana.

Rešenje



Slika 44. *Rešenje zadatka 5 – ER dijagram konsultantske firme sa agregacijom*

- **Zašto agregacija:** oprema i menadžer ne vezuju se za konsultanta niti za projekat posebno, već za konkretan angažman (ovaj konsultant na ovom projektu). Zato se KONSULTANT – RADI_NA – PROJEKAT uokviruje u entitet višeg nivoa ANGAŽMAN.
- **RADI_NA (M:N)** nosi BrojSati; PROJEKAT ima potpuno učešće jer ima bar jednog konsultanta.
- **KORISTI (M:N)** povezuje agregat sa OPREMA, uz atribut Količina. **NADGLEDA (N:1):** svaki angažman ima tačno jednog menadžera.
- **NARUČUJE (N:1):** projekat naručuje tačno jedan klijent (potpuno učešće projekta). Adresa klijenta je složeni atribut (Ulica, Grad, PoštanskiBroj), Ekspertiza konsultanta viševrednosni.
- **Relaciona šema agregata:** ANGAŽMAN(ŠifraKonsultanta, ŠifraProjekta, BrojSati); KORISTI(ŠifraKonsultanta, ŠifraProjekta, InvBroj, Količina). Ključ agregata ulazi u tabelu veze višeg nivoa.

DEO III

Relacioni model

9. Uvod u relacioni model

9.1 Šta je relacioni model?

> DEFINICIJA
> **Relacioni model** organizuje podatke u tabele, koje se nazivaju **relacije**. Svaka tabela ima više kolona, a svaka kolona ima jedinstveno ime. Model je uveo dr Edgar F. Kod (*Edgar F. Codd*), a njegova **Kodova pravila** određuju šta baza mora ispuniti da bi bila prava relaciona baza.

Relacioni model pojednostavljuje skladištenje, pretraživanje i upravljanje podacima pomoću redova i kolona, i obezbeđuje tačnost podataka. Za rad sa podacima prvenstveno se koristi **SQL** (*Structured Query Language*). NoSQL baze nisu deo relacionog modela i često se koriste kao alternativa za nestrukturirane podatke, ali relacione baze sa SQL-om ostaju standard tamo gde su potrebni jaka konzistentnost i složeni upiti.

- **Jednostavnost:** jednostavna implementacija i operacije nad podacima.
- **Povezivanje:** primarni i strani ključevi povezuju tabele.
- **Normalizacija:** teorija normalizacije služi za projektovanje modela bez redundanse.
- **Obrada podataka:** relaciona algebra i relacioni račun služe za manipulaciju relacijama.

9.2 Terminologija

Tabela 8. *Osnovni pojmovi relacionog modela*

Pojam	Značenje
Relacija (tabela)	Osnovna struktura u kojoj se čuvaju podaci; sastoji se od redova i kolona.
Relaciona šema	Struktura relacije: naziv i atributi, npr. STUDENT(StudentID, Name, Age, Course).
Relaciona instanca	Skup vrednosti u relaciji u određenom trenutku.
Atribut (kolona)	Svojstvo kojim je relacija definisana, npr. StudentID, Name.
Domen atributa	Skup svih dozvoljenih vrednosti atributa, npr. važeće godine za Age.
Torka (red)	Jedan red u relaciji.
Kardinalnost	Broj torki (redova) u relaciji; za kolonu, broj različitih vrednosti u odnosu na broj redova.
Stepen (arnost)	Broj atributa (kolona) relacije; tabela sa 4 kolone ima stepen 4.
Primarni ključ	Atribut ili skup atributa koji jedinstveno identifikuje svaku torku.

Pojam	Značenje
NULL vrednost	Nepoznata, nedostajuća ili nedefinisana vrednost; dve NULL vrednosti smatraju se različitim.

Tabela 9. *Primer relacije STUDENT: 4 torke, stepen 4*

StudentID	Name	Age	Course
1	Vaibhav Garg	21	Computer Science
2	Komal Gupta	22	Mathematics
3	Varnika Gangwar	21	Physics
4	Shubhi Chandra	23	Computer Science

Ista baza zapisana relacionom šemom (primarni ključevi su podvučeni):

- STUDENT (StudNo, Sname, Special)
- ENROLLMENT (StudNo, Subcode, Marks)
- SUBJECT (Subcode, Subname, Maxmarks, Faccode)
- FACULTY (Faccode, Fname, Dept)

9.3 Najpoznatiji RDBMS sistemi

Tabela 10. *Proizvođači relacionih SUBP-a*

Proizvođač / sistem	Osobine
Oracle	Jedan od najzastupljenijih; robusnost, skalabilnost, pouzdanost.
Microsoft SQL Server	Popularan u Windows okruženju; napredna analitika i izveštavanje.
IBM DB2	Velika enterprise okruženja; visoka dostupnost i oporavak od havarija.
MySQL	Open-source; jednostavan, fleksibilan, jeftin – male i srednje kompanije.
PostgreSQL	Open-source; napredna podrška za složene transakcije i skalabilnost.
SAP HANA	In-memory RDBMS za analitiku visokih performansi u realnom vremenu.

10. Relaciona algebra

Relaciona algebra je **proceduralni jezik**: skup operatora koji se primenjuju nad relacijama i daju nove relacije. Ona je osnova visokonivojskih upitnih jezika kao što je SQL.

Tabela 11. *Devet osnovnih operatora relacione algebre*

Operator	Simbol	Zapis	Rezultat
Unija (UNION)	∪	A ∪ B	Sve torke iz obe relacije, bez duplikata
Presek (INTERSECTION)	∩	A ∩ B	Samo torke koje su u obe relacije
Razlika (MINUS)	−	A − B	Torke iz A koje nisu u B
Dekartov proizvod	×	A × B	Svi parovi torki; m × n torki
Selekcija	σ	σ uslov (R)	Torke koje zadovoljavaju uslov
Projekcija	π	π kol1, kol2 (R)	Samo izabrane kolone
Spajanje (JOIN)	⋈	R ⋈ S	Torke spojene po povezanim atributima (inner, outer, left, right)
Deljenje (DIVISION)	÷	A ÷ B	Torke iz A povezane sa svim torkama iz B
Preimenovanje (RENAME)	ρ	ρ NOVO_IME (R)	Novo ime relacije ili atributa

> 💡 Primer: A = {clerk, manager, salesman}, B = {president, clerk, manager}. Tada je A ∪ B = {clerk, manager, salesman, president}, A ∩ B = {clerk, manager}, a A − B = {salesman}.

Prednosti

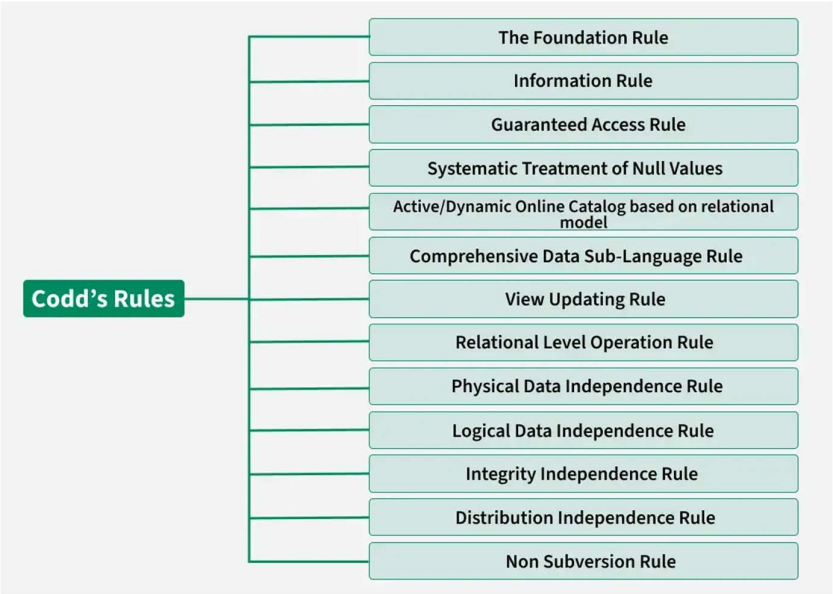
- **Jednostavnost i formalnost:** zasnovana na matematici, strog skup operatora.
- **Apstrakcija:** rad sa logičkom strukturom, bez brige o fizičkom zapisu.
- **Prenosivost:** nezavisna od konkretnog SUBP-a.
- **Efikasnost i proširivost:** pogodna za optimizaciju upita.

Nedostaci

- **Ograničena ekspresivnost:** složeni upiti zahtevaju složene izraze.
- **Ograničeni tipovi:** najbolje radi sa brojevima, tekstom, datumima; slabije sa multimedijom.
- **Nema direktne integracije:** potreban je dodatni sloj za rad iz programskih jezika.

11. Kodova pravila

Kod je definisao pravila kojima se proverava da li SUBP zaista ispunjava zahteve pravog relacionog sistema. U praksi većina komercijalnih sistema zadovoljava 8 do 9 pravila.



Slika 45. *Kodova pravila*

Tabela 12. *Pregled Kodovih pravila*

Br.	Pravilo	Suština
0	Osnovno pravilo	RDBMS mora upravljati bazom isključivo kroz svoje relacione mogućnosti.
1	Pravilo informacija	Svi podaci predstavljeni su na jedan način – kao vrednosti u ćelijama tabela.
2	Zagarantovan pristup	Svakom podatku se pristupa preko naziva tabele, primarnog ključa i naziva atributa.
3	Sistematski tretman NULL	NULL se dosledno tretira kao nepoznata ili neprimenjiva vrednost, bez obzira na tip.
4	Aktivni onlajn katalog	Metapodaci se čuvaju u katalogu dostupnom istim upitnim jezikom.
5	Sveobuhvatni podjezik	Bar jedan jezik za DDL, DML, integritet i transakcije.
6	Ažuriranje pogleda	Svi teorijski ažurni pogledi moraju se moći ažurirati i u sistemu.
7	Visokonivojske operacije	Unos, izmena i brisanje nad skupovima redova u jednom koraku.
8	Fizička nezavisnost	Promena fizičkog zapisa ne zahteva izmene aplikacija.
9	Logička nezavisnost	Promena logičke šeme ne utiče na aplikacije koje ne koriste izmenjene tabele direktno.

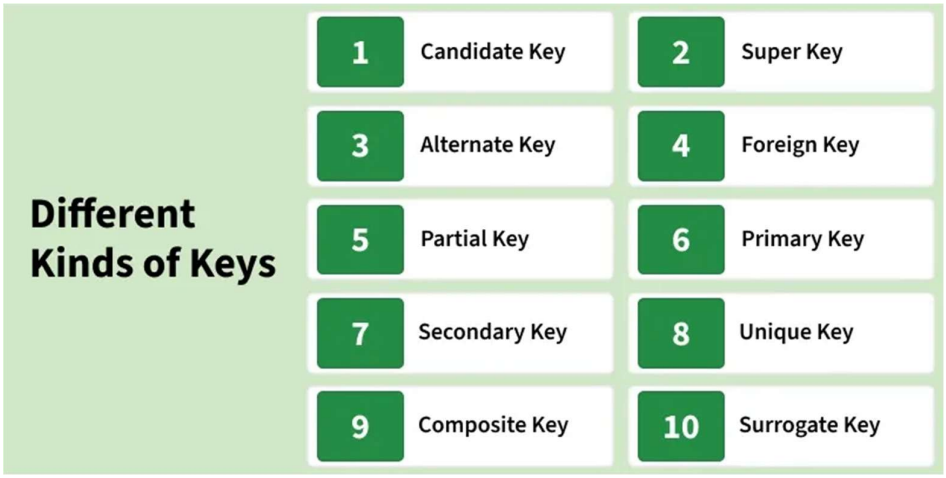
Br.	Pravilo	Suština
10	Nezavisnost integriteta	Ograničenja integriteta definišu se u bazi, ne u kodu aplikacije.
11	Nezavisnost od distribucije	Korisnik ne primećuje da su podaci raspoređeni na više lokacija.
12	Nesubverzija	Jezik niskog nivoa ne sme zaobići pravila integriteta.

ZAKLJUČAK

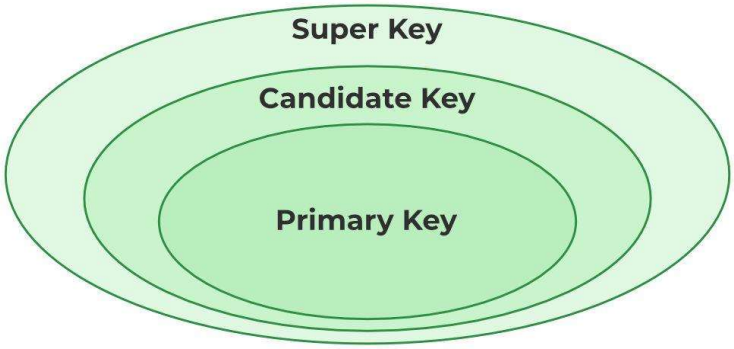
Relacioni model je pokazao da se podaci mogu čuvati u dvodimenzionalnim tabelama radi lakše manipulacije. Retko koji komercijalni sistem u potpunosti primenjuje svih 12 pravila, ali ona ostaju srž projektovanja relacionih baza i vodilja za skalabilne, sigurne i efikasne sisteme.

12. Ključevi u relacionom modelu

Ključevi jedinstveno identifikuju svaki red u tabeli, sprečavaju dupliranje i održavaju konzistentnost, i uspostavljaju veze između tabela. Bez njih bi pretraživanje velikih baza bilo sporo i podložno greškama.



Slika 46. *Deset vrsta ključeva*



Slika 47. *Svaki primarni ključ je kandidat za ključ, a svaki kandidat je superključ*

12.1 Kandidat za ključ (Candidate Key)

Minimalni skup atributa koji jedinstveno identifikuje torku, odnosno **minimalni superključ** bez suvišnih atributa. Svaka tabela ima bar jedan kandidatski ključ, a može ih imati više, ali samo jedan primarni. Na primer, STUD_NO u tabeli STUDENT, ili kombinacija {STUD_NO, COURSE_NO} kao složen kandidatski ključ tabele STUDENT_COURSE.

STUD_NO	SNAME	ADDRESS	PHONE
1	Tom	New York	123456789
2	John	Los Angeles	223365796
3	Alice	Chicago	175468965

Slika 48. *Tabela STUDENT*

STUD_NO	TEACHER_NO	COURSE_NO
1	001	C001
2	056	C005

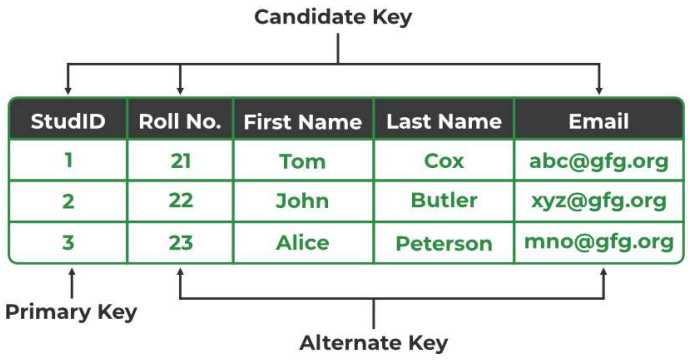
Slika 49. *Tabela STUDENT_COURSE*

12.2 Superključ (Super Key)

Skup od jednog ili više atributa koji jedinstveno identifikuje torku, ali može sadržati i suvišne atribute. Ako STUD_NO identifikuje studenta, onda su i {STUD_NO, SNAME} i {STUD_NO, PHONE} superključevi.

12.3 Alternativni ključ (Alternate Key)

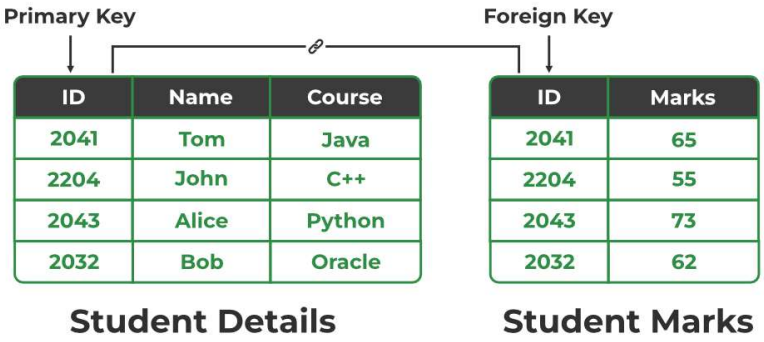
Svaki kandidatski ključ koji nije izabran za primarni. Ako su STUD_NO i PHONE kandidati, a izabran je STUD_NO, onda je PHONE alternativni ključ.



Slika 50. *Kandidatski, primarni i alternativni ključ*

12.4 Strani (spoljni) ključ (Foreign Key)

Atribut u jednoj tabeli koji upućuje na primarni ključ druge tabele. Tabela sa stranim ključem je **referencirajuća**, a tabela na koju se upućuje **referencirana**. Strani ključ povezuje tabele, čuva integritet i sprečava redundansu.



Slika 51. *Strani ključ povezuje dve tabele*

> 💡 Za razliku od primarnog ključa, strani ključ **može imati duplikate i NULL**. STUD_NO se ponavlja u STUDENT_COURSE jer student sluša više predmeta, a u tabeli STUDENT je primarni ključ, jedinstven i NOT NULL.

12.5 Delimični ključ (Partial Key)

Delimični ključ (diskriminator) slabog entiteta pomaže u identifikaciji zapisa, ali ga sam ne identifikuje jedinstveno, već tek u kombinaciji sa ključem jakog entiteta. Ne sme biti NULL i može imati jednu ili više kolona.

12.6 Primarni ključ (Primary Key)

Bira se iz skupa kandidatskih ključeva. **Ne može biti NULL**, može biti prost ili složen, a baze ga obično indeksiraju radi bržeg pristupa. Primer: STUDENT(STUD_NO, SNAME, ADDRESS, PHONE), gde je STUD_NO primarni ključ.

12.7 Sekundarni ključ (Secondary Key)

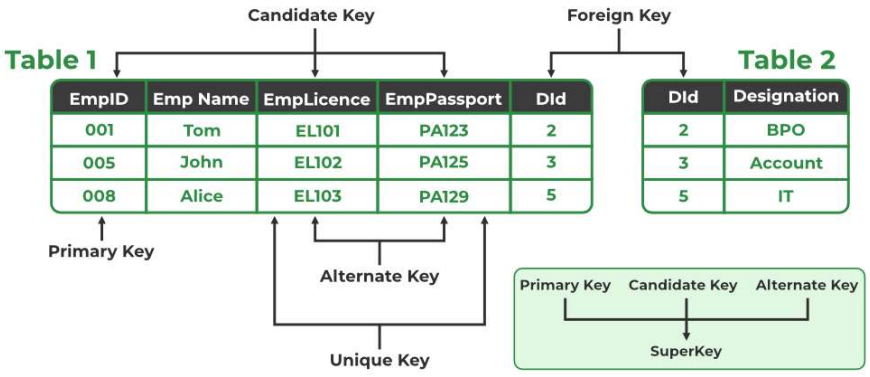
Atribut koji se koristi za pretraživanje, najčešće preko indeksa, ali **ne garantuje jedinstvenost**. Npr. STUD_NAME: više studenata se može zvati John.

12.8 Jedinstveni ključ (Unique Key)

Ograničenje koje osigurava da su vrednosti u koloni ili kombinaciji kolona jedinstvene. Za razliku od primarnog ključa, **dozvoljava NULL** (obično jednu NULL vrednost, zavisno od RDBMS-a). Npr. kombinacija STUD_EMAIL i STUD_NAME.

12.9 Složeni ključ (Composite Key)

Kada jedna kolona nije dovoljna, ključ čini kombinacija više atributa, npr. {STUD_NO, COURSE_NO} u tabeli STUDENT_COURSE.



Slika 52. *Ključevi na primeru dve povezane tabele*

12.10 Surogat (veštački) ključ (Surrogate Key)

Veštački atribut, najčešće automatski generisan (*auto-increment*), koji služi kao primarni ključ kada nema pogodnog prirodnog ključa. **Nema značenje u realnom svetu**. Npr. STUDENT_ID.

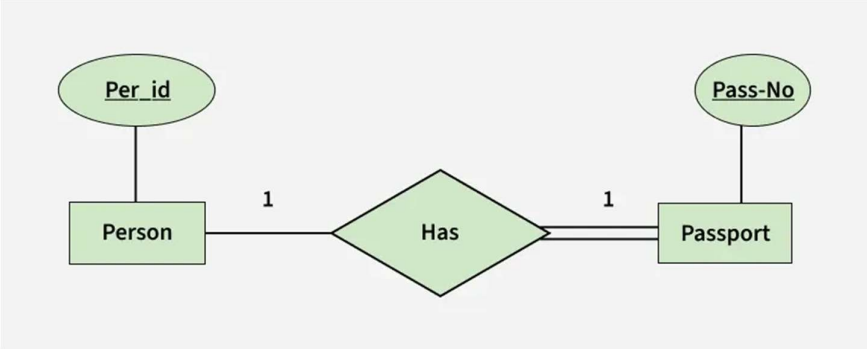
Tabela 13. *Poređenje vrsta ključeva*

Ključ	Jedinstven?	NULL?	Uloga
Primarni	da	ne	glavni identifikator reda
Kandidatski	da	ne	mogući primarni ključ, minimalan
Alternativni	da	ne	kandidat koji nije izabran
Superključ	da	ne	kandidat + eventualni suvišni atributi
Jedinstveni	da	da (obično jedan)	ograničenje jedinstvenosti
Strani	ne	da	veza ka drugoj tabeli
Sekundarni	ne	da	brže pretraživanje (indeks)

13. Prevođenje ER modela u relacioni model

ER model opisuje konceptualnu strukturu baze, a relacioni model je logička reprezentacija koja se direktno implementira u RDBMS-u (Oracle, MySQL). Postupak je uvek isti: prvo se svaki entitet i veza prevedu u posebnu tabelu, a zatim se spajaju tabele koje imaju isti primarni ključ.

13.1 Slučaj 1: veza 1:1, jedan entitet ima potpuno učešće



Slika 53. *Person – Has – Passport*

Osoba ima 0 ili 1 pasoš, a pasoš uvek pripada tačno jednoj osobi: veza je 1:1 sa potpunim učešćem pasoša. Najpre dobijamo tri tabele:

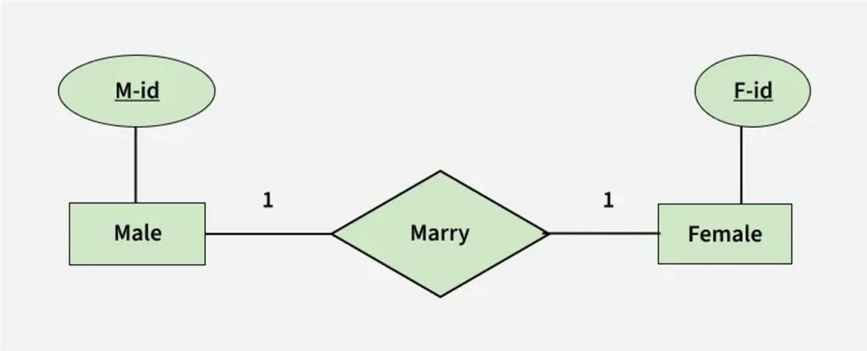
Tabela 14. *Person, Has i Passport*

Person.Per-Id	ostalo	Has.Per-Id	Has.Pass-No	Passport.Pass-No	ostalo
PR1	–	PR1	PS1	PS1	–
PR2	–	PR2	PS2	PS2	–
PR3	–	–	–	–	–

Svaki Per-Id i Pass-No pojavljuje se najviše jednom u tabeli Has, pa se sve tri mogu spojiti u **jednu tabelu**: PERSON (Per-Id, ostali atributi osobe, Pass-No, ostali atributi pasoša). Pass-No ne može biti primarni ključ, jer kod osobe bez pasoša ima vrednost NULL.

> 💡 Praktično pravilo: kod 1:1 sa potpunim učešćem jedne strane dovoljno je i dve tabele, pri čemu primarni ključ entiteta sa delimičnim učešćem ide kao strani ključ (NOT NULL, UNIQUE) u tabelu entiteta sa potpunim učešćem.

13.2 Slučaj 2: veza 1:1, oba entiteta imaju delimično učešće



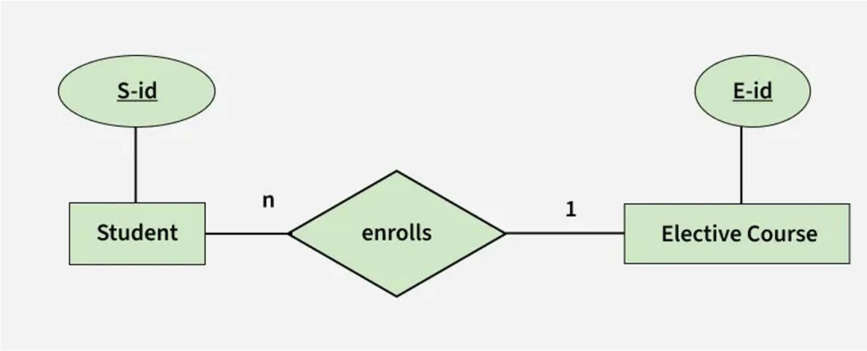
Slika 54. *Male – Marry – Female*

Muškarac je u braku sa 0 ili 1 ženom i obrnuto. Ako bismo sve spojili u jednu tabelu, za neke M-Id atribut F-Id bio bi NULL i obrnuto, pa nijedan atribut ne bi bio uvek NOT NULL. Zato se dobijaju **dve tabele**: MALE (M-Id, ostali atributi, F-Id) i FEMALE (F-Id, ostali atributi). Strani ključ se stavlja u tabelu entiteta koji češće učestvuje u vezi, da bi bilo manje NULL vrednosti.

Tabela 15. *Male, Marry i Female*

Male.M-Id	ostalo	Marry.M-Id	Marry.F-Id	Female.F-Id	ostalo
M1	–	M1	F2	F1	–
M2	–	M2	F1	F2	–
M3	–	–	–	F3	–

13.3 Slučaj 3: veza N:1 (1:N)

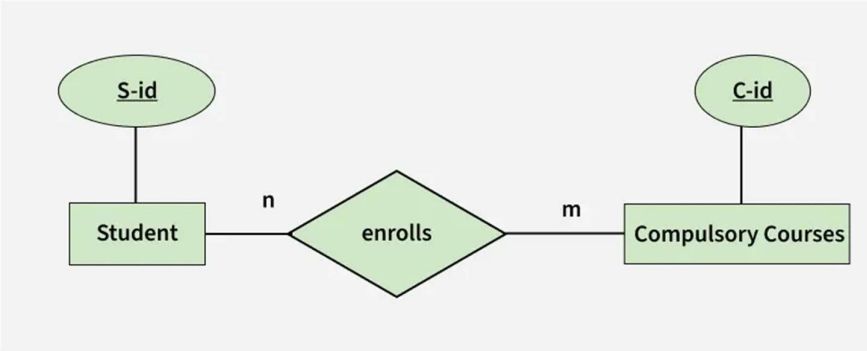


Slika 55. *Student – enrolls – Elective Course*

Student se upisuje na jedan izborni predmet, a na predmet se upisuje više studenata. S-Id se u tabeli Enrolls ne ponavlja, pa je to njen primarni ključ, isti kao kod tabele Student. Te dve tabele se spajaju, pa su rezultat **dve tabele**: STUDENT (S-Id, ostali atributi, E-Id) i ELECTIVE_COURSE (E-Id, ostali atributi).

> **PRAVILO**
> Primarni ključ entiteta sa strane **1** ide kao strani ključ u tabelu entiteta sa strane **N**.

13.4 Slučaj 4: veza M:N



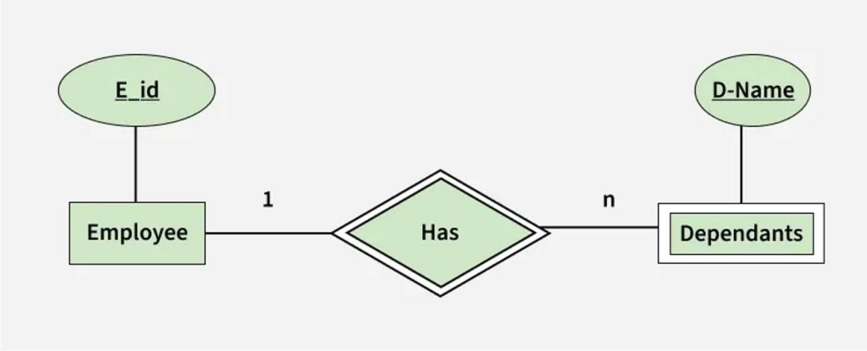
Slika 56. *Student – enrolls – Compulsory Courses*

Student sluša više obaveznih predmeta, a predmet sluša više studenata. U tabeli Enrolls i S-Id i C-Id se ponavljaju, ali je kombinacija (S-Id, C-Id) jedinstvena i postaje **složeni primarni ključ**. Pošto su sva tri primarna ključa različita, tabele se ne mogu spojiti: potrebne su **tri tabele** – STUDENT (S-Id, …), COMPULSORY_COURSES (C-Id, …) i ENROLLS (S-Id, C-Id).

Tabela 16. *Vezna tabela Enrolls: ponavljaju se pojedinačne vrednosti, ali ne i parovi*

Enrolls.S-Id	Enrolls.C-Id	Student.S-Id	Course.C-Id
S1	C1	S1	C1
S1	C2	S2	C2
S3	C1	S3	C3
S4	C3	S4	C4
S4	C2	–	–
S3	C3	–	–

13.5 Slučaj 5: veza sa slabim entitetom



Slika 57. *Employee – Has – Dependants*

Zaposleni ima više izdržavanih lica, a izdržavano lice zavisi od tačno jednog zaposlenog i ne postoji bez njega. Ključ slabog entiteta je kombinacija ključa identifikacionog entiteta (E-Id) i delimičnog

ključa (D-Name). Tabele Has i Dependents imaju isti ključ (E-Id, D-Name), pa se spajaju. Rezultat su **dve tabele**: EMPLOYEE (E-Id, …) i DEPENDENTS (E-Id, D-Name, …), gde je E-Id istovremeno i strani ključ.

13.6 Pregled pravila prevođenja

Tabela sažima slučajeve 1–5 i dopunjuje ih pravilima za elemente iz ER poglavlja (složeni, viševrednosni i izvedeni atributi, ternarne veze, specijalizacija i agregacija), koja su potrebna za zadatke u poglavlju 15.

Tabela 17. *Pravila prevođenja ER modela u relacioni model*

Element ER modela	Kako se prevodi
Jaki entitet	Tabela sa svim prostim atributima; ključ entiteta je primarni ključ.
Složeni atribut	U tabelu idu samo njegovi delovi (Ulica, Grad…), ne i sam složeni atribut.
Izvedeni atribut	Ne čuva se; izračunava se upitom (npr. starost iz datuma rođenja).
Viševrednosni atribut	Posebna tabela (ključ entiteta, vrednost), oba dela čine primarni ključ.
1:1	Strani ključ u tabelu entiteta sa potpunim učešćem (ili spajanje u jednu tabelu, slučaj 1); kod delimičnog oba – dve tabele, slučaj 2.
1:N	Strani ključ strane 1 u tabelu strane N; atributi veze idu uz njega. Potpuno učešće strane N → NOT NULL.
M:N	Vezna tabela sa ključevima oba entiteta kao složenim primarnim ključem i atributima veze.
Rekurzivna veza	Kao binarna, ali oba strana ključa upućuju na istu tabelu (npr. ŠifraMentora → LEKAR).
Ternarna veza	Tabela sa stranim ključevima sva tri entiteta i atributima veze.
Slabi entitet	Tabela sa ključem jakog entiteta (strani ključ) + delimičnim ključem kao složenim primarnim ključem.
Specijalizacija (IS-A)	Tabela za nadtip i po jedna za svaki podtip; primarni ključ podtipa je istovremeno strani ključ ka nadtipu.
Agregacija	Agregirana veza dobija svoju tabelu; njen složeni ključ je strani ključ u tabeli veze višeg nivoa.

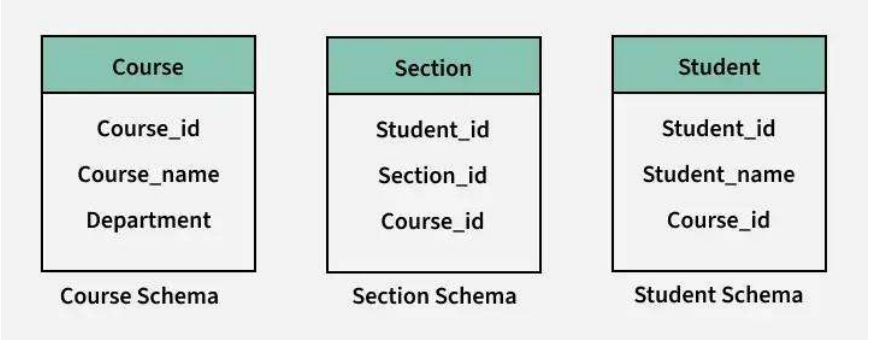
14. Strategije projektovanja šeme

Projektovanje šeme je kreiranje logičke strukture baze: tabela, kolona, veza, ograničenja i drugih elemenata koji određuju kako se podaci čuvaju i kako im se pristupa.

14.1 Četiri strategije

Tabela 18. *Strategije projektovanja šeme*

Strategija	Kako se radi
Odozgo nadole (Top-Down)	Počinje se od nekoliko entiteta visokog nivoa (npr. ER dijagram sa Zaposleni, Odeljenje, Projekat), pa se oni postepeno razlažu na atribute, podtipove i tabele. Primer je specijalizacija.
Odozdo nagore (Bottom-Up)	Počinje se od atributa i njihovih funkcionalnih zavisnosti, koji se grupišu u tabele normalizacijom (1NF, 2NF, 3NF, BCNF). Pogodno kada već postoje obrasci i dokumenti. Primer je generalizacija.
Iznutra ka spolja (Inside-Out)	Poseban slučaj odozdo nagore: prvo se projektuju centralni, najočigledniji koncepti, a zatim se šema širi dodavanjem povezanih entiteta.
Mešovita (Mixed)	Zahtevi se dele odozgo nadole, svaki deo se projektuje odozdo nagore, a dobijene šeme se zatim spajaju.



Slika 58. *Šeme tabela Course, Section i Student*

14.2 Koraci za kreiranje baze podataka

1. **Odredite svrhu i opseg baze:** koje podatke čuva, kako se koristi i kakvi će upiti biti.
2. **Normalizujte bazu:** smanjite redundansu i eliminišite anomalije.
3. **Izaberite odgovarajuće tipove podataka:** numerički za brojeve, datumski za datume.
4. **Uspostavite veze između tabela:** strani ključevi čuvaju konzistentnost.
5. **Koristite ograničenja:** primarni ključ, CHECK i druga pravila nad podacima.
6. **Optimizujte performanse:** indeksi, particionisanje, pravi tipovi podataka.

14.3 Pristupi projektovanju šeme

Tabela 19. *Poređenje pristupa*

Pristup	Dobija se	Cena
Normalizacija	Manje redundanse, nema anomalija ažuriranja, umetanja i brisanja.	Više tabela i više JOIN operacija u upitima.
Denormalizacija	Brži i jednostavniji upiti, manje spajanja.	Redundansa, više prostora, rizik od nekonzistentnosti.
Vertikalno particionisanje	Tabela se deli po kolonama; čitaju se samo potrebne kolone.	Sporije kada upit traži kolone iz više delova.
Horizontalno particionisanje	Tabela se deli po redovima; manje podataka za skeniranje, bolja skalabilnost.	Sporije kada upit traži podatke iz više particija.

14.4 Ograničenja u SUBP-u

Tabela 20. *Vrste ograničenja*

Ograničenje	Osigurava	Primer
PRIMARY KEY	svaki zapis je jedinstven	StudentID u tabeli Students
FOREIGN KEY	vrednost postoji u drugoj tabeli	OrderID u OrderDetails postoji u Orders
UNIQUE	sve vrednosti u koloni su različite	Email u tabeli Users
NOT NULL	kolona ne sme biti prazna	LastName u tabeli Employees
CHECK	vrednost ispunjava uslov	Age > 18 u tabeli Persons
DEFAULT	podrazumevana vrednost kada nije zadata	Status = 'Pending' u tabeli Orders

> **ZAKLJUČAK**
> Dobra šema zahteva razumevanje podataka i načina na koji će se koristiti, kao i poznavanje najboljih praksi projektovanja i optimizacije. Prateći ove strategije dobija se robusna i efikasna šema koja podržava poslovne ciljeve.

15. Zadaci: prevođenje ER modela u relacioni model

U svakom zadatku prevedite ER dijagram iz odgovarajućeg rešenja u poglavlju 8 u relacioni model. Navedite sve tabele, podvucite primarne ključeve, označite strane ključeve i za svaki strani ključ navedite da li sme biti NULL. Koristite pravila iz tabele u odeljku 13.6.

> 💡 Oznake u rešenjima: podvučeno je primarni ključ, *kurziv* je strani ključ; strelica pokazuje na koju tabelu strani ključ upućuje.

15.1 Zadatak 1 – Biblioteka u relacionom modelu

Prevedite ER dijagram gradske biblioteke (rešenje zadatka 8.1) u relacioni model. Obratite pažnju na složeni atribut ImePrezime, viševrednosni atribut Telefon i izvedeni atribut Starost.

Rešenje

Tabela 21. *Relaciona šema biblioteke – 7 tabela*

Tabela	Šema	Iz čega nastaje
IZDAVAČ	IZDAVAČ(PIB, Naziv)	jaki entitet
KNJIGA	KNJIGA(ISBN, Naslov, GodinaIzdanja, *PIB* → IZDAVAČ, NOT NULL)	entitet + veza IZDAJE (N:1)
AUTOR	AUTOR(ŠifraAutora, ImeAutora)	jaki entitet
NAPISAO	NAPISAO(*ISBN*, *ŠifraAutora*)	veza M:N
ČLAN	ČLAN(ČlanskiBroj, Ime, Prezime, DatumRođenja)	entitet; složeni atribut razložen
TELEFON_ČLANA	TELEFON_ČLANA(*ČlanskiBroj*, Telefon)	viševrednosni atribut
POZAJMICA	POZAJMICA(*ČlanskiBroj*, *ISBN*, DatumPozajmice, RokVraćanja)	veza M:N sa atributima

- **IZDAJE (N:1)** ne dobija svoju tabelu: PIB izdavača ide u KNJIGA. Pošto knjiga ima potpuno učešće, PIB je NOT NULL.
- **Starost se ne čuva**, jer se izračunava iz DatumRođenja; ImePrezime se razlaže na Ime i Prezime.
- **POZAJMICA:** po osnovnom pravilu ključ je (ČlanskiBroj, ISBN). Ako isti član može više puta pozajmiti istu knjigu, u ključ se dodaje i DatumPozajmice – dobra tema za diskusiju.

15.2 Zadatak 2 – Bolnica u relacionom modelu

Prevedite ER dijagram bolnice (rešenje zadatka 8.2). Posebnu pažnju obratite na slabi entitet SOBA, vezu 1:1 RUKOVODI i rekurzivnu vezu NADZIRE.

Rešenje

Tabela 22. *Relaciona šema bolnice – 6 tabela*

Tabela	Šema	Iz čega nastaje
ODELJENJE	ODELJENJE(ŠifraOdeljenja, Naziv, Sprat, *ŠifraŠefa* → LEKAR, NOT NULL, UNIQUE)	entitet + RUKOVODI (1:1)
LEKAR	LEKAR(ŠifraLekara, ImePrezime, Specijalizacija, *ŠifraOdeljenja* → ODELJENJE, NOT NULL, *ŠifraMentora* → LEKAR, NULL)	entitet + RADI_U (N:1) + NADZIRE (rekurzivna 1:N)
SOBA	SOBA(*ŠifraOdeljenja*, BrojSobe, BrojKreveta)	slabi entitet + IMA
PACIJENT	PACIJENT(JMBG, ImePrezime, (*ŠifraOdeljenja*, *BrojSobe*) → SOBA, NULL)	entitet + SMEŠTEN_U (N:1)
ALERGIJA	ALERGIJA(*JMBG*, Alergija)	viševrednosni atribut
LEČENJE	LEČENJE(*ŠifraLekara*, *JMBG*, Dijagnoza, DatumPregleda)	veza M:N sa atributima

- **RUKOVODI (1:1):** odeljenje ima potpuno učešće, pa strani ključ ŠifraŠefa ide u ODELJENJE, NOT NULL i UNIQUE (jedan lekar vodi najviše jedno odeljenje) – slučaj 1.
- **NADZIRE:** rekurzivna 1:N postaje strani ključ ŠifraMentora u istoj tabeli LEKAR; NULL je dozvoljen, jer nemaju svi mentora.
- **SOBA:** ključ je (ŠifraOdeljenja, BrojSobe) – slučaj 5. Zato i strani ključ u PACIJENT ima dve kolone, a NULL je dozvoljen jer smeštaj nije obavezan.
- **Kružne reference** ODELJENJE ↔ LEKAR zahtevaju da se jedan strani ključ doda naknadno (ALTER TABLE) ili da se proverava na kraju transakcije.

15.3 Zadatak 3 – Fakultet u relacionom modelu

Prevedite ER dijagram fakulteta (rešenje zadatka 8.3). Iskoristite (min, max) oznake da odredite koji strani ključevi su NOT NULL i kako se prevode ternarna veza POLAŽE i rekurzivna veza PREDUSLOV.

Rešenje

Tabela 23. *Relaciona šema fakulteta – 7 tabela*

Tabela	Šema	Iz čega nastaje
SMER	SMER(ŠifraSmera, NazivSmera)	jaki entitet
STUDENT	STUDENT(BrojIndeksa, ImePrezime, Email, *ŠifraSmera* → SMER, NOT NULL)	entitet + UPISAN_NA
PREDMET	PREDMET(ŠifraPredmeta, Naziv, ESPB)	jaki entitet
PROFESOR	PROFESOR(ŠifraProfesora, ImePrezime, Zvanje)	jaki entitet
PREDAJE	PREDAJE(*ŠifraPredmeta*, *ŠifraProfesora*)	veza M:N

Tabela	Šema	Iz čega nastaje
POLAŽE	POLAŽE(*BrojIndeksa, ŠifraPredmeta, ŠifraProfesora*, Datum, Ocena)	ternarna veza
PREDUSLOV	PREDUSLOV(*ŠifraPredmeta, ŠifraPreduslova*) – oba → PREDMET	rekurzivna M:N

- **STUDENT (1,1) u UPISAN_NA:** maksimum 1 znači da veza ide kao strani ključ u STUDENT, a minimum 1 da je NOT NULL.
- **POLAŽE:** ternarna veza postaje tabela sa tri strana ključa. Ako student može više puta izlaziti na isti ispit, i Datum ulazi u primarni ključ.
- **PREDUSLOV:** dva strana ključa ka istoj tabeli PREDMET, sa različitim imenima kolona prema ulogama.
- **Ograničenje PREDMET (1,N) u PREDAJE** ne može se izraziti samim stranim ključem; obezbeđuje se aplikacijom ili okidačem (trigerom).

15.4 Zadatak 4 – Aviokompanija u relacionom modelu

Prevedite ER dijagram aviokompanije (rešenje zadatka 8.4). Primenite pravilo za specijalizaciju: tabela za nadtip i po jedna za svaki podtip.

Rešenje

Tabela 24. *Relaciona šema aviokompanije – 11 tabela*

Tabela	Šema	Iz čega nastaje
OSOBA	OSOBA(JMBG, ImePrezime, DatumRođenja)	nadtip
PUTNIK	PUTNIK(*JMBG* → OSOBA, BrojPasoša, Državljanstvo)	podtip
ZAPOSLENI	ZAPOSLENI(*JMBG* → OSOBA, Plata, DatumZaposlenja)	podtip i nadtip
PILOT	PILOT(*JMBG* → ZAPOSLENI, BrojLicence, SatiLetenja)	podtip
MEHANIČAR	MEHANIČAR(*JMBG* → ZAPOSLENI)	podtip
SPECIJALNOST	SPECIJALNOST(*JMBG*, Specijalnost)	viševrednosni atribut
AVION	AVION(RegBroj, Model)	jaki entitet
LET	LET(BrojLeta, DatumLeta, Relacija, *RegBroj* → AVION, NOT NULL)	entitet + KORISTI (N:1)
UPRAVLJA	UPRAVLJA(*JMBG, BrojLeta*), JMBG → PILOT	veza M:N
REZERVACIJA	REZERVACIJA(*JMBG, BrojLeta*, Sedište), JMBG → PUTNIK	veza M:N sa atributom

Tabela	Šema	Iz čega nastaje
ODRŽAVA	ODRŽAVA(*JMBG, RegBroj*, DatumServisa), JMBG → MEHANIČAR	veza M:N sa atributom

- **Podtipovi nemaju svoj ključ:** primarni ključ PILOT-a je JMBG, koji je istovremeno strani ključ ka ZAPOSLENI. Tako se nasleđuju svi atributi nadtipa, bez ponavljanja.
- **GodineStaža** je izvedeni atribut i ne čuva se.
- **MEHANIČAR** ima samo ključ, ali je tabela potrebna jer na nju upućuju ODRŽAVA i SPECIJALNOST.
- **Disjunktnost (d)** i **potpunost** specijalizacije ne mogu se izraziti samim ključevima; obezbeđuju se proverama (CHECK, trigeri) ili kolonom TipZaposlenog u tabeli ZAPOSLENI.

15.5 Zadatak 5 – Konsultantska firma u relacionom modelu

Prevedite ER dijagram konsultantske firme (rešenje zadatka 8.5). Pokažite kako se prevodi agregacija ANGAŽMAN i veze KORISTI i NADGLEDA koje je koriste.

Rešenje

Tabela 25. *Relaciona šema konsultantske firme – 8 tabela*

Tabela	Šema	Iz čega nastaje
KONSULTANT	KONSULTANT(ŠifraKonsultanta, ImePrezime)	jaki entitet
EKSPERTIZA	EKSPERTIZA(*ŠifraKonsultanta*, Ekspertiza)	viševrednosni atribut
KLIJENT	KLIJENT(PIB, Naziv, Ulica, Grad, PoštanskiBroj)	entitet; složena Adresa razložena
PROJEKAT	PROJEKAT(ŠifraProjekta, NazivProjekta, Budžet, *PIB* → KLIJENT, NOT NULL)	entitet + NARUČUJE (N:1)
MENADŽER	MENADŽER(ŠifraMenadžera, ImePrezime)	jaki entitet
ANGAŽMAN	ANGAŽMAN(*ŠifraKonsultanta, ŠifraProjekta*, BrojSati, *ŠifraMenadžera* → MENADŽER, NOT NULL)	agregirana veza RADI_NA + NADGLEDA (N:1)
OPREMA	OPREMA(InvBroj, Naziv)	jaki entitet
KORISTI	KORISTI(*ŠifraKonsultanta, ŠifraProjekta, InvBroj*, Količina), (ŠifraKonsultanta, ŠifraProjekta) → ANGAŽMAN	veza M:N višeg nivoa

- **Agregacija:** veza RADI_NA dobija tabelu ANGAŽMAN sa složenim ključem (ŠifraKonsultanta, ŠifraProjekta) i atributom BrojSati.
- **NADGLEDA (N:1)** se ne prevodi u posebnu tabelu: ključ menadžera ide kao strani ključ u ANGAŽMAN, NOT NULL jer svaki angažman ima menadžera.

- **KORISTI** upućuje na ANGAŽMAN, a ne posebno na KONSULTANT i PROJEKAT: strani ključ je par kolona, što garantuje da se oprema vezuje samo za postojeći angažman.
- **Klijent:** složena Adresa razložena je na Ulica, Grad i PoštanskiBroj.

DEO IV

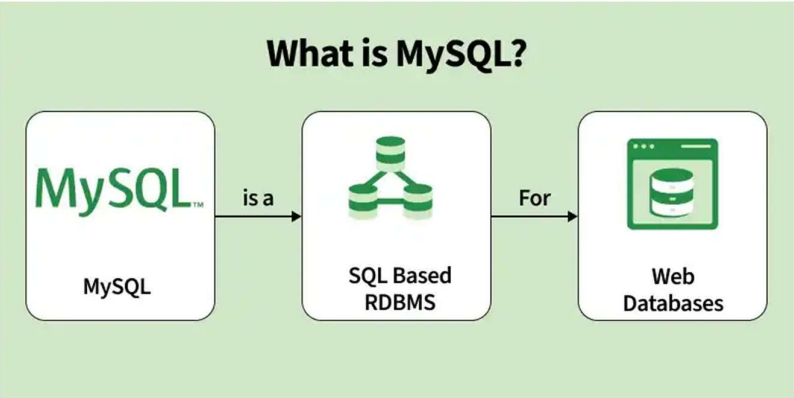
MySQL u praksi

16. Uvod u MySQL

16.1 Šta je MySQL?

DEFINICIJA

MySQL je relacioni SUBP (RDBMS) otvorenog koda koji za skladištenje, upravljanje i manipulaciju podacima koristi jezik **SQL**. Razvija ga i održava kompanija **Oracle Corporation**, a zbog brzine, pouzdanosti i jednostavnosti jedan je od najkorišćenijih sistema u veb-aplikacijama.



Slika 59. *MySQL je SQL-zasnovan RDBMS za veb-baze podataka*

Radi na Windows-u, Linux-u i macOS-u, najčešće se koristi uz PHP, Java i Python, a poznat je po skalabilnosti, bezbednosnim mehanizmima i velikoj zajednici. Osnovni pojmovi koje koristi – tabela, red, kolona, primarni i strani ključ – opisani su u poglavljima 9 i 12.

16.2 MySQL i SQL nisu isto

Tabela 26. *MySQL naspram SQL-a*

Kriterijum	MySQL	SQL
Šta je	softver – sistem za upravljanje relacionim bazama (RDBMS)	standardni jezik za rad sa relacionim bazama
Funkcija	skladišti, organizuje i štiti podatke na disku	služi za pisanje upita (SELECT, INSERT, UPDATE…)
Gde se koristi	jedan konkretan proizvod	u svim RDBMS-ovima: MySQL, PostgreSQL, Oracle, SQL Server

💡 Ista greška kao „MySQL je baza”: MySQL je SUBP koji razume jezik SQL, a baza podataka je skup podataka kojim on upravlja.

16.3 Kako MySQL obrađuje upit

1. **Zahtev klijenta:** korisnik šalje SQL upit preko aplikacije, API-ja ili komandne linije.
2. **Konekcija:** server uspostavlja vezu i otvara sesiju za klijenta.
3. **Parsiranje:** server proverava sintaksu upita.
4. **Optimizacija:** optimizator poredi više načina izvršavanja i bira najefikasniji.
5. **Izvršavanje:** upit se izvršava preko mehanizma skladištenja (*storage engine*), npr. **InnoDB** ili **MyISAM**, koji čita i upisuje podatke na disk.
6. **Odgovor:** rezultat ili potvrda vraća se klijentu.

16.4 Ključne mogućnosti

Tabela 27. *Mogućnosti MySQL-a*

Mogućnost	Šta znači
Otvoren kod	besplatan za korišćenje, kod se može menjati
Performanse	brza obrada i čitanje velikih skupova podataka
Transakcije i ACID	pouzdana obrada povezanih operacija, naročito uz InnoDB
Dnevnik i oporavak	beleženje promena omogućava oporavak posle pada sistema
Replikacija i bekap	kopije podataka na više servera radi visoke dostupnosti
Skalabilnost	particionisanje i klasterovanje za velika opterećenja
Više mehanizama skladištenja	izbor između InnoDB, MyISAM, Memory i drugih
Bezbednost	SSL enkripcija, autentifikacija i detaljne privilegije

16.5 Gde se koristi

Tabela 28. *Primena MySQL-a*

Oblast	Primena
E-trgovina	katalozi proizvoda, korpe, nalozi kupaca, plaćanja
CMS sistemi	sadržaj, podešavanja i komentari (WordPress, Joomla, Drupal)
Mala i srednja preduzeća	klijenti, fakture, poslovanje
Velike platforme i društvene mreže	ogroman broj interakcija, objava i profila
Obrazovanje i razvoj	izrada veb-aplikacija i obuka iz baza podataka

U oblaku se MySQL nudi kao upravljana usluga (AWS RDS, Google Cloud SQL, Azure Database for MySQL): provajder preuzima održavanje, bekap i bezbednosne zakrpe, a resursi se automatski proširuju sa opterećenjem.

17. Instalacija MySQL-a na Windows

Instalacijom MySQL-a na sopstveni računar dobija se lokalni server na kome se mogu raditi sve vežbe iz ovog praktikuma: kreiranje baza i tabela, upiti, korisnici i privilegije.

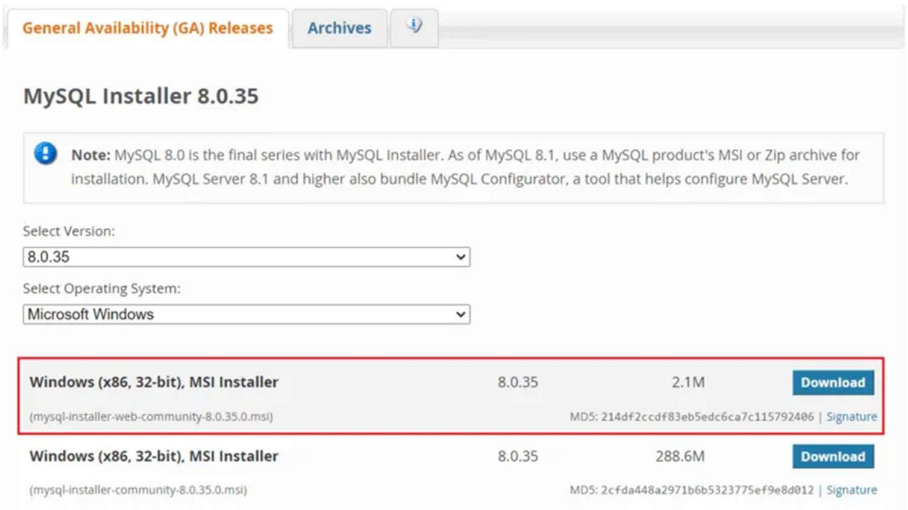
17.1 Sistemski zahtevi

Tabela 29. *Sistemski zahtevi*

Kategorija	Zahtev
Operativni sistem	Windows 11, 10, 8, 7 ili Windows Server 2016/2019+
Hardver	najmanje 4 GB RAM-a i 2 GB slobodnog prostora na disku
Softver	ažuriran Windows; preporučuje se uklanjanje starijih verzija MySQL-a
Mreža	internet konekcija za preuzimanje instalera

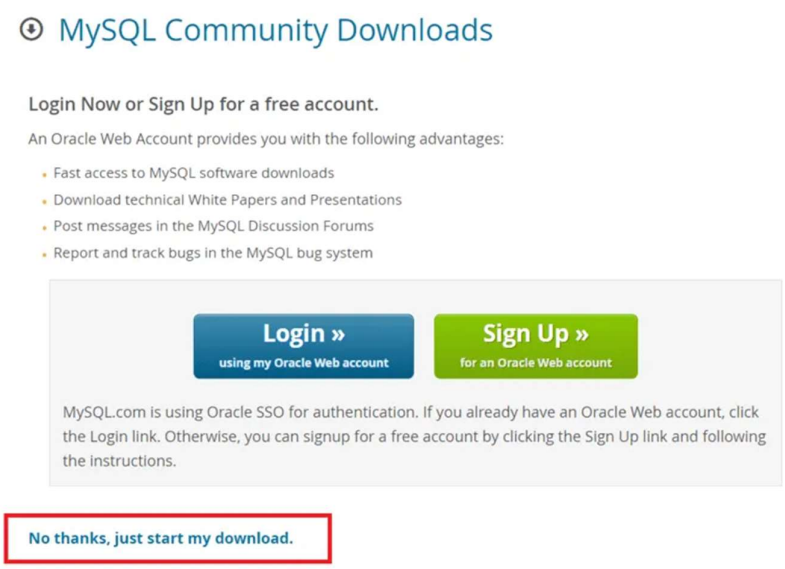
17.2 Preuzimanje instalera

1. Na sajtu **mysql.com** otvorite **Downloads → MySQL Community (GPL) Downloads → MySQL Installer for Windows**.
2. Izaberite instaler: **Web Installer** (oko 2 MB, komponente preuzima tokom instalacije) ili **kompletan instaler** sa svim komponentama, koji se preporučuje.
3. Registracija nije potrebna: na stranici za prijavu kliknite **„No thanks, just start my download."** i preuzmite .msi fajl.



Slika 60. *Izbor instalera na stranici za preuzimanje*

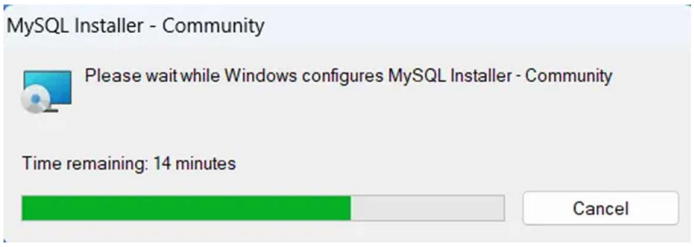
> 💡 Kao što piše na stranici za preuzimanje, MySQL Installer postoji za seriju 8.0. Novije verzije (8.1 i dalje) instaliraju se MSI paketom samog servera, a Workbench i ostali alati preuzimaju se posebno.



Slika 61. *Preuzimanje bez prijave: „No thanks, just start my download."*

17.3 Instalacija komponenti

Pokrenite preuzeti fajl (npr. mysql-installer-community-8.x.x.msi) iz foldera **Downloads**. Ako Windows (UAC) pita da li dozvoljavate izmene na uređaju, kliknite **Yes**.



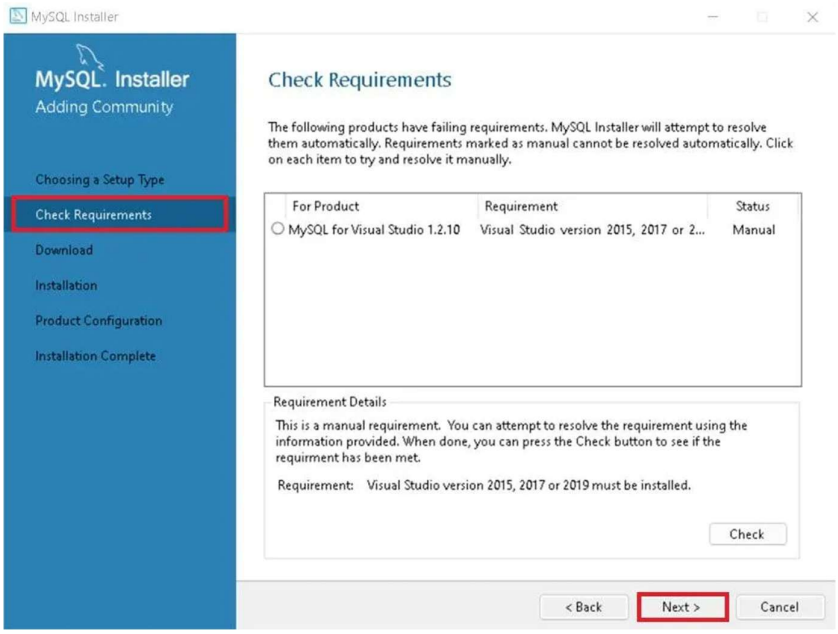
Slika 62. *Pokretanje MySQL Installer-a*

Tip instalacije (Choose Setup Type):

Tabela 30. *Tipovi instalacije*

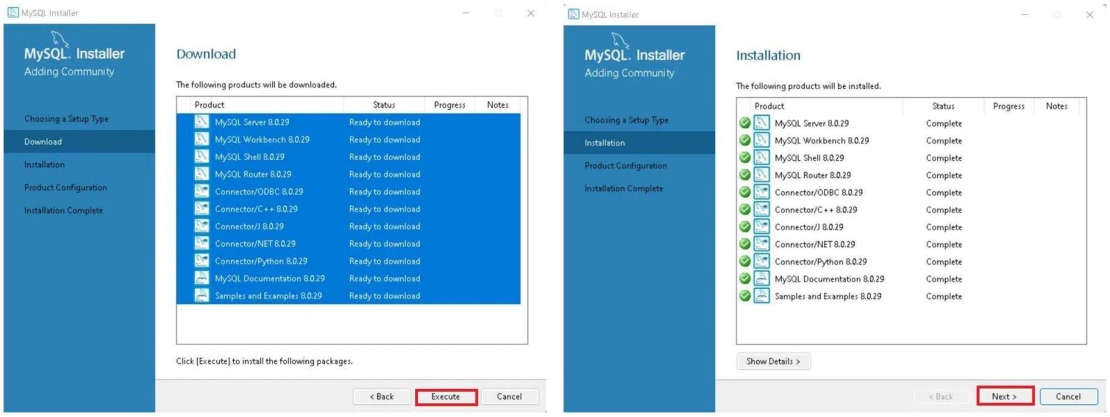
Opcija	Šta instalira
Developer Default	Server, Workbench (grafički alat), Shell, dokumentaciju i konektore (Python, Java, C++…) – preporuka za vežbe
Server only	samo MySQL Server, bez alata
Client only	samo klijentske alate za rad sa udaljenim serverom
Full	sve komponente iz paketa
Custom	ručni izbor komponenti i verzija

Provera preduslova (Check Requirements): instaler proverava da li postoje potrebne sistemske komponente, npr. *Microsoft Visual C++ Redistributable*. Većinu rešava sam klikom na **Execute**; ono što ne može, preuzmete ručno sa Microsoft sajta i kliknete **Check**. Zatim **Next**.



Slika 63. *Provera preduslova*

Preuzimanje i instalacija: kliknite **Execute** i sačekajte da se pored svake komponente pojavi zelena kvačica, pa **Next**.



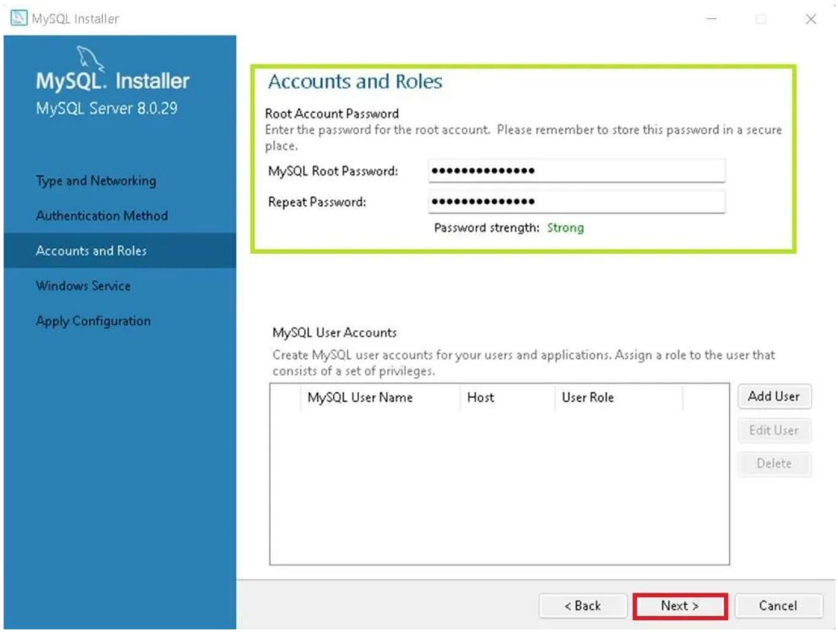
Slika 64. *Preuzimanje (levo) i instalacija (desno) komponenti*

17.4 Konfiguracija servera

Tabela 31. *Koraci konfiguracije*

Korak	Šta podesiti
Type and Networking	Config Type: *Development Computer* (štedi memoriju); TCP/IP na portu **3306**; označeno *Open Windows Firewall port for network access*
Authentication Method	*Use Strong Password Encryption for Authentication* (preporučeno)
Accounts and Roles	unesite i potvrdite **root lozinku**; po želji dodajte korisnike dugmetom *Add User*
Windows Service	servis **MySQL80**, označeno *Start the MySQL Server at System Startup*

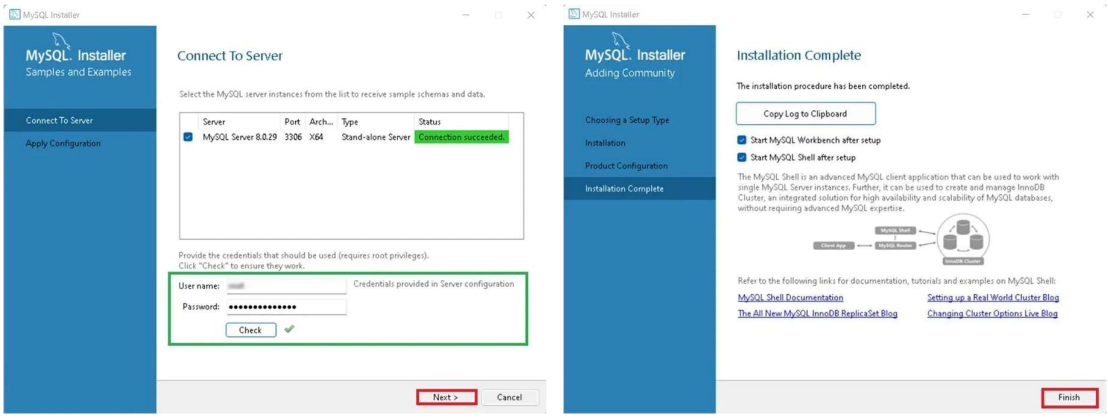
Korak	Šta podesiti
Apply Configuration	**Execute**, pa **Finish** kada sve stavke budu zelene



Slika 65. *Root lozinka i korisnički nalozi (Accounts and Roles)*

> 💡 Root lozinku obavezno zapišite: potrebna je za Workbench, komandnu liniju i povezivanje iz programa.

Provera konekcije (Connect To Server): unesite root lozinku i kliknite **Check**. Poruka *Connection succeeded* potvrđuje da server radi. Zatim **Next** i **Finish** zatvaraju instaler.



Slika 66. *Provera konekcije (levo) i završetak instalacije (desno)*

17.5 Dodavanje MySQL-a u PATH (opciono)

Da bi se komanda mysql mogla pokretati iz bilo kog terminala: **Start → „Environment Variables" → Edit the system environment variables → Environment Variables… → System variables → Path → Edit… → New**, dodajte putanju do bin foldera i potvrdite sa **OK**:

```
C:\Program Files\MySQL\MySQL Server 8.0\bin
```

Prijava iz Command Prompt-a ili PowerShell-a:

```
mysql -u root -p
```

17.6 Provera instalacije

- **MySQL Command Line Client:** pokrenite ga iz Start menija, unesite root lozinku; odzivnik mysql> znači da ste povezani.
- **MySQL Workbench:** na početnom ekranu kliknite **Local instance MySQL80** (ili napravite konekciju na 127.0.0.1:3306) i unesite root lozinku; otvara se SQL editor.

```
SHOW DATABASES;
```

18. DDL naredbe i tipovi podataka

Kada je server instaliran (poglavlje 17), prvi korak u radu je definisanje strukture: pravljenje baze, tabela, izbor tipova podataka i ograničenja. Za to služe DDL naredbe, nabrojane u odeljku 1.6. Ovo poglavlje prati redosled MySQL tutorijala na sajtu w3schools.com (www.w3schools.com/mysql), pa studenti mogu svaki primer da probaju i u njegovom onlajn editoru.

18.1 Kako se čitaju šabloni

Za svaku naredbu najpre je dat **šablon** (opšti oblik), a zatim **primer**. U šablonima važi:

Tabela 32. *Pravila zapisa šablona*

Zapis	Značenje
CREATE TABLE	velikim slovima: ključne reči, pišu se tačno tako
ime_tabele	malim slovima: zamenjuje se sopstvenim imenom ili vrednošću
[NOT NULL]	uglaste zagrade: deo koji nije obavezan
kolona1, kolona2, ...	lista koja se može nastaviti
;	svaka naredba se završava tačkom-zarezom

> 💡 MySQL ne razlikuje velika i mala slova u ključnim rečima (create table = CREATE TABLE). Velika slova su konvencija koja čini upite čitljivijim.

18.2 Rad sa bazama

```
CREATE DATABASE ime_baze;          -- nova baza
SHOW DATABASES;                    -- spisak baza na serveru
USE ime_baze;                      -- izbor baze u kojoj se radi
DROP DATABASE ime_baze;            -- brisanje baze sa svim tabelama
```

Primer:

```
CREATE DATABASE testDB;
USE testDB;
```

> 💡 DROP DATABASE nepovratno briše bazu i sve podatke u njoj. Za brisanje je potrebno odgovarajuće pravo (poglavlje 19). Pre ovakvih naredbi treba napraviti bekap.

18.3 Kreiranje tabele (CREATE TABLE)

```
CREATE TABLE ime_tabele (
  kolona1 tip_podatka [ograničenja],
  kolona2 tip_podatka [ograničenja],
  ...
);
```

Svaka kolona ima ime i **tip podatka**, koji određuje kakve vrednosti sme da sadrži (odeljak 18.4). Ograničenja su opisana u odeljku 18.5.

Primer: tabela Persons sa pet kolona.

```
CREATE TABLE Persons (
  PersonID  INT,
  LastName  VARCHAR(255),
  FirstName VARCHAR(255),
  Address   VARCHAR(255),
  City      VARCHAR(255)
);
```

Tabela se može napraviti i kao kopija dela postojeće tabele; nova tabela preuzima izabrane kolone, njihove tipove i podatke:

```
CREATE TABLE nova_tabela AS
  SELECT kolona1, kolona2, ... FROM postojeca_tabela [WHERE uslov];

-- primer
CREATE TABLE PersonsBeograd AS
  SELECT PersonID, LastName, FirstName FROM Persons WHERE City = 'Beograd';
```

Struktura postojeće tabele proverava se naredbom DESCRIBE ime_tabele;.

18.4 Tipovi podataka

MySQL ima tri glavne grupe tipova: tekstualne, numeričke i tipove za datum i vreme. Izbor tipa utiče na to koje vrednosti kolona prima, koliko prostora zauzima i kako se vrednosti porede i sortiraju.

Tekstualni tipovi

Tabela 33. *Tekstualni tipovi*

Tip	Opis
CHAR(n)	tekst fiksne dužine, n od 0 do 255; kraće vrednosti se dopunjuju razmacima
VARCHAR(n)	tekst promenljive dužine do n znakova (n do 65 535) – najčešći izbor
TEXT	duži tekst, do 65 535 bajtova (postoje i MEDIUMTEXT i LONGTEXT)
ENUM('a', 'b', ...)	tačno jedna vrednost iz zadate liste
SET('a', 'b', ...)	nula ili više vrednosti iz zadate liste
BLOB	binarni podaci (slike, fajlovi)

Numerički tipovi

Tabela 34. *Numerički tipovi*

Tip	Opis
TINYINT, SMALLINT, MEDIUMINT	celi brojevi manjeg opsega (TINYINT: −128 do 127)
INT	ceo broj, oko ±2,1 milijardu – najčešći izbor za ključeve
BIGINT	veliki ceo broj (npr. broj telefona bez razmaka)

Tip	Opis
BOOL / BOOLEAN	isto što i TINYINT(1): 0 je netačno, sve ostalo tačno
DECIMAL(p, d)	tačan decimalni broj: p cifara ukupno, d posle zareza – za novac
FLOAT, DOUBLE	približan decimalni broj – za merenja, ne za novac

> 💡 Uz brojeve se može dodati UNSIGNED, čime kolona prima samo nenegativne vrednosti, a gornja granica se udvostručuje.

Datum i vreme

Tabela 35. *Tipovi za datum i vreme*

Tip	Format	Opseg
DATE	'YYYY-MM-DD'	'1000-01-01' do '9999-12-31'
DATETIME	'YYYY-MM-DD hh:mm:ss'	'1000-01-01 00:00:00' do '9999-12-31 23:59:59'
TIMESTAMP	'YYYY-MM-DD hh:mm:ss'	1970-01-01 do 2038-01-19 (UTC); čuva se u UTC-u
TIME	'hh:mm:ss'	'-838:59:59' do '838:59:59'
YEAR	YYYY	1901 do 2155

Tip **DATE** čuva samo datum, bez vremena. Unosi se kao tekst u formatu godina-mesec-dan ('2004-12-22') ili kao broj (20041222). Neispravan datum, npr. '2004-02-30', u strogom SQL režimu (podrazumevan u MySQL 8) izaziva grešku; samo u nestrogom režimu upisuje se '0000-00-00'.

Primer: tabela sa datumom rođenja.

```
CREATE TABLE StudentDetails (
  Student_Id      INT AUTO_INCREMENT,
  First_name      VARCHAR(100) NOT NULL,
  Last_name       VARCHAR(100) NOT NULL,
  Date_Of_Birth   DATE NOT NULL,
  Class           VARCHAR(10)  NOT NULL,
  Contact_Details BIGINT NOT NULL,
  PRIMARY KEY (Student_Id)
);

INSERT INTO StudentDetails (First_name, Last_name, Date_Of_Birth, Class,
Contact_Details)
VALUES ('Amit',  'Jana',     '2004-12-22', 'XI',   1234567890),
       ('Manik', 'Aggarwal', '2006-07-04', 'IX',   1245678998);
```

Funkcija **CURRENT_DATE()** (ili **CURDATE()**) vraća današnji datum servera, a **NOW()** trenutni datum i vreme:

```
INSERT INTO Orders (OrderID, OrderDate) VALUES (1, CURRENT_DATE());
SELECT * FROM Orders WHERE OrderDate = '2026-10-06';
```

> 💡 Poređenje datuma je jednostavno dok kolona sadrži samo datum. Ako je kolona tipa DATETIME, uslov OrderDate = '2026-10-06' ne pronalazi redove sa vremenom različitim od ponoći – zato se za datume bez vremena bira DATE.

18.5 Ograničenja (constraints)

Ograničenja su pravila nad podacima u koloni ili tabeli; vrste i njihova uloga opisani su u odeljku 14.4, a ključevi u poglavlju 12. Ovde je prikazano kako se zadaju u MySQL-u. Ograničenje se navodi pri kreiranju tabele ili se dodaje kasnije naredbom ALTER TABLE.

NOT NULL

ŠABLON

```
kolona tip_podatka NOT NULL
```

PRIMER

```
CREATE TABLE Persons (
  ID        INT NOT NULL,
  LastName  VARCHAR(255) NOT NULL,
  FirstName VARCHAR(255) NOT NULL,
  Age       INT
);
```

UNIQUE

ŠABLON

```
UNIQUE (kolona)
CONSTRAINT ime_ogranicenja UNIQUE (kolona1, kolona2)   -- imenovano, nad više kolona
```

PRIMER

```
CREATE TABLE Persons (
  ID       INT NOT NULL,
  LastName VARCHAR(255) NOT NULL,
  Email    VARCHAR(255),
  UNIQUE (Email)
);
```

PRIMARY KEY

ŠABLON

```
PRIMARY KEY (kolona)
CONSTRAINT ime_ogranicenja PRIMARY KEY (kolona1, kolona2)   -- složeni ključ
```

PRIMER

```
CREATE TABLE Persons (
  ID       INT NOT NULL,
  LastName VARCHAR(255) NOT NULL,
  PRIMARY KEY (ID)
);
```

AUTO_INCREMENT

```
kolona INT NOT NULL AUTO_INCREMENT
```

Kolona automatski dobija sledeći broj pri svakom unosu; brojanje počinje od 1. Početna vrednost se menja naredbom ALTER TABLE ime_tabele AUTO_INCREMENT = 100;. Ovako se pravi surogat ključ (odeljak 12.10).

```
CREATE TABLE Persons (
  PersonID INT NOT NULL AUTO_INCREMENT,
  LastName VARCHAR(255) NOT NULL,
  PRIMARY KEY (PersonID)
);

INSERT INTO Persons (LastName) VALUES ('Petrović');   -- PersonID = 1
```

FOREIGN KEY

ŠABLON

```
FOREIGN KEY (kolona) REFERENCES druga_tabela(kolona_kljuca)
  [ON DELETE CASCADE | SET NULL | RESTRICT]
```

PRIMER

```
CREATE TABLE Orders (
  OrderID     INT NOT NULL,
  OrderNumber INT NOT NULL,
  PersonID    INT,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

> 💡 Ovako se u MySQL-u zapisuje rezultat prevođenja ER modela iz poglavlja 13 i 15: veza 1:N postaje strani ključ u tabeli strane N. Tabela na koju se upućuje (Persons) mora postojati pre tabele koja upućuje (Orders). Bez ON DELETE, red iz Persons koji ima porudžbine ne može se obrisati.

CHECK

ŠABLON

```
CHECK (uslov)
CONSTRAINT ime_ogranicenja CHECK (uslov1 AND uslov2)
```

PRIMER

```
CREATE TABLE Persons (
  ID   INT NOT NULL,
  Age  INT,
  City VARCHAR(255),
  CHECK (Age >= 18)
);
```

> 💡 MySQL proverava CHECK ograničenja tek od verzije 8.0.16; starije verzije ih prihvataju u sintaksi, ali ih ne primenjuju.

DEFAULT

ŠABLON

```
kolona tip_podatka DEFAULT vrednost
```

PRIMER

```
CREATE TABLE Orders (
  ID        INT NOT NULL,
  City      VARCHAR(255) DEFAULT 'Beograd',
  OrderDate DATE DEFAULT (CURRENT_DATE)
);
```

💡 Izraz kao podrazumevana vrednost (npr. CURRENT_DATE) piše se u zagradama; to je podržano od MySQL 8.0.13.

18.6 Izmena tabele (ALTER TABLE)

```
ALTER TABLE ime_tabele ADD kolona tip_podatka;               -- nova kolona
ALTER TABLE ime_tabele DROP COLUMN kolona;                   -- brisanje kolone
ALTER TABLE ime_tabele MODIFY COLUMN kolona novi_tip;        -- promena tipa
ALTER TABLE ime_tabele RENAME COLUMN staro_ime TO novo_ime;  -- preimenovanje
ALTER TABLE ime_tabele ADD CONSTRAINT ime ograničenje;      -- novo ograničenje
```

Primer: dodavanje i izmena kolone u tabeli Persons.

```
ALTER TABLE Persons ADD DateOfBirth DATE;
ALTER TABLE Persons MODIFY COLUMN DateOfBirth YEAR;
ALTER TABLE Persons DROP COLUMN DateOfBirth;

ALTER TABLE Persons ADD CONSTRAINT UC_Email UNIQUE (Email);
ALTER TABLE Orders  ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

18.7 Indeksi (CREATE INDEX)

ŠABLON

```
CREATE INDEX ime_indeksa ON ime_tabele (kolona1, kolona2, ...);
CREATE UNIQUE INDEX ime_indeksa ON ime_tabele (kolona);
ALTER TABLE ime_tabele DROP INDEX ime_indeksa;
```

PRIMER

```
CREATE INDEX idx_lastname ON Persons (LastName);
```

Indeks ubrzava pretraživanje po koloni, ali usporava unos i izmenu, jer se i indeks mora ažurirati. Pravi se nad kolonama po kojima se često pretražuje – to je sekundarni ključ iz odeljka 12.7. Za primarni ključ i UNIQUE kolone MySQL indeks pravi sam.

18.8 Brisanje tabele (DROP i TRUNCATE)

```
DROP TABLE [IF EXISTS] ime_tabele;    -- briše tabelu i sve podatke
TRUNCATE TABLE ime_tabele;            -- briše sve redove, tabela ostaje
```

Tabela 36. *DROP naspram TRUNCATE*

	DROP TABLE	TRUNCATE TABLE
Struktura	nestaje	ostaje
Podaci	nestaju	nestaju
AUTO_INCREMENT	–	vraća se na početak

	DROP TABLE	TRUNCATE TABLE
Kada	tabela više nije potrebna	tabelu treba isprazniti

> 💡 Ovo je i odgovor na pitanje iz prve prezentacije: TRUNCATE je DDL, jer radi nad tabelom kao celinom, a DELETE je DML, jer briše red po red i može imati uslov WHERE.

18.9 Pregled DDL naredbi

Tabela 37. *DDL naredbe u MySQL-u*

Naredba	Svrha
CREATE DATABASE / DROP DATABASE	pravljenje i brisanje baze
USE	izbor baze za rad
CREATE TABLE	nova tabela sa kolonama, tipovima i ograničenjima
ALTER TABLE	dodavanje, izmena, brisanje kolona i ograničenja
CREATE INDEX	ubrzanje pretraživanja
DROP TABLE	brisanje tabele
TRUNCATE TABLE	pražnjenje tabele
DESCRIBE	pregled strukture tabele

18.10 Zadaci: DDL i tipovi podataka

Zadaci se nadovezuju jedan na drugi i rade se u istoj bazi.

Zadatak 1 – Baza za vežbe

Napravite bazu biblioteka, izaberite je i proverite da se nalazi na spisku baza.

Rešenje:

```
CREATE DATABASE biblioteka;
USE biblioteka;
SHOW DATABASES;
```

Zadatak 2 – Izbor tipova

Napravite tabelu IZDAVAC: PIB (tačno 9 cifara, ključ), Naziv (do 100 znakova, obavezan) i Grad (do 50 znakova, podrazumevano Beograd). Obrazložite izbor tipa za PIB.

Rešenje:

```
CREATE TABLE IZDAVAC (
  PIB   CHAR(9) NOT NULL,
  Naziv VARCHAR(100) NOT NULL,
  Grad  VARCHAR(50) DEFAULT 'Beograd',
  PRIMARY KEY (PIB)
);
```

> 💡 PIB je CHAR(9), a ne INT: uvek ima tačno 9 znakova, sa njim se ne računa, a broj koji počinje nulom kao INT bi izgubio vodeću nulu.

Zadatak 3 – Strani ključ i ograničenja

Napravite tabelu KNJIGA: ISBN (13 znakova, ključ), Naslov (obavezan), GodinaIzdanja (godina), Cena (novčani iznos sa dve decimale, ne manja od 0) i PIB izdavača kao strani ključ. Ovo je tabela KNJIGA iz rešenja zadatka 15.1.

Rešenje:

```
CREATE TABLE KNJIGA (
  ISBN          CHAR(13) NOT NULL,
  Naslov        VARCHAR(200) NOT NULL,
  GodinaIzdanja YEAR,
  Cena          DECIMAL(8, 2) CHECK (Cena >= 0),
  PIB           CHAR(9) NOT NULL,
  PRIMARY KEY (ISBN),
  FOREIGN KEY (PIB) REFERENCES IZDAVAC(PIB)
);
```

> 💡 PIB je NOT NULL jer knjiga u ER modelu ima potpuno učešće u vezi IZDAJE. Cena je DECIMAL, a ne FLOAT, jer se za novac traži tačna vrednost.

Zadatak 4 – Izmena postojeće tabele

Tabeli KNJIGA naknadno dodajte kolonu DatumNabavke (datum, podrazumevano današnji), zatim promenite dužinu kolone Naslov na 300 znakova i proverite strukturu tabele.

Rešenje:

```
ALTER TABLE KNJIGA ADD DatumNabavke DATE DEFAULT (CURRENT_DATE);
ALTER TABLE KNJIGA MODIFY COLUMN Naslov VARCHAR(300) NOT NULL;
DESCRIBE KNJIGA;
```

> 💡 Uz MODIFY COLUMN treba ponoviti i NOT NULL; u suprotnom kolona postaje opciona.

Zadatak 5 – Pražnjenje i brisanje

Posle probnog unosa, obrišite sve redove iz KNJIGA tako da struktura ostane. Zatim pokušajte da obrišete tabelu IZDAVAC i objasnite šta se dešava i kako se tabele brišu ispravnim redosledom.

Rešenje:

```
TRUNCATE TABLE KNJIGA;

DROP TABLE IZDAVAC;   -- greška: na IZDAVAC upućuje strani ključ iz KNJIGA

DROP TABLE IF EXISTS KNJIGA;
DROP TABLE IF EXISTS IZDAVAC;
```

> 💡 Tabela na koju upućuje strani ključ ne može se obrisati dok postoji tabela koja na nju upućuje. Brišu se obrnutim redosledom od kreiranja: prvo KNJIGA, pa IZDAVAC.

19. Upravljanje korisnicima u MySQL-u

U poglavlju 1.6 jezik DCL (GRANT, REVOKE) opisan je u opštem obliku. Ovde je prikazan ceo životni ciklus korisničkog naloga u MySQL-u: kreiranje, dodela prava, promena lozinke i brisanje.

19.1 Kreiranje korisnika (CREATE USER)

MySQL nalog čine korisničko ime i računar sa kog je prijava dozvoljena: **'username'@'host'**. Podaci o nalozima čuvaju se u sistemskoj tabeli **mysql.user**.

```
CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';

-- jedan korisnik, prijava samo sa lokalnog računara
CREATE USER 'gfguser1'@'localhost' IDENTIFIED BY 'abcd';

-- više korisnika jednom naredbom
CREATE USER 'gfguser2'@'localhost' IDENTIFIED BY 'efgh',
            'gfguser3'@'localhost' IDENTIFIED BY 'uvxy',
            'gfguser4'@'localhost' IDENTIFIED BY 'ijkl';

-- prijava sa bilo kog računara (% je džoker)
CREATE USER 'gfguser_net'@'%' IDENTIFIED BY 'abcd';
```

> 💡 CREATE USER pravi nalog, ali mu **ne daje nikakva prava** nad podacima. Prava se dodeljuju naredbom GRANT.

19.2 Pregled prava (SHOW GRANTS)

```
SHOW GRANTS FOR 'gfguser1'@'localhost';

+-----------------------------------------------+
| Grants for gfguser1@localhost                 |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `gfguser1`@`localhost`  |
+-----------------------------------------------+
```

- **USAGE** znači „bez privilegija”: korisnik sme samo da se prijavi na server.
- **Oznaka posle ON** ima oblik baza.tabela: zvezdica na mestu baze znači sve baze, a na mestu tabele sve tabele u njima.

19.3 Dodela i oduzimanje prava (GRANT, REVOKE)

```
-- sva prava nad jednom bazom
GRANT ALL PRIVILEGES ON naziv_baze.* TO 'gfguser1'@'localhost';

-- samo čitanje i unos nad jednom tabelom (čest izbor za aplikativni nalog)
GRANT SELECT, INSERT ON naziv_baze.StudentDetails TO 'gfguser1'@'localhost';

-- oduzimanje prava
REVOKE INSERT ON naziv_baze.StudentDetails FROM 'gfguser1'@'localhost';
```

> 💡 Prava dodeljena naredbama GRANT i REVOKE važe odmah. **FLUSH PRIVILEGES** je potreban samo kada se sistemske tabele prava (npr. mysql.user) menjaju direktno, naredbama INSERT ili UPDATE.

19.4 Promena lozinke

Tabela 38. *Načini promene lozinke*

Metoda	Naredba	Status u MySQL 8
ALTER USER	ALTER USER 'gfguser1'@'localhost' IDENTIFIED BY 'newpass';	**preporučeno**
SET PASSWORD	SET PASSWORD FOR 'gfguser1'@'localhost' = 'newpass';	podržano
UPDATE mysql.user	direktna izmena sistemske tabele + FLUSH PRIVILEGES	*ne koristiti*

> 💡 Direktan UPDATE nad mysql.user zaobilazi heširanje lozinke (caching_sha2_password), a funkcija PASSWORD() uklonjena je u MySQL 8.0. Lozinka se zato menja isključivo naredbom ALTER USER (ili SET PASSWORD).

19.5 Brisanje korisnika (DROP USER)

DROP USER trajno uklanja nalog i automatski poništava sva njegova prava. Host mora tačno odgovarati onom pri kreiranju (npr. '%' za naloge sa džokerom). Pođimo od četiri naloga gfguser1–gfguser4 na localhost-u:

```
DROP USER 'gfguser1'@'localhost';
Query OK, 0 rows affected (0.01 sec)      -- ostaju gfguser2, gfguser3, gfguser4

DROP USER IF EXISTS 'gfguser1'@'localhost', 'gfguser2'@'localhost';
Query OK, 0 rows affected (0.01 sec)      -- ostaju gfguser3, gfguser4
```

- **0 rows affected** je normalan odgovor kod brisanja naloga, iako je nalog obrisan.
- **Bez IF EXISTS**, ako bilo koji nalog sa liste ne postoji, javlja se greška ERROR 1396 i ne briše se nijedan. Sa IF EXISTS nepostojeći nalog (ovde već obrisani gfguser1) samo se preskače.
- **Aktivna sesija** obrisanog korisnika ne prekida se odmah; korisnik radi do odjave, a posle toga se više ne može prijaviti.

19.6 Ko je prijavljen: USER() i srodne funkcije

Funkcija **USER()** nema parametre i vraća nalog trenutne konekcije u obliku 'username'@'hostname'. Koristi se za reviziju (ko je šta menjao), otklanjanje problema sa pravima i nadzor konekcija.

```
SELECT USER();                                  -- npr. john@localhost

INSERT INTO user_logs (username, log_time)
VALUES (USER(), NOW());                         -- beleženje aktivnosti
```

Tabela 39. *Funkcije za identifikaciju korisnika*

Funkcija	Vraća
USER()	korisnika i host kojima se klijent predstavio pri prijavi

Funkcija	Vraća
CURRENT_USER()	nalog prema kom MySQL proverava prava; može se razlikovati od USER() kod anonimnih naloga ili procedura sa DEFINER pravima
SESSION_USER()	sinonim za USER()

> 💡 U običnoj konekciji sve tri funkcije vraćaju istu vrednost; razlika se vidi kod anonimnih naloga i skladištenih procedura sa DEFINER pravima.

19.7 Pregled naredbi

Tabela 40. *Životni ciklus korisničkog naloga*

Operacija	Naredba	Svrha
Kreiranje	CREATE USER 'u'@'h' IDENTIFIED BY 'p';	nov nalog, bez prava
Pregled prava	SHOW GRANTS FOR 'u'@'h';	dodeljene privilegije
Dodela prava	GRANT SELECT, INSERT ON baza.tabela TO 'u'@'h';	dozvola rada sa podacima
Oduzimanje prava	REVOKE INSERT ON baza.tabela FROM 'u'@'h';	povlačenje dozvole
Promena lozinke	ALTER USER 'u'@'h' IDENTIFIED BY 'p2';	nova lozinka
Brisanje	DROP USER IF EXISTS 'u'@'h';	uklanjanje naloga i svih prava

19.8 Zadaci: upravljanje korisnicima

Pet kratkih scenarija iz prakse. Za svaki napišite SQL naredbe, pa proverite rešenje. Lozinke u primerima su ilustrativne.

Zadatak 1 – Nalog za veb-prodavnicu

Aplikacija veb-prodavnice radi na istom serveru kao baza **prodavnica**. Napravite nalog shop_app koji sme da čita, unosi i menja podatke u svim tabelama te baze, ali ne sme da briše redove niti da menja strukturu tabela. Proverite dodeljena prava.

Rešenje:

```
CREATE USER 'shop_app'@'localhost' IDENTIFIED BY 'Shop#2026';
GRANT SELECT, INSERT, UPDATE ON prodavnica.* TO 'shop_app'@'localhost';
SHOW GRANTS FOR 'shop_app'@'localhost';
```

> 💡 Host je localhost jer aplikacija radi na istom računaru. Nema DELETE ni DDL prava (CREATE, ALTER, DROP): aplikacija dobija samo ono što joj treba – **princip najmanjih privilegija**. GRANT ALL bi ovde bio greška.

Zadatak 2 – Asistent iz računarske laboratorije

Asistent pristupa bazi **fakultet** sa bilo kog računara u laboratoriji, čije su adrese 192.168.1.1 do 192.168.1.254. Treba da vidi samo tabelu **ocene**, bez ikakvih izmena.

Rešenje:

```
CREATE USER 'asistent'@'192.168.1.%' IDENTIFIED BY 'Lab#2026';
GRANT SELECT ON fakultet.ocene TO 'asistent'@'192.168.1.%';
```

💡 Džoker % u delu host ograničava prijavu na mrežu laboratorije. Pravo je dato na nivou jedne tabele (fakultet.ocene), a ne cele baze (fakultet.*).

Zadatak 3 – Previše dodeljenih prava

Praktikantu ('praktikant'@'localhost') greškom su data prava SELECT, INSERT i DELETE nad tabelom **fakultet.studenti**. Oduzmite mu pravo brisanja, zadržite ostala i proverite rezultat.

Rešenje:

```
REVOKE DELETE ON fakultet.studenti FROM 'praktikant'@'localhost';
SHOW GRANTS FOR 'praktikant'@'localhost';
-- GRANT SELECT, INSERT ON `fakultet`.`studenti` TO `praktikant`@`localhost`
```

💡 REVOKE oduzima samo navedeno pravo; nalog i ostala prava ostaju. Izmena važi odmah, FLUSH PRIVILEGES nije potreban.

Zadatak 4 – Procurela lozinka

Lozinka naloga shop_app iz zadatka 1 slučajno je objavljena u javnom repozitorijumu koda. Odmah postavite novu lozinku i podesite da ubuduće lozinka ovog naloga ističe na svakih 90 dana.

Rešenje:

```
ALTER USER 'shop_app'@'localhost' IDENTIFIED BY 'N0va#Lozinka!';
ALTER USER 'shop_app'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;
```

💡 Lozinka se menja naredbom ALTER USER, ne direktnim UPDATE-om nad mysql.user. Posle promene treba ažurirati i konfiguraciju aplikacije, inače se ona više neće povezati na bazu.

Zadatak 5 – Kraj semestra

Na kraju semestra treba ukloniti naloge demonstratora demo1, demo2 i demo3 (svi na localhost-u). Nije sigurno da nalog demo3 uopšte postoji, a skripta se ne sme prekinuti greškom. Posle brisanja proverite da nijedan nalog nije ostao.

Rešenje:

```
DROP USER IF EXISTS 'demo1'@'localhost', 'demo2'@'localhost', 'demo3'@'localhost';

SELECT user, host FROM mysql.user WHERE user LIKE 'demo%';
-- Empty set
```

💡 Bez IF EXISTS, nepostojeći demo3 izazvao bi grešku ERROR 1396 i ne bi bio obrisan nijedan nalog. Uz DROP USER nestaju i sva prava tih naloga, pa nije potreban poseban REVOKE.

20. Naredba SELECT: zadaci nad bazom w3schools

Naredba SELECT čita podatke iz tabela; pripada DQL-u iz odeljka 1.6, a njena teorijska osnova su selekcija (σ) i projekcija (π) iz poglavlja 10. Za vežbu se koristi baza iz MySQL i SQL tutorijala sajta w3schools.com, pa se svaki upit može uporediti sa primerima na tom sajtu.

20.1 Baza za vežbe

SQL skriptu w3schools.sql dobijate od nastavnika. Kada se izvrši, pravi bazu **w3schools** sa osam tabela i podacima. Uvoz u Workbench-u: **File → Open SQL Script…**, izaberite fajl i izvršite ga ikonom munje; ili iz terminala:

```
mysql -u root -p < w3schools.sql
```

Tabela 41. *Tabele baze w3schools*

Tabela	Kolone	Redova
customers	CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country	91
products	ProductID, ProductName, SupplierID, CategoryID, Unit, Price	77
orders	OrderID, CustomerID, EmployeeID, OrderDate, ShipperID	196
order_details	OrderDetailID, OrderID, ProductID, Quantity	518
categories	CategoryID, CategoryName, Description	8
employees	EmployeeID, LastName, FirstName, BirthDate, Photo, Notes	10
suppliers	SupplierID, SupplierName, ContactName, Address, City, PostalCode, Country, Phone	29
shippers	ShipperID, ShipperName, Phone	3

> 💡 U ovoj skripti imena tabela su malim slovima (customers, a ne Customers kao na sajtu). Na Windows-u MySQL podrazumevano ne razlikuje velika i mala slova u imenima tabela, ali na Linux-u razlikuje – zato je najsigurnije pisati ih tačno kao u skripti.

20.2 Šablon naredbe SELECT

```
SELECT [DISTINCT] kolona1, kolona2, ...   -- ili * za sve kolone
FROM ime_tabele
[WHERE uslov]
[ORDER BY kolona [ASC | DESC], ...]
[LIMIT broj];
```

Tabela 42. *Delovi naredbe SELECT*

Deo upita	Uloga
SELECT kolone	koje kolone se prikazuju – projekcija (π)
DISTINCT	uklanja ponovljene redove iz rezultata

Deo upita	Uloga
WHERE uslov	koji redovi ulaze u rezultat – selekcija (σ)
ORDER BY	redosled rezultata; ASC rastuće (podrazumevano), DESC opadajuće
LIMIT n	najviše n redova rezultata
LIKE 'a%'	poređenje sa šablonom: % je bilo koji niz znakova, _ tačno jedan znak
IN (…) / BETWEEN a AND b	vrednost iz liste / vrednost u opsegu, uključujući granice

> 💡 Redosled delova u upitu je obavezan: SELECT, FROM, WHERE, ORDER BY, LIMIT. Tekst se piše u apostrofima ('Germany'), brojevi bez njih (Price > 20).

20.3 Zadaci: SELECT

Svi zadaci rade se u bazi w3schools (USE w3schools;). Uz rešenje je naveden očekivani rezultat, pa studenti mogu sami da provere upit.

Zadatak 1 – Kupci iz Nemačke

Prikažite naziv i grad svih kupaca iz Nemačke, sortirano po gradu, a u okviru istog grada po nazivu kupca.

Rešenje:

```
SELECT CustomerName, City
FROM customers
WHERE Country = 'Germany'
ORDER BY City, CustomerName;
```

> 💡 Očekivano: 11 redova; prvi je Drachenblut Delikatessend (Aachen), poslednji Die Wandernde Kuh (Stuttgart). ORDER BY sa dve kolone sortira po drugoj samo kada su vrednosti prve jednake.

Zadatak 2 – Proizvodi srednje cene

Prikažite naziv i cenu proizvoda čija je cena od 20 do 30 (uključujući granice), od najskupljeg ka najjeftinijem; proizvode iste cene poređajte po nazivu.

Rešenje:

```
SELECT ProductName, Price
FROM products
WHERE Price BETWEEN 20 AND 30
ORDER BY Price DESC, ProductName;
```

> 💡 Očekivano: 14 redova, od Uncle Bob's Organic Dried Pears (30) do Maxilaku (20). Oba proizvoda sa granične cene su u rezultatu, jer BETWEEN uključuje granice; isti uslov se može napisati kao Price >= 20 AND Price <= 30.

Zadatak 3 – Iz kojih zemalja su kupci?

Prikažite abecedni spisak zemalja iz kojih dolaze kupci, bez ponavljanja. Zatim izvršite isti upit bez DISTINCT i uporedite broj redova.

Rešenje:

```
SELECT DISTINCT Country
FROM customers
ORDER BY Country;
```

> 💡 Očekivano: 21 red, od Argentina do Venezuela. Bez DISTINCT upit vraća 91 red, po jedan za svakog kupca, pa se npr. Germany ponavlja 11 puta.

Zadatak 4 – Najskuplji proizvodi iz dve kategorije

Prikažite naziv, kategoriju i cenu pet najskupljih proizvoda iz kategorija 1 (Beverages) i 2 (Condiments).

Rešenje:

```
SELECT ProductName, CategoryID, Price
FROM products
WHERE CategoryID IN (1, 2)
ORDER BY Price DESC
LIMIT 5;
```

> 💡 Očekivano: 5 redova – Côte de Blaye (263.5), Ipoh Coffee (46), Vegie-spread (43.9), Northwoods Cranberry Sauce (40) i Sirop d'érable (28.5). LIMIT ima smisla tek uz ORDER BY: bez sortiranja bi upit vratio bilo kojih pet proizvoda.

Zadatak 5 – Kupci iz tri prestonice

Prikažite naziv i grad kupaca iz Londona, Pariza i Madrida čiji naziv ne počinje slovom B, sortirano po gradu, pa po nazivu.

Rešenje:

```
SELECT CustomerName, City
FROM customers
WHERE City IN ('London', 'Paris', 'Madrid')
  AND CustomerName NOT LIKE 'B%'
ORDER BY City, CustomerName;
```

> 💡 Očekivano: 9 redova (5 iz Londona, 2 iz Madrida, 2 iz Pariza). Bez drugog uslova bilo bi 11 – izostaju B's Beverages i Bólido Comidas preparadas. IN zamenjuje tri uslova povezana sa OR.

21. Aliasi, agregatne funkcije i pogledi: zadaci

Zadaci iz ovog poglavlja rade se na času računskih vežbi, nad istom bazom w3schools kao u poglavlju 20 (USE w3schools;). Uz svako rešenje naveden je očekivani rezultat, pa studenti mogu sami da provere upit. Svi rezultati su provereni izvršavanjem upita.

21.1 Podsetnik: šabloni

ŠABLON

```
SELECT kolona AS alias, FUNKCIJA(kolona) AS alias
FROM ime_tabele
[WHERE uslov_nad_redovima]
[GROUP BY kolona]
[HAVING uslov_nad_grupama]
[ORDER BY kolona_ili_alias [ASC | DESC]]
[LIMIT broj];
```

ŠABLON

```
CREATE VIEW ime_pogleda AS SELECT ...;
CREATE OR REPLACE VIEW ime_pogleda AS SELECT ...;
DROP VIEW [IF EXISTS] ime_pogleda;
```

Tabela 43. *Agregatne funkcije*

Funkcija	Vraća
COUNT(*)	broj svih redova
COUNT(kolona)	broj redova u kojima kolona nije NULL
COUNT(DISTINCT kolona)	broj različitih vrednosti
SUM(kolona) / AVG(kolona)	zbir / prosek (samo brojevi; NULL se preskače)
MIN(kolona) / MAX(kolona)	najmanju / najveću vrednost (i tekst i datumi)

> 💡 Logički redosled izvršavanja je FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY, LIMIT. Zato alias iz SELECT-a sme da se koristi u ORDER BY, ali ne i u WHERE; uslov nad agregatnom funkcijom piše se u HAVING, a ne u WHERE. Uz GROUP BY, u SELECT-u smeju da stoje samo kolone po kojima se grupiše i agregatne funkcije.

21.2 Zadaci

Zadatak 1 – Cenovnik sa PDV-om

Za proizvode iz kategorije 2 (Condiments) prikažite naziv, cenu i cenu sa PDV-om od 20 %, zaokruženu na dve decimale. Kolone nazovite Proizvod, Cena i CenaSaPDV i sortirajte od najveće cene sa PDV-om.

Rešenje:

```
SELECT ProductName AS Proizvod,
       Price AS Cena,
       ROUND(Price * 1.2, 2) AS CenaSaPDV
FROM products
WHERE CategoryID = 2
ORDER BY CenaSaPDV DESC;
```

> 💡 Očekivano: 12 redova, od Vegie-spread (43.9 → 52.68) do Aniseed Syrup (10 → 12.00). Alias CenaSaPDV sme da se koristi u ORDER BY; u WHERE bi izazvao grešku 1054.

Zadatak 2 – Skupi proizvodi

Za proizvode skuplje od 50 izračunajte koliko ih ima, najnižu i najvišu cenu i prosečnu cenu zaokruženu na dve decimale. Kolonama dajte čitljiva imena.

Rešenje:

```
SELECT COUNT(*)             AS BrojProizvoda,
       MIN(Price)           AS Najniza,
       MAX(Price)           AS Najvisa,
       ROUND(AVG(Price), 2) AS Prosek
FROM products
WHERE Price > 50;
```

> 💡 Očekivano: jedan red – 7 proizvoda, najniža cena 53, najviša 263.5, prosek 105.11. WHERE se primenjuje pre agregacije, pa funkcije računaju samo nad 7 izabranih proizvoda.

Zadatak 3 – Pregled porudžbina

Jednim upitom izračunajte ukupan broj porudžbina, broj različitih kupaca koji su poručivali i datum prve i poslednje porudžbine.

Rešenje:

```
SELECT COUNT(*)                  AS BrojPorudzbina,
       COUNT(DISTINCT CustomerID) AS BrojKupaca,
       MIN(OrderDate)             AS Prva,
       MAX(OrderDate)             AS Poslednja
FROM orders;
```

> 💡 Očekivano: 196 porudžbina od 74 različita kupca, od 1996-07-04 do 1997-02-12. Od 91 kupca, 17 nije poručivalo ništa. MIN i MAX rade i nad datumima.

Zadatak 4 – Dobavljači po zemljama

Za svaku zemlju prikažite koliko dobavljača iz nje posluje sa prodavnicom, od zemlje sa najviše dobavljača; zemlje sa istim brojem poređajte abecedno.

Rešenje:

```
SELECT Country  AS Zemlja,
       COUNT(*) AS BrojDobavljaca
FROM suppliers
GROUP BY Country
ORDER BY BrojDobavljaca DESC, Zemlja;
```

> 💡 Očekivano: 16 redova; prvi je USA (4), zatim France i Germany (po 3), a poslednji su zemlje sa po jednim dobavljačem, do Spain. Ukupan zbir kolone BrojDobavljaca je 29, koliko ima redova u tabeli suppliers.

Zadatak 5 – Najprodavaniji proizvodi

Iz tabele stavki porudžbina prikažite pet proizvoda (ProductID) sa najvećom ukupnom naručenom količinom.

Rešenje:

```
SELECT ProductID,
       SUM(Quantity) AS UkupnoNaruceno
FROM order_details
GROUP BY ProductID
ORDER BY UkupnoNaruceno DESC
LIMIT 5;
```

> 💡 Očekivano: proizvodi 31 (458), 60 (430), 35 (369), 59 (346) i 2 (341). Naziv proizvoda nalazi se u drugoj tabeli; da bi se prikazao uz zbir, potrebno je spajanje tabela (JOIN), koje se radi kasnije.

Zadatak 6 – Gradovi sa više kupaca

Među kupcima iz Nemačke, Francuske, Španije i Velike Britanije pronađite gradove u kojima ima najmanje dva kupca. Prikažite grad, zemlju i broj kupaca, od grada sa najviše kupaca.

Rešenje:

```
SELECT City     AS Grad,
       Country  AS Zemlja,
       COUNT(*) AS BrojKupaca
FROM customers
WHERE Country IN ('Germany', 'France', 'Spain', 'UK')
GROUP BY City, Country
HAVING COUNT(*) >= 2
ORDER BY BrojKupaca DESC, Grad;
```

> 💡 Očekivano: 4 reda – London (6), Madrid (3), Nantes (2) i Paris (2). WHERE bira kupce pre grupisanja, HAVING bira gradove posle grupisanja. Grupiše se po gradu i zemlji, jer se obe kolone prikazuju.

Zadatak 7 – Pogled za izveštaj o zaposlenima

Napravite pogled porudzbine_po_zaposlenom koji za svakog zaposlenog (EmployeeID) sadrži broj porudžbina i datum njegove prve porudžbine. Zatim iz pogleda prikažite zaposlene sa više od 20 porudžbina, od najuspešnijeg.

Rešenje:

```
CREATE VIEW porudzbine_po_zaposlenom AS
SELECT EmployeeID,
       COUNT(*)       AS BrojPorudzbina,
       MIN(OrderDate) AS PrvaPorudzbina
FROM orders
GROUP BY EmployeeID;

SELECT *
FROM porudzbine_po_zaposlenom
WHERE BrojPorudzbina > 20
ORDER BY BrojPorudzbina DESC;
```

💡 Očekivano: pogled ima 9 redova (jedan zaposleni nema nijednu porudžbinu), a upit vraća 4 – zaposleni 4 (40), 3 (31), 1 (29) i 8 (27). U upitu nad pogledom BrojPorudzbina je obična kolona pogleda, pa se uslov piše u WHERE.

Zadatak 8 – Pogled i nalog za odeljenje prodaje

Odeljenje prodaje sme da vidi samo naziv kupca, ime kontakt osobe i zemlju – ne i adrese. Napravite pogled kupci_kontakt sa te tri kolone i nalog prodaja (samo lokalno) koji sme da čita isključivo taj pogled. Proverite prava, a zatim, prijavljeni kao prodaja, prikažite kupce iz Švedske i pokušajte da pročitate adresu iz tabele customers.

Rešenje:

```
-- kao root
CREATE VIEW kupci_kontakt AS
SELECT CustomerName, ContactName, Country
FROM customers;

CREATE USER 'prodaja'@'localhost' IDENTIFIED BY 'Prodaja#2026';
GRANT SELECT ON w3schools.kupci_kontakt TO 'prodaja'@'localhost';
SHOW GRANTS FOR 'prodaja'@'localhost';

-- kao prodaja
SELECT * FROM kupci_kontakt WHERE Country = 'Sweden';
SELECT Address FROM customers LIMIT 1;
```

💡 Očekivano: SHOW GRANTS pored USAGE prikazuje GRANT SELECT ON w3schools.kupci_kontakt. Upit nad pogledom vraća 2 kupca iz Švedske, a čitanje tabele customers završava se greškom 1142 (SELECT command denied). Na kraju se pogledi i nalog uklanjaju: DROP VIEW IF EXISTS kupci_kontakt, porudzbine_po_zaposlenom; DROP USER IF EXISTS 'prodaja'@'localhost';