1 Oblikovanje relacijskih modela baze podataka

Nakon ove cjeline moći ćeš:

- analizirati jednostavne korisničke zahtjeve i prepoznati elemente važne za oblikovanje baze podataka i odrediti njihovu povezanost
- nacrtati E-R dijagram s naznačenim vrstama veza
- pretvoriti E-R model u relacijski model i zapisati relacijsku shemu baze
- prepoznati probleme u postojećem relacijskom modelu i predložiti rješenja te ih primijeniti na primjeru.

1.1. E-R model baze podataka

Start

Mia i Leo dobili su zadatak osmisliti sustav za školsku knjižnicu. Prije nego što počnu pisati SQL kod i izrađivati stvarnu bazu podataka, trebaju najprije jasno prikazati kako su podatci međusobno povezani.

Leo je predložio da odmah započnu s izradom tablica, ali Mia je istaknula da bi to moglo dovesti do pogrešaka ako nemaju jasan plan. Zato su odlučili najprije napraviti E-R model, odnosno dijagram koji prikazuje glavne dijelove sustava i njihove veze.

Počeli su razmišljati o tome koje sve cjeline postoje u knjižnici, tko sudjeluje u posudbi knjiga te kako su knjige i korisnici povezani. Postupno su izdvojili osnovne entitete i njihove odnose čime su dobili prvi pregled buduće baze podataka.

Na kraju su si postavili nekoliko pitanja:

1. Koje su osnovne cjeline (entiteti) u sustavu knjižnice?
2. Koje podatke trebamo zapisivati o svakoj cjelini?
3. Kako su te cjeline međusobno povezane?

Tako su napravili prvi korak u oblikovanju baze podataka – izradu E-R modela.

Zašto

Relacijski model baze podataka omogućava pohranjivanje podataka u povezanim tablicama čime se osiguravaju preglednost, dosljednost i jednostavno održavanje podataka. Svaka tablica predstavlja određenu skupinu podataka, a odnosi među njima ostvaruju se primarnim ili stranim ključevima. Tako se izbjegava ponavljanje informacija i olakšava njihovo ažuriranje. Relacijski model najčešće se primjenjuje u poslovnim sustavima, školama, bankama i internetskim aplikacijama gdje je potrebno učinkovito upravljati velikim količinama međusobno povezanih podataka.

Relacijski model baze podataka najrašireniji je način organiziranja i upravljanja podatcima u suvremenim informacijskim sustavima. Predstavio ga je **Edgar F. Codd** 1970. godine, a njegova glavna ideja bila je pohranjivanje podataka u obliku **tablica** (relacija) koje su međusobno povezane. Ovaj pristup pokazao se jednostavnim, logičnim i učinkovitim, zbog čega je postao standard u gotovo svim modernim sustavima za upravljanje bazama podataka (npr. MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server).

U relacijskom modelu **svi se podatci organiziraju u tablice**, a svaka tablica predstavlja jedan entitet iz stvarnog svijeta. Na primjer, ako želimo napraviti bazu podataka za fakultet, možemo imati tablicu *Studenti*, tablicu *Predmeti* i tablicu *Upisi*. Svaka tablica sastoji se od **redaka** i **stupaca**. **Redak** (koji se naziva i **zapis**) predstavlja **jedan konkretan primjerak entiteta**, npr. jednog studenta ili jedan predmet dok **stupac** (**atribut**) predstavlja neko **svojstvo tog entiteta**, npr. ime, prezime ili broj indeksa.

U svakoj tablici mora postojati atribut ili kombinacija atributa koji jednoznačno identificira svaki zapis. Taj se atribut naziva **primarni ključ**. Na primjer, u tablici *Studenti* primarni ključ može biti *broj_indeksa* jer svaki student ima jedinstven broj. Ovakav pristup omogućava da se svaki redak lako pronađe i da se podatci ne mogu duplicirati. Kada želimo povezati podatke iz dviju tablica, služimo se **stranim ključem** koji je atribut u jednoj tablici i upućuje na primarni ključ u drugoj. Tako se ostvaruje **veza** među tablicama i podatci se mogu kombinirati bez ponavljanja.

Relacijski model omogućuje da se podatci iz više tablica povezuju i obrađuju **odnosima** između zapisa u različitim tablicama. Najčešće vrste tih odnosa jesu:

- **jedan na jedan (1 : 1)** – jednom zapisu u prvoj tablici odgovara točno jedan zapis u drugoj
- **jedan na više (1 : N)** – jednom zapisu u prvoj tablici može odgovarati više zapisa u drugoj (npr. jedan student može imati više upisa)
- **više na više (M : N)** – više zapisa u jednoj tablici može biti povezano s više zapisa u drugoj (npr. više studenata može slušati više predmeta). Ovaj se odnos obično rješava pomoću dodatne tablice (tzv. spojne tablice).

Jedna je od najvećih prednosti relacijskog modela što korisnik ne mora znati kako su podatci fizički pohranjeni ni kako se dohvaćaju. Umjesto toga upotrebljava se **deklarativni jezik** – najčešće **SQL** (engl. *Structured Query Language*) kojim se jednostavno opisuje što želimo dohvatiti ili promijeniti. Na primjer, ako želimo pronaći sve studente koji su upisali predmet „Baze podataka“, dovoljno je napisati jednostavnu naredbu u SQL-u, a sustav će sam pronaći i spojiti podatke iz potrebnih tablica.

Primjer jednostavne tablice *Studenti* u relacijskom modelu prikazan je u tablici 1.1.

Tablica 1.1. Primjer tablice *Studenti*

broj_indeksa	ime	prezime	godina
100/2025	Ana	Kovač	2
101/2025	Marko	Horvat	1
102/2025	Ivana	Perić	3

Ovdje je svaki redak jedan student, a svaki stupac jedno svojstvo. Stupac *broj_indeksa* primarni je ključ jer je jedinstven za svakog studenta. Kad bismo imali još jednu tablicu, npr. *Upisi*, ona bi mogla sadržavati stupac *broj_indeksa* kao **strani ključ** koji povezuje zapis o upisu s određenim studentom. Tablica *Upisi* prikazana je tablicom 1.2.

Tablica 1.2. Tablica *Upisi*

upis_id	student_id	predmet_id	datum_upisa	status_upisa	ocjena
1	100/2025	201	2025-09-30	aktivan	NULL
2	101/2025	203	2025-09-28	položen	9
3	102/2025	202	2025-09-25	ponavlja	NULL

Vrlo važan dio relacijskog modela jesu i **ograničenja integriteta** koja osiguravaju da podatci u bazi ostanu ispravni i dosljedni. Primarni ključ osigurava da ne postoje dva zapisa s istim identifikatorom. Strani ključ osigurava da se vrijednost odnosi na postojeći zapis u drugoj tablici čime se održava tzv. **referencijalni integritet**. Osim toga, moguće je definirati i **ograničenja domena** kojima se određuje koje su vrijednosti dopuštene za pojedine atribute (npr. godina studija mora biti cijeli broj između 1 i 5).

Rad s relacijskim modelom podrazumijeva i nekoliko osnovnih koraka. Prvi je **analiza podataka**. Potrebno je prepoznati koje entitete želimo pohraniti i koja su njihova svojstva. Zatim se ti entiteti prikazuju kao tablice s odgovarajućim atributima. Nakon toga definiraju se **ključevi** i **odnosi** između tablica. Na kraju se pomoću SQL-a podatci unose, pretražuju, mijenjaju i brišu.

Relacijski model ima brojne prednosti:

- **jednostavan je za razumijevanje**
- **podatci su organizirani logično i pregledno**
- **lako se povezuju i pretražuju**
- **sustavi koji ga upotrebljavaju visoko su pouzdani i sigurni**
- **standardiziran je i široko podržan**

što znači da se znanja i vještine stečene radom s jednom relacijskom bazom mogu lako primijeniti i na druge sustave.

E-R (engl. *Entity-Relationship*) dijagram vizualno prikazuje strukturu baze podataka. Njegovi su osnovni elementi:

- **entiteti** (pravokutnici) – objekti o kojima se prikupljaju podatci
- **atributi** (elipse) – svojstva entiteta
- **veze** (rombovi) – odnosi između entiteta
- **brojnosti** – određuju broj mogućih veza između entiteta.

E-R dijagram prvi je korak u izradi baze jer omogućava da se stvarni svijet prikaže logički i pregledno. Nakon što se model završi, služi kao temelj za izradu relacijskih tablica u SQL-u.

U prikazu E-R modela baze podataka mogu se koristiti različite notacije. U ovom udžbeniku primjenjuje se **Chenova notacija** u kojoj se **entiteti** prikazuju **pravokutnicima**, **atributi elipsama**, a **odnosi rombovima**. Takav prikaz omogućuje jasno razlikovanje pojedinih elemenata modela i olakšava razumijevanje njihove uloge.

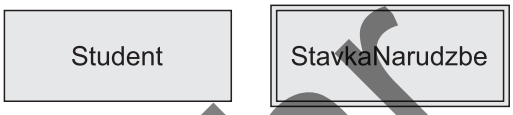
U praksi i suvremenim alatima za modeliranje baza podataka često se koristi i tzv. **Crow's Foot** (**Martinova**) notacija. Ona prikazuje **odnose** pomoću **linija** i posebnih oznaka na krajevima linija koje označavaju brojnost. Ova se notacija češće koristi u industriji.

Postoji i **Barkerova notacija** koja predstavlja **varijaciju Crow's Foot notacije** i primjenjuje se u nekim profesionalnim alatima, primjerice u sustavima koje razvija Oracle.

1.1.1. Entiteti

Entitet predstavlja stvar, osobu, pojam ili događaj o kojem želimo pohraniti podatke. Primjeri: *Student*, *Profesor*, *Predmet*, *Narudžba*.

- **Prikaz**: pravokutnik s nazivom entiteta (imenica u jednini). Na slici 1.1. prikazano je pravilno označavanje entiteta.



Slika 1.1. Pravilno označavanje entiteta

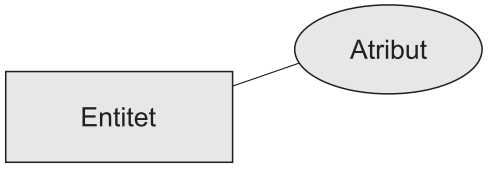
Vrste entiteta:

- **glavni** (**nezavisni**) – postoji samostalno (npr. *Student*)
- **slabi** – ovisi o drugom entitetu (npr. *StavkaNarudzbe*):
  - prikazuje se **dvostrukim pravokutnikom**
  - mora imati **identifikacijski odnos** s „najjačim" entitetom.

1.1.2. Atributi

Atribut opisuje svojstva entiteta ili odnosa. Primjeri: *JMBAG*, *ime*, *datum rođenja*, *ocjena*.

- **Prikaz**: elipsa povezana linijom s entitetom ili relacijom. Na slici 1.2. prikazano je povezivanje atributa s glavnim entitetom.



Slika 1.2. Povezivanje atributa s glavnim entitetom

Vrste atributa:

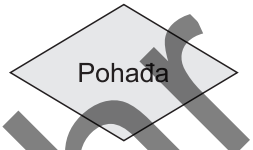
- **jednostavni** – ne dijeli se na manje dijelove (*ime*)
- **složeni** – sastoji se od više dijelova (*adresa* → *ulica*, *grad*, *poštanski broj*)
- **viševrijednosni** – može imati više vrijednosti (**telefonski_broj**) → **prikazuje se dvostrukom elipsom**
- **izvedeni** – može se izračunati iz drugih podataka (*starost* iz *datuma rođenja*) → prikazuje se isprekidanom elipsom.
- **ključni atribut** – jednoznačno identificira entitet (*JMBAG*, *OIB*) → **podcrtava se**.

1.1.3. Odnosi (relacije)

Odnos povezuje dva ili više entiteta.

Primjeri: *Student* – [pohađa] → *Predmet*, Kupac – [naručuje] → Narudžba.

- **Prikaz**: romb između entiteta. Na slici 1.3. prikazano je pravilno označavanje relacije.
- [Student] → (pohađa) → [Predmet]

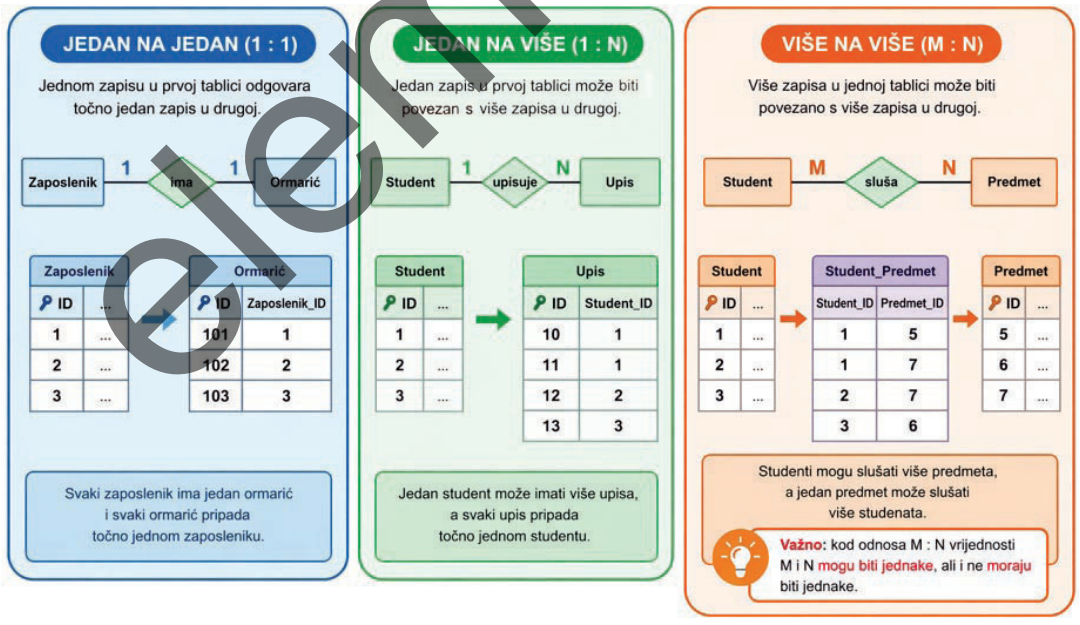


Slika 1.3. Pravilno označavanje relacije

Vrste odnosa:

- **1 : 1** (jedan na jedan) – svaki entitet A može biti povezan s najviše jednim entitetom B, npr. *Zaposlenik – Ormarić*
- **1 : N** (jedan na više) – jedan entitet A može biti povezan s više entiteta B, npr. *Student – Upis*
- **M : N** (više na više) – više entiteta A može biti povezano s više entiteta B, npr. *Student – Predmet*.
  - U bazi podataka se poslije pretvara u **posebnu tablicu**.

Na slici 1.4. prikazani su odnosi između entiteta.



Slika 1.4. Odnosi između entiteta

1.1.4. Brojnost i obveznost odnosa

Brojnost označava **broj mogućih veza** između entiteta. Obično se prikazuje uz linije koje spajaju entitete i relaciju:

- **1** – točno jedan
- **N/M** – više (engl. *many*).

Obveznost:

- **obvezna veza** – mora postojati (puna linija)
- **neobvezna veza** – može, ali ne mora postojati (isprekidana linija ili oznaka "0").

1.1.5. Pravila imenovanja

Pri izradi E-R modela važno je **pridržavati se određenih pravila** kako bi model bio jasan, logičan i jednoznačan. **Nazivi entiteta trebaju biti napisani kao imenice u jednini** (npr. Student, Predmet) čime se jasno naglašava da svaki entitet predstavlja jednu vrstu objekta. Također, **nazivi se pišu bez razmaka i dijakritičkih znakova** kako bi bili jednostavni za korištenje u kasnijim fazama (npr. BrojIndeksa, a ne Broj indeksa).

Odnosi između entiteta označavaju se glagolima u trećem licu jer opisuju radnju ili vezu među njima (npr. pohađa, predaje, naručuje). Svaki **entitet** mora imati **barem jedan ključni atribut** koji ga jednoznačno identificira čime se osigurava razlikovanje pojedinih zapisa.

Relacije trebaju imati smislene i jasno razumljive nazive, a njihova **brojnost (kardinalitet) mora biti točno određena** kako bi se pravilno opisao odnos među entitetima. Osim toga, važno je izbjegavati nepotrebne ili dvosmislene veze jer one mogu otežati razumijevanje modela i dovesti do pogrešaka u kasnijoj izradi baze podataka.

Primjenom ovih pravila dobiva se uredan i logičan E-R model koji predstavlja dobru osnovu za daljnju izradu relacijskog modela i baze podataka.

Izrada E-R dijagrama odvija se postupno, u 7 koraka (slika 1.5.):

1. analiza sustava
2. izdvajanje entiteta
3. određivanje atributa