ALGEBRA
UČILIŠTE

SQL osnove

PRIRUČNIK ZA POLAZNIKE

Tina Kaštelan

info@algebra.hr • www.algebra.hr

autori:
Silvije Davila
Tina Kaštelan

urednica:
Ana Rutar, prof.

naslov:
Osnove SQL-a

stručni recenzent:
Mario Šipek

lektorica:
Ankica Tomić

grafički urednik:
Krešimir Pletikosa, ACE

nakladnik:
Algebra d.o.o., 2010.

za nakladnika:
mr.sc. Mislav Balković

mjesto i godina izdavanja:
Zagreb, 2010

CIP zapis dostupan u računalnom katalogu Nacionalne i sveučilišne knjižnice u Zagrebu pod brojem 748825

ISBN 978-953-322-008-6

Sadržaj:

1. Poglavlje: RELACIJSKI MODEL BAZE PODATAKA

U ovom poglavlju naučiti ćete:

- ☑ Što je baza podataka
- ☑ Sustavi za upravljanje bazama
- ☑ Dizajniranje baze podataka
- ☑ Modeliranje entiteta i veza
- ☑ ER-dijagrami
- ☑ Relacijski model baze podataka
- ☑ Postupak normalizacije

1.1 UVOD

Baze podataka su jedan od važnijih objekata koji se koriste u svakodnevnom radu. Primjerice, svaka tvrtka ima obavezu voditi knjigovodstvene papire, pa podatke o svom poslovanju mora negdje zapisati. Također, svaka tvrtka vodi razne interne evidencije robe na skladištu, u prodavaonici, o kupcima i dobavljačima..., te podatke mora negdje zapisati. Svaka banka, videoteka, trgovina, bolnica, hotel, također, moraju zapisivati veliku količinu podataka o svom poslovanju. Na internetu se većina stranica oslanja na podatke koji su negdje pohranjeni.

Prije su se ti podaci pohranjivali pismeno, u knjige. Nakon toga su došla računala, pa su se podaci počeli pohranjivati u datoteke koje su se spremale na diskete, trake ili na disk računala. Kako su potrebe za organizacijom i lakšim dohvatom podataka rasle, tako se s običnih datoteka prešlo na **bazu podataka**, koja onda pomaže da se podaci lakše obrade.

Naravno, kako je baza podataka pohranjena na računalu u nekom digitalnom formatu, tako je *izmišljen* jezik pomoću kojeg se s tom bazom podataka *razgovara*, tako da baza može odgovarati na postavljena pitanja. Taj *izmišljeni* jezik danas se naziva SQL (čitati: *es kju el*) i on je tema ovog priručnika.

1.2 BAZE PODATAKA

1.2.1 DIREKTNE DATOTEKE

Kao što smo spomenuli u uvodu, nekad su se podaci pohranjivali u datoteke koje možemo promatrati kao kolekciju zapisa, gdje se svaki zapis sastoji od polja. Svako polje pripada nekom tipu podataka (broj, slovo, ...). Takve se datoteke nazivaju *linearne* ili *direktne datoteke* (engl. *flat file*) i kod njih nema nikakve dodatne organizacije zapisa. Jedan od glavnih problema kod takve organizacije jest da zapisivanje na fizički medij (disketa, traka, tvrdi disk, ...) ovisi o samom mediju. Ostali uočeni problemi su:

- **konkurentnost** – znači da podacima u jednom trenutku može pristupati najviše jedna osoba, tj. najviše jedan program
- **integritet** – ako više programa koristi iste informacije, može doći do iskrivljavanja podataka jer ne postoji nikakva kontrola
- **relacije** – teško je uspostaviti bilo kakve relacije među podacima jer ne postoji nikakva predefinirana struktura, pa je podacima teško pristupati
- **ponovna iskoristivost** – direktna datoteka dizajnirana za jedan sustav teško se može iskoristiti na nekom drugom
- **sigurnost** – ne postoji univerzalan način osiguravanja pristupa podacima (npr. pomoću šifre), pa su podaci podložni neautoriziranom pristupanju

1.2.2 SUSTAV ZA UPRAVLJANJE BAZOM PODATAKA – DBMS

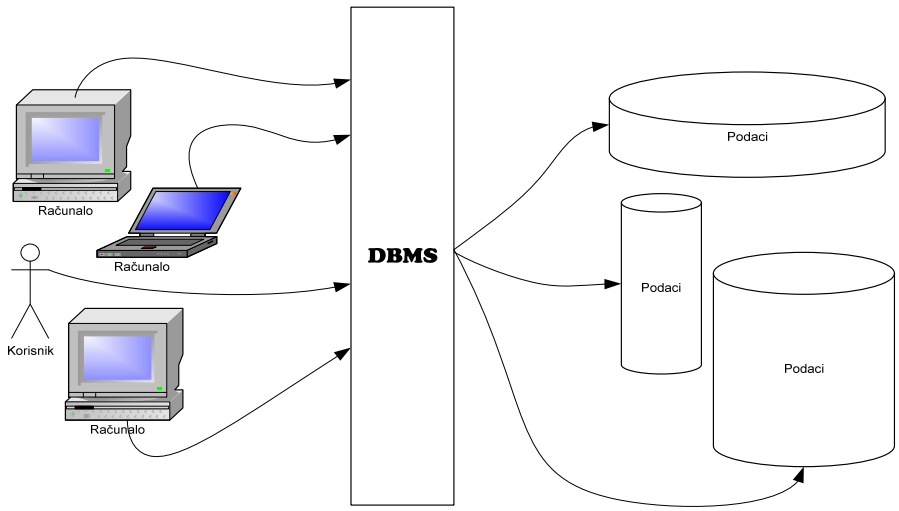
Baza podataka je skup međusobno povezanih podataka pohranjenih na vanjskoj memoriji, koji su istovremeno dostupni raznim korisnicima i programima.

Podaci koji se nalaze u nekoj bazi fizički su pohranjeni kao datoteke, pa se postavlja pitanje po čemu se baza podataka razlikuje od direktne datoteke? Naravno, od same baze podataka nemamo puno više koristi nego što imamo od direktnih datoteka. Ono od čega imamo velike koristi jest sustav za upravljanje bazom podataka ili skraćeno DBMS (**D**ata**B**ase **M**anagement **S**ystem).

DBMS je poslužitelj baze podataka koji se nalazi između korisnika baze podataka i same baze. On u ime korisnika obavlja sve operacije s podacima.

Kako rad s bazom podataka ne znači samo dohvaćanje podataka nego i njihovo dodavanje, mijenjanje i brisanje, te čuvanje integriteta podataka, čuvanje sigurnosnih kopija baze, omogućavanje istovremenog pristupa bazi više korisnika, provjeru identiteta korisnika, postaje jasno da je za to potreban DBMS.

Ključna stvar koja razlikuje bazu podataka od direktne datoteke leži u svojstvu nezavisnosti podataka koju pruža DBMS.



Postoje dva nivoa nezavisnosti:

- **Fizička nezavisnost** – baza može promjeniti medij na kojemu zapisuje svoje podatke, a korisnik baze tu promjenu ne vidi. Baza se čak može distribuirati na više računala, a korisnik će vidjeti bazu kao jedinstvenu cjelinu.
- **Logička nezavisnost** – označuje mogućnost promjene logičke sheme baze bez potrebe da korisnik išta mijenja u svom načinu pristupanja bazi.

Podaci u bazi logički su organizirani u skladu s nekim modelom podataka. **Model podataka** je skup pravila koja određuju kako može izgledati logička struktura baze.

Dosadašnji DBMS-i obično su podržavali neki od sljedećih modela:

- **Relacijski model**
  Zasnovan je na matematičkom pojmu relacije. I podaci i veze medu podacima prikazuju se tablicama. Tablice su međusobno povezane relacijama čija je uloga izbjeći redundanciju (ponavljanje), očuvati integritet podataka, te povećati brzinu pretraživanja.
- **Mrežni model**
  Baza je predočena usmjerenim grafom. Čvorovi su tipovi zapisa, a lukovi definiraju veze među njima.
- **Hijerarhijski model**
  To je specijalni slučaj mrežnoga modela. Baza je predočena jednim stablom ili skupom stabala. Čvorovi su tipovi zapisa, a hijerarhijski odnos "nadređeni-podređeni" izražava veze među njima.
- **Objektni model**
  Inspiriran je objektno-orijentiranim programskim jezicima. Baza je skup trajno pohranjenih objekata koji se sastoje od svojih internih podataka i metoda za rukovanje tim podacima. Svaki objekt pripada nekoj klasi. Izmedu klasa uspostavljaju se veze nasljeđivanja, agregacije, odnosno međusobnog korištenja operacija.

Od 80-tih godina pa sve do danas prevladava relacijski model, a očekivani prijelaz na objektni model još se nije dogodio. Što se tiče objektnog modela, on je dosta moćniji od ostalih, ali je zato i kompliciraniji, pa još uvijek nekako prevladava jednostavnost relacijskoga modela.

1.3 DIZAJNIRANJE BAZE PODATAKA

Prije nego što se upustimo u kreiranje i rad s bazama podataka, dobro je proći barem neku fazu dizajniranja. Naravno, kada se radi s nekom manjom bazom, mnogo se toga može dizajnirati '*napamet'*, dovoljna će biti i skica na papiru. No, čim se krene raditi sa većim bazama, treba uložiti veći napor da bi se krajnja baza dobro napravila.

Uvođenje baze podataka u neko poduzeće ili ustanovu predstavlja složeni zadatak koji zahtijeva timski rad stručnjaka raznih profila.

Taj se projekt koji se može podijeliti u pet faza:

- analiza potreba
- modeliranje podataka
- implementacija
- testiranje
- održavanje

1.3.1 ANALIZA POTREBA

Da bismo razvili bazu za neki sustav moramo prikupiti informacije o tom sustavu, bilo kroz proučavanje samog sustava bilo komunikacijom sa stručnjacima koji se njime služe. Tako se sustav i podaci koje je potrebno zapamtiti u bazi detaljno upoznaju, kao i način na koji se ti podaci prikupljaju. Rezultat ovog koraka treba biti početna dokumentacija sustava koja će služiti da bi se izveli daljnji koraci. Rezultat analize je dokument (pisan neformalno u prirodnom jeziku) koji se zove **specifikacija potreba**. Kada imamo takvu dokumentaciju krećemo na konceptualni dizajn. To znači da trebamo detaljno raščlaniti podatke koji čine bazu služeći se modelom visokog nivoa (koriste se tzv. ER dijagrami).

1.3.2 MODELIRANJE PODATAKA

U ovom koraku identificiramo pripadnike (entitete) koji čine sustav i shvaćamo način na koji su povezani (tj. točno utvrđujemo njihovu međusobnu relaciju). Sljedeći je korak logički dizajn u kojem iz modela visokoga nivoa izvodimo model za implementaciju, tj. iz ER dijagrama kreiramo stvarnu bazu.

1.3.3 IMPLEMENTACIJA

Na temelju sheme i uz pomoć dostupnog DBMS-a, fizički se kreira baza podataka na računalu. U DBMS-u postoje parametri kojima se može utjecati na fizičku organizaciju baze, a oni se podešavaju tako da se osigura efikasan rad najvažnijih transakcija. Baza se puni podacima.

U fizičkom dizajnu, odlučujemo (točnije, administrator baze odlučuje) kako će baza biti zapisana (na disku, traci, na više diskova...) i na koje će se sve to načine događati. Te se odluke obično donose nakon razmišljanja o performansama i dostupnosti hardvera.

1.3.4 TESTIRANJE

Korisnici testiraju rad s bazom i provjeravaju zadovoljava li ona svim zahtjevima. Nastoje se otkriti greške koje su se mogle potkrasti u svakoj od faza razvoja. Greške u ranijim fazama imaju teže posljedice. Na primjer, greška u analizi potreba dovodi do toga da transakcije možda korektno rade, ali ne i ono što korisnicima treba. Dobro bi bilo kad bi se takvi propusti otkrili prije implementacije. Zato se u novije vrijeme, prije prave implementacije, razvijaju i približni prototipovi baze podataka, koji se pokazuju korisnicima.

1.3.5 ODRŽAVANJE

Održavanje se odvija nakon što baza već uđe u redovnu upotrebu. Sastoji se od popravaka grešaka koje nisu bile otkrivene u fazi testiranja, te od uvođenja promjena zbog novih zahtjeva korisnika i podešavanja parametara u DBMS-u u svrhu poboljšavanja performansi.

1.4 MODELIRANJE PODATAKA

1.4.1 MODELIRANJE ENTITETA I VEZA

Entitet je objekt, pojava ili događaj o kojem želimo spremati podatke. To je nešto što se može identificirati, npr. student, kupac, proizvođač, polaznik seminara, servisiranje auta...

Svaki entitet ima svoje karakteristike koje ga jednoznačno ili jedinstveno opisuju, a nazivaju se **atributi**. Na primjer, atributi kuće su kućni_broj, visina, širina, vrijednost ... Za svaki entitet stvara se tablica ili **relacija**.

Veza je nešto što veže dva ili više entiteta. Razlikujemo tri vrste binarnih veza i to zovemo funkcionalnost veze:

- 1 –1 veza
  jednom zapisu iz jedne tablice odgovara jedan i samo jedan zapis iz druge tablice;
  npr. veza JE_U_BRAKU_SA između entiteta MUŠKARAC i ŽENA
- 1 – n veza (ili 1:∞)
  jednom zapisu iz jedne tablice odgovara 0, 1 ili više zapisa iz druge;
  npr. veza PREDAJE između entiteta PROFESOR i KOLEGIJ
  (jedan profesor može predavati više kolegija na fakultetu)
- m – n veza (ili ∞:∞)
  jedan zapis prvog entiteta može biti u vezi s 0, 1 ili više primjeraka drugog entiteta, te također jedan zapis drugog entiteta može biti u vezi s 0, 1 ili više zapisa prvog entiteta;
  npr. veza UPISALA između entiteta OSOBA i SEMINAR

Ako svaki zapis nekog entiteta mora sudjelovati u određenoj vezi, onda kažemo da taj entitet ima obavezno članstvo u toj vezi. Inače, entitet ima neobavezno članstvo. Na primjer, između entiteta OSOBA i GRAD zadana je veza ŽIVI_U. OSOBA u vezi ŽIVI_U ima obavezno članstvo jer svaka osoba mora negdje stanovati.

Rezultat ovog koraka je popisivanje svih veza i entiteta te njihovih atributa.

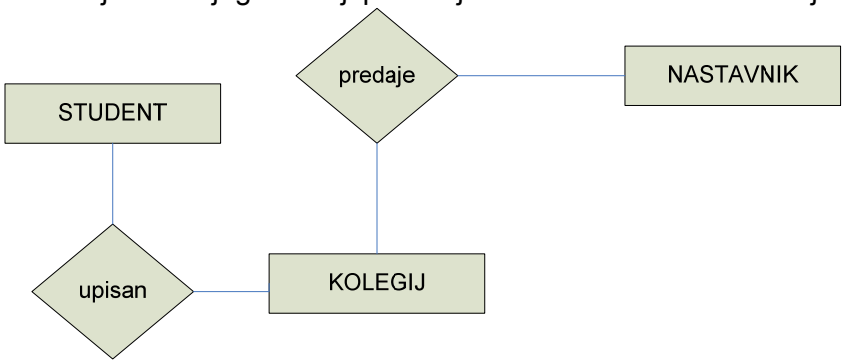
1.4.2 ER – DIJAGRAMI

ER dijagram je dijagram koji pokazuje relacije između entiteta u sustavu. Kratica 'ER dijagram' dolazi od engleskih riječi ***E**ntity **R**elationship Diagram*.

ER model dovoljno je jednostavan da ga ljudi različitih struka mogu razumjeti. Zato ER shema služi za komunikaciju projektanta baze podataka i korisnika, i to u najranijoj fazi razvoja baze. Postojeći DBMS ne mogu direktno implementirati ER shemu, već zahtijevaju da se ona detaljnije razradi.

Na ER dijagramu entiteti se prikazuju pravokutnicima, a veze među njima rombovima koji su s entitetima povezani bridovima. Imena entiteta i veza, te funkcionalnost veza, uneseni su u dijagram. Relacije među entitetima mogu biti unarne, binarne ili n-arne. Ako je relacija unarna to znači da je broj entiteta koji sudjeluju u relaciji jedan. U binarnoj relaciji sudjeluju dva, a u n-arnoj više od dva entiteta.

Pogledajmo primjer. Pokušavamo modelirati jedan dio fakultetskih odnosa, tj. na fakultetu postoje nastavnici koji predaju neke kolegije te studenti koji ih slušaju. ER dijagram koji prikazuje te odnose nalazi se na sljedećoj slici.



Student je upisao kolegij koji predaje nastavnik, npr. studentica Kristina Naglić je upisala kolegij „Baze podataka“ koji predaje nastavnik Marko Jurić.

Entiteti su:

1. *STUDENT* s atributima BR INDEKSA, IME STUDENTA, ADRESA, SPOL ...
2. *KOLEGIJ* s atributima BR KOLEGIJA, NASLOV, SEMESTAR ...
3. *NASTAVNIK*, s atributima IME NASTAVNIKA, BR KABINETA ...

Veze su:

1. *UPISAN* s atributom DATUM UPISA.
2. *PREDAJE* bez atributa. KOLEGIJ ima obavezno članstvo.

Kreiranje ER dijagrama je iterativan proces, što znači da kako bismo došli do konačne verzije, moramo nacrtati više „probnih“ verzija, tj. više puta probati dok ne dobijemo odgovarajući zapis. Kada se skupi iskustva, broj probnih dijagrama se smanjuje. Nakon što dobijemo ER dijagram, pristupit ćemo izradi same baze. Baza se iz ER dijagrama dobiva pretvorbom dijagrama u tablice.

1.5 RELACIJSKI MODEL

Godine 1970. dr. Edgar F. Codd objavio je rad u kojem govori o „relacijskom modelu podataka za velike, dijeljene *banke* podataka“. Taj je model danas široko prihvaćen kao model za **relacijsku bazu podataka**. Tijekom nekoliko slijedećih godina u IBM-ovu istraživačkom centru razvijen je sustav pod nazivom *System R* koji se temeljio na tom modelu.

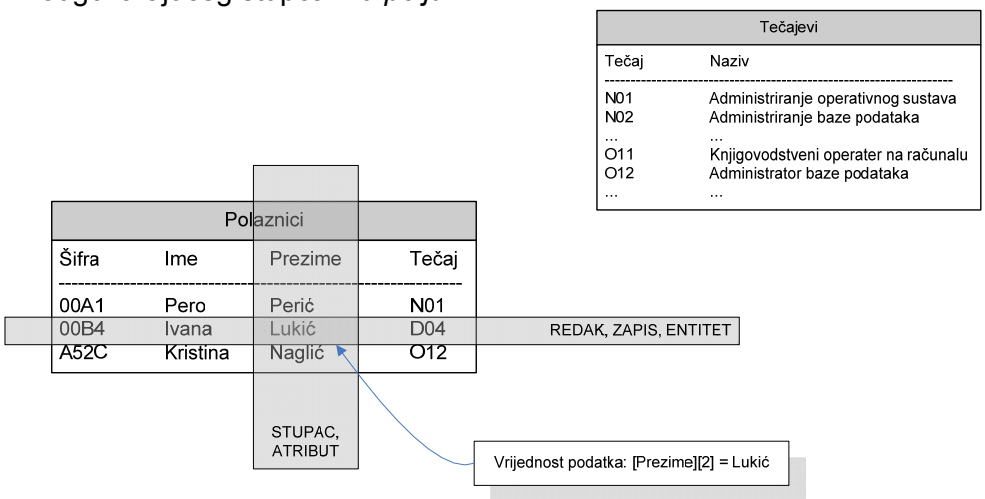
Što znači da je DBMS relacijski?

Relacijski model baze podataka zasnovan je na ideji da se cjelokupni skup podataka koji želimo da baza prati razdijeli na pravokutne tablice tzv. **relacije**.

Svaka relacija ima svoje ime po kojem je razlikujemo od ostalih u istoj bazi. Jedan stupac relacije obično sadrži vrijednost jednog atributa (za entitet ili vezu), pa zato stupac poistovjećujemo s **atributom**. Atribut ima svoje ime po kojem ga razlikujemo od ostalih u istoj relaciji. Vrijednosti jednog atributa čine podaci istog tipa. Dakle, definiran je skup dopuštenih vrijednosti za atribut, koji se zove **domena** atributa. Vrijednost atributa mora biti jednostruka i jednostavna (ne da se rastaviti na dijelove). Vrijednost atributa nije uvijek obvezna za unos. Jedan redak relacije predstavlja jedan zapis entiteta.

Neka svojstva tablice u relacijskom modelu su:

- tablica je skup zapisa u bazi podataka
- tablica sadrži više *stupaca*
- svaki stupac ima jedinstven naziv
- svaki *redak* predstavlja jedan *zapis* nekog entiteta
- vrijednost nekog podatka dobije se na presjeku odgovarajućeg retka i odgovarajućeg stupca – u *polju*



Danas je činjenica da se RDBMS koristi u većini komercijalnih sustava. Preostaje još pitanje zašto je RDBMS bolji od ostalih modela? Prvo, pokazalo se da su i mrežni i hijerarhijski modeli usko vezani uz implementaciju baze, a relacijski model nije, pa je relacijski model popularniji izbor (kada je nešto usko vezano uz implementaciju, veliki problem postaju naknadne izmjene u strukturi, što uvelike otežava održavanje i proširivanje baze). Drugo, i mrežni i hijerarhijski model uspostavljaju vezu između zapisa pomoću dodatnih struktura (pokazivači ili

poveznice), a relacijski uspostavlja vezu među zapisima na temelju vrijednosti koje ti zapisi sadrže, npr. tablice *Polaznici* i *Tečajevi* povezane su preko vrijednosti stupca *Tečaj*. Pa ako nas zanima koji je naziv tečaja koji pohađa *Kristina Naglić*, trebamo pogledati odgovarajući stupac *Tečaj* u tablici *Polaznici*, zapamtiti šifru (*O12*), te u tablici *Tečajevi* potražiti redak sa tom šifrom. Dobit ćemo da je *Kristina* upisala tečaj *Administrator baze podataka.*

Glavna prednost relacijskoga modela u tome je što vrlo efikasno rješava dva velika problema koja se javljaju kod pohrane podataka: redundanciju (ponavljanje) i nekonzistentnost podataka.

Polaznici i tečajevi				
Šifra polaznika	**Ime polaznika**	**Mjesto stanovanja**	**Šifra tečaja**	**Naziv tečaja**
1	Ana Milić	Zagreb	P01	Osnove rada PC računala
2	Sanja Tarak	Split	P02	Microsoft Word
3	Mladen Gork	Osijek	D01	SQL - osnove
4	Ivana Matkić	Split	O01	Računalni operator – uredsko poslovanje
4	Ivana Matkić	Split	D05	Osnove i teorija C++
5	Marina Anić	Osijek	O02	Specijalist poslovne primjene računala
6	Ivica Limac	Split	P01	Osnove rada PC računala

U ovom primjeru imamo duple podatke o mjestu stanovanja Ivane Matkić i dva puta zapisanu njezinu šifru, ime i prezime. Zamislimo da sada svatko od npr. 10000 polaznika upiše po nekoliko tečajeva, koliku bismo hrpu nepotrebno dupliciranih podataka dobili! Ali, nije to sve: što ako se Ivana preseli iz Splita u Varaždin? Onda moramo locirati sva pojavljivanja Ivane Matkić u tablici (koja sada ima više od npr. 40.000 redaka), te promijeniti Split u Varaždin. Zar ne bi bilo bolje da smo morali tu informaciju promijeniti samo na jednome mjestu?

Kod takvih i sličnih problema pomaže nam postupak koji nazivamo **normalizacijom**.

Sljedeći korak u stvaranju relacijskog modela jest da u svakoj tablici definiramo polje primarnog ključa.

1.5.1 PRIMARNI KLJUČ

Primarni ključ je podatak (ili više podataka) koji razlikuje jedan zapis od drugoga. Po njemu gledajući, svi su zapisi različiti. Dakle, primarni ključ je vrijednost stupca koja na jedinstven način određuje neki redak.

Pretpostavimo da u sustavu imamo dva polaznika tečajeva istoga imena i prezimena. Kako ćemo znati koji od ta dva polaznika ide na seminar SQL – osnove, a koji na Osnove rada PC računala? Kako ih razlikovati?

Rješenje je u definiranju primarnoga ključa.

Primjerice u našoj tablici polaznika stupac *Šifra polaznik*' na jedinstven način određuje neki redak (ime i prezime se može ponoviti, mjesto stanovanja također, pa nema govora o jedinstvenosti).

Primarni se ključ javlja u praksi – na primjer JMBG, ISBN knjige, poštanski broj, bar_kod proizvoda ...

Polje primarni ključ čini jedinstvenim svaki zapis u tablici i time ih međusobno distancira. U relacijskim je bazama obavezno definiranje primarnoga ključa za svaku tablicu.

Tečajevi	
Šifra tečaja	**Naziv tečaja**
P01	Osnove rada PC računala
P02	Microsoft Word
D01	SQL - osnove
O01	Računalni operator – uredsko poslovanje
O02	Specijalist poslovne primjene računala
O03	Grafički dizajner

Primarni ključ može sadržavati jedno ili više polja. Ako se sastoji od više polja dopušta se ponavljanje nekih, ali ne i svih polja koja čine primarni ključ.

Napomena: Polja primarnih ključeva u tablicama su obično masnije (Bold) otisnuta.

1.5.2 PRETVORBA ENTITETA I VEZA U RELACIJE

Pretvorba entiteta vrši se tako da svaki atribut ima svoj stupac u tablici, a u retke upisujemo primjerke tog entiteta i nazivamo ih **zapisima** (**records**). Tako dakle, pretvaramo entitet u relaciju, tj. tablicu.

Pretvorba veza je malo kompleksnija:

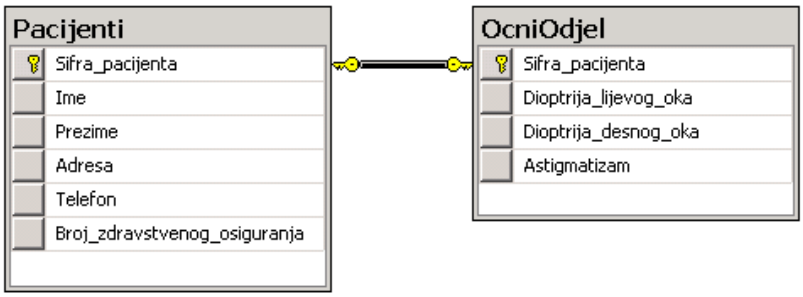
Ako u vezi 1 – N postoji obavezno članstvo, onda entitetu na strani N dodajemo jedan atribut i to primarni ključ entiteta na strani 1. Taj atribut nazivamo **stranim ključem**.

Ako u vezi 1 – N ne postoji obavezno članstvo, veza postaje nova tablica s atributima. Primarni ključ jednog i drugog entiteta (tj. ta dva atributa zajedno) tvori primarni ključ nove tablice.

Veza M – N uvijek postaje nova tablica. Naime, ona se ne može direktno ostvariti, pa se razbija u dvije 1 – N veze.

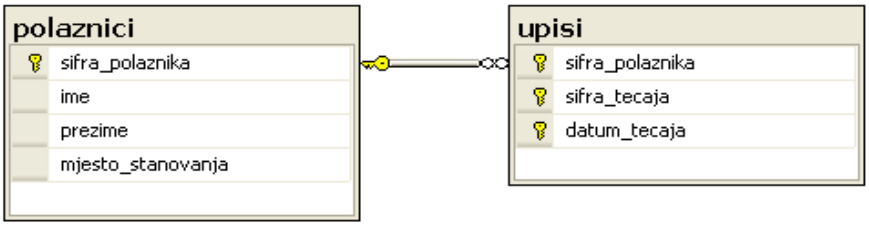
PRIMJER VEZE 1-1:

Neka se u jednoj tablici nalaze imena i prezimena pacijenata s osobnim podacima i adresama, a u drugoj tablici podaci o dijagnozi na očnom odjelu. Te dvije tablice povezane su relacijom 1:1. Jednom pacijentu odgovara samo jedna dioptrija, jer nije moguće da jedan pacijent ima dvije različite dioptrije.



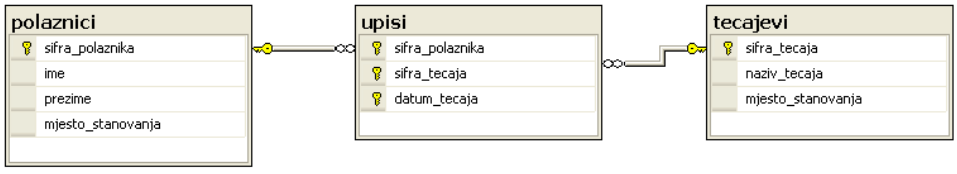
PRIMJER VEZE 1-N:

U tablici *Polaznici* Ana Milić će se, primjerice, pojaviti samo jednom. Njena šifra broj 1 u toj će se tablici pojaviti samo jednom. No, Ana može upisati puno različitih tečajeva, pa će se u tablici *Upisi* njezina šifra pojaviti više puta.



PRIMJER VEZE M-N:

U ovom bismo primjeru htjeli direktno spojiti tablice *Polaznici* i *Tecajevi*. Naime, jedan polaznik iz tablice *Polaznici* može se prijaviti na 0,1,2,... n tečajeva iz tablice *Tecajevi*. U isto vrijeme na samo jedan tečaj upisano je 0,1,2,... m polaznika. Ako bismo htjeli direktno povezati tablice uvodeći jedno novo polje koje je strani ključ u tablicu *Tecajevi* došlo bi do problema redundancije (ponavljanja) podataka o tečajevima. Kako bismo izbjegli taj problem uvodimo novu tablicu *Upisi* koja čuva podatke o upisu polaznika na tečaj.



1.6 NORMALIZACIJA

Normalizacija je proces organiziranja podataka s ciljem minimalnog dupliciranja podataka, tj. proces kreiranja efikasne, pouzdane i fleksibilne baze podataka. Taj se postupak temelji na matematički dokazanim tvrdnjama, što znači da ćemo, budemo li ga slijedili, kao rezultat *sigurno* dobiti *dobru* bazu podataka (naravno, kao i sve drugo, i ovaj se postupak može *loše provesti* i dovesti do loših rezultata, no to se rijetko događa).

Kako tablicu sa slike 4 možemo dovesti u dobro stanje, tj. normalizirali? Dovoljno je prethodnu tablicu *razbiti* na tri nove. Jednu koja bi popisala polaznike, drugu koja bi popisala kolegije i treću, *relacijsku tablicu*, koja bi povezala prve dvije.

Polaznici			
Šifra polaznika	**Ime polaznika**	**Prezime polaznika**	**Mjesto stanovanja**
1	Ana	Milić	Zagreb
2	Sanja	Tarak	Split
3	Mladen	Gork	Osijek
4	Ivana	Matkić	Split
5	Marina	Anić	Osijek
6	Ivica	Limac	Split

Tečajevi	
Šifra tečaja	**Naziv tečaja**
P01	Osnove rada PC računala
P02	Microsoft Word
N01	SQL – osnove
O01	Računalni operator – uredsko poslovanje
O02	Specijalist poslovne primjene računala
O03	Grafički dizajner

Upisi	
Šifra polaznika	**Šifra tečaja**
1	P01
2	P02
3	N01
4	O01
5	O02
6	P01

Ako Ivana želi upisati još jedan tečaj, tu ćemo informaciju jednostavno dodati kao novi redak u tablicu UPISI, a ostale tablice nećemo dirati. Ako se Ivana želi preseliti u Varaždin, tu ćemo informaciju promijeniti samo na jednome mjestu, u tablici POLAZNICI. Jasno je da smo time dobili puno, pogotovo u pogledu fleksibilnosti, integriteta podataka i efikasnosti.

Do ovakvih rezultata dolazi se konzistentnom primjenom pravila koja se nazivaju *normalne forme*.

Postoji 6 normalnih formi:

1. Prva normalna forma (1NF)
2. Druga normalna forma (2NF)
3. Treća normalna forma (3NF)
4. Boyce-Coddova normalna forma (BCNF)
5. Četvrta normalna forma (4NF)
6. Peta normalna forma (5NF)

1NF zahtijeva da svaka vrijednost u stupcu koja se pojavljuje bude atomarna i da svi reci imaju isti broj polja. 1NF zapravo govori da se kao jedna vrijednost ne smije pojaviti skup podataka. Pravila prve normalne forme krše se najčešće tako da, primjerice, imate u jednom polju i ime i prezime polaznika, što komplicira stvari već prilikom sortiranja zapisa po npr. prezimenima. Rijetko kada se dogodi da tablica nije u 1NF, jer nam nekako iskustvo i intuicija odmah daju „dobru stvar“.

2NF zahtijeva da bude zadovoljena 1NF, te da svi reci budu jednoznačno određeni i potpuno ovisni o primarnome ključu. Primjerice u prvoj tablici stupci *Šifra tečaja* i *Naziv tečaja* nemaju veze s primarnim ključem *Šifra polaznika*. Naravno, tečajevi postoje neovisno o polaznicima, postoji samo veza među njima (i to tek kada polaznik upiše neki tečaj).

3NF zahtijeva da bude zadovoljena 2NF, te da ne postoje tzv. *tranzitivne ovisnosti* o primarnome ključu, tj. svi stupci koji nisu primarni ključ moraju biti neovisni jedni o drugima. To znači da se ne smije dogoditi da neki podatak ovisi o nekome drugom, koji pak ovisi o primarnome ključu. To se nama dogodilo u prvoj tablici: *ime tečaja* ovisi o *šifri tečaja* koja pak ovisi o *šifri polaznika*. Najočitije kršenje treće normalne forme predstavljaju izračunata polja – problem ažuriranja podataka. Primjerice, u jednoj tablici postoje polja *Cijena* i *Količina*, a mi želimo pomnožiti te podatke i umnožak sačuvati u tablici. Problem se javlja ako se promijene *Cijena* ili *Količina* za neki proizvod jer se polje *Ukupno* automatski ne ažurira.

BCNF otprilike zahtijeva da, ako postoji stupac o kojem je neki drugi ovisan, onda on mora biti ključ u tablici. Ako je neka relacija u BCNF, onda je i u 3NF (naravno, i u 2NF i u 1NF), ali postoje situacije kada je relacija u 3NF, a nije u BCNF.

4NF i 5NF se u praksi rijetko koriste, čak se i BCNF ne koristi tako često jer se u većini situacija možemo „snaći“ i problem riješiti drukčije. Naravno, idealno bi bilo dovesti bazu u 5NF, ali je s praktičnog stajališta to često nepotrebno. Naime, razbijanjem baze na mnogo malih tablica usporava se čitanje s medija, što nekad može biti lošije rješenje od nenormalizirane tablice. Osoba koja implementira bazu trebala bi isprobati i na temelju iskustva zaključiti dokle treba provesti normalizaciju.

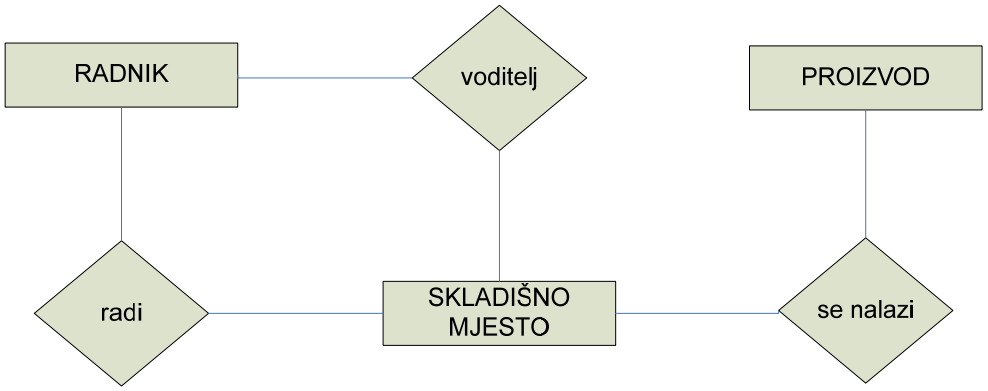
1.7 ZADACI ZA VJEŽBU

ZADATAK 1.1

Neka tvrtka sadrži nekoliko odjela u kojima zapošljava radnike. Svaki odjel ima šefa koje je ujedno i radnik poduzeća. Kreirajte ER dijagram, te ga pretvorite u tablicu i na kraju normalizirajte.

ZADATAK 1.2

Trgovina ima nekoliko skladišnih mjesta po kojima grupira svoje proizvode. Npr. cigle i crijepovi idu na jedno skladišno mjesto, drvena građa (daske, grede ...) na drugo skladišno mjesto. Svako skladišno mjesto ima nekoliko radnika koji na njemu rade i svako ima svog voditelja. Na temelju danog ER dijagrama kreirajte normalizirane tablice.



ZADATAK 1.3

Kreirajte ER dijagram, a zatim i normalizirane tablice potrebne za poslovanje jedne videoteke. Videoteka članovima izdaje članske iskaznice, te se na temelju članskog broja osoba identificira kako bi mogla posuditi filmove. Filmovi su po policama složeni po pripadajućim žanrovima. Videoteka ima definiran cjenik za izdavanje hit filma, filma koji nije hit te starog filma. Jedan film može biti na DVD-u i na VHS-u. Film se posuđuje na rok od jednog dana i ako ga član ne vrati u navedeno vrijeme, zaračunava mu se zakasnina.