# Baze podataka

Prof. dr. sc. Damir Medak

Geodetski fakultet Sveučilišta u Zagrebu

Zagreb, listopad 2008.

# Sadržaj kolegija

- Cilj uvodnog predavanja: kratak pregled sadržaja kolegija "Baze podataka"
- Sadržaji svih kolegija na Geodetskom fakultetu dostupni su na internetu: `http://www.geof.hr/`

# Cilj kolegija

- ovladati temeljima suvremenih baza podataka s posebnim naglaskom na prostorne (geo-) podatke
- usvojiti osnove samostalnog rukovanja komercijalnim i slobodnim programskim paketima za baze podataka (relacijski model, ER-model)
- steći osnovna znanja o konceptima baza podataka koje će postati industrijski standard u vrijeme početka profesionalne karijere (objektno-relacijske baze podataka)

# Temeljni pojmovi i definicije

- Pojam baze podataka, informacijskog sustava, informacije, podatka, modela podataka
- Dizajn baza podataka:
  - konceptualni
  - implementacijski
  - fizički dizajn
- Fizička organizacija podataka

# Relacijske baze podataka

- Osnovni elementi: tablice i relacije, atributi i ključevi
- Osnove relacijske algebre: selekcija, projekcija, produkt, povezivanje (*join*), razlika, unija, presjek
- Upitni jezici (SQL)
- ER-dijagram
- Normalne forme relacijskih baza podataka
- Primjene u geodeziji i geoinformatici

# Sustav za upravljanje bazom podataka (DBMS)

- eng. DataBase Management System
- Integritet podataka
- Transakcije
- ACID uvjeti:
  - atomičnost (eng. *atomicity*)
  - konzistentnost (eng. *consistency*)
  - izolacija (eng. *isolation*)
  - trajnost (eng. *durability*)
- Višekorisničke baze podataka

# Uvod u objektno-orijentirani dizajn

- Osnove OO programskih jezika i baza podataka:
  - nasljeđivanje (eng. *inheritance*)
  - višeobličnost (eng. *polymorphysm*)
  - prikrivanje unutarnjeg ustroja (eng. *encapsulation*)
- Primjene u geodeziji i geoinformatici (kompleksnost prostornih podataka i procesa)
- Uvod u UML (Unified Modeling Language): dijagram geometrijski klasa, nasljeđivanje

# Budućnost baza podataka

- Objektno-relacijski model
- Objektno-orijentirane baze podataka
- Deduktivne baze podataka (Prolog)

# Program vježbi

- Računalne vježbe (podjela u 8 grupa)
- Relacijske baze podataka, SQL (PostgreSQL, MySQL, MSSQL)
- Fakultativno: Deduktivne baze podataka, predikatna logika (Prolog)
- Pohađanje vježbi obavezno!
- Asistenti: Ivan Medved, Almin Đapo, Mario Miler i Branko Kordić
- Suradnici: Dražen Odobašić i Vedran Tatarević

# Obaveze, ispit, ocjena

- Vježbe obavezne: šest zadataka i jedan projekt uvjet za potpis
- Međuispit (30 bodova): 10 pitanja, 3 boda za točan, -1 za netočan odgovor, 0 za neodgovoreno)
- Pismeni ispit (30 bodova): 10 pitanja, 3 boda za točan, -1 za netočan odgovor, 0 za neodgovoreno)
- Dva međuispita: više od 40 bodova - ocjena vrlo dobar ili odličan

# Kad ne bi bilo baza podataka...

- Redundancija i inkonzistentnost
- Ograničena mogućnost pristupa podacima
- Problemi pri višekorisničkoj upotrebi podataka
- Gubitak podataka
- Povreda integriteta podataka
- Sigurnosni problemi
- Visoki troškovi razvoja aplikacija

# Baza podataka i informacijski sustav

- **Baza podataka** je centralno mjesto informacijskog sustava. Pohranjeni podaci u bazi podataka opisuju trenutačno stanje dijela realnog svijeta za koji je i razvijen informacijski sustav, naravno na način pogodan za računalnu obradu.
- **Informacijski sustav** je uvijek podsustav nekog organizacijskog sustava, a svrha mu je prikupljanje, obrada, pohranjivanje i distribucija informacija, koje su potrebne za praćenje rada i upravljanje tim organizacijskim sustavom ili nekim njegovim podsustavom.

# Informacija i podatak

- **Informacija** je znanje koje primatelju opisuje nove činjenice. To znanje se materijalizira u obliku **podataka**, simbola koji služe za prikaz informacija u svrhu spremanja, prijenosa i obrade.
- **Informacija** je i obrađeni podatak koji za primatelja ima karakter novosti, otklanja neizvjesnost i služi kao podloga za odlučivanje.
- **Podatak** je skup znakova u memoriji koji prikazuje jedan ili više elemenata informacije.

# Baza podataka - definicija

- **Baza podataka** je skup međusobno povezanih podataka pohranjenih bez nepotrebne zalihosti s ciljem da na optimalni način posluže u raznim primjenama. Podaci se spremaju neovisno o programima koji ih koriste, zajedničkim pristupom dodaju se novi podaci te mijenjaju i premještaju postojeći.
- Podaci se pohranjuju u bazu podataka na jedan organizirani način koristeći odgovarajući **model podataka**.

# Model podataka - definicija

- **Model podataka** je **formalni sustav** koji mora imati barem sljedeće tri komponente:
  1. Skup **objekata** koji su osnovni elementi baze podataka;
  2. Skup **operacija** koje možemo izvoditi nad objektima definiranim pod 1) i kojima se mogu pretraživati, dobivati i modificirati podaci o tim objektima;
  3. Skup **općih pravila integriteta** podataka koja implicitno ili eksplicitno definiraju skup konzistentnih stanja podataka ili promjena stanja, ili oboje i koja su općenita u smislu da su primjenjiva na bilo koju bazu podataka koja koristi taj model.

# Model podataka se koristi za:

- razvijanje sustava za upravljanje bazom podataka,
- razvijanje programskih jezika za rad s podacima u bazi podataka,
- razvijanje opcih teorija oblikovanja baza podataka,
- istraživanje svojstava podataka, odnosno svojstava baza podataka.

# Sustav za upravljanje bazom podataka

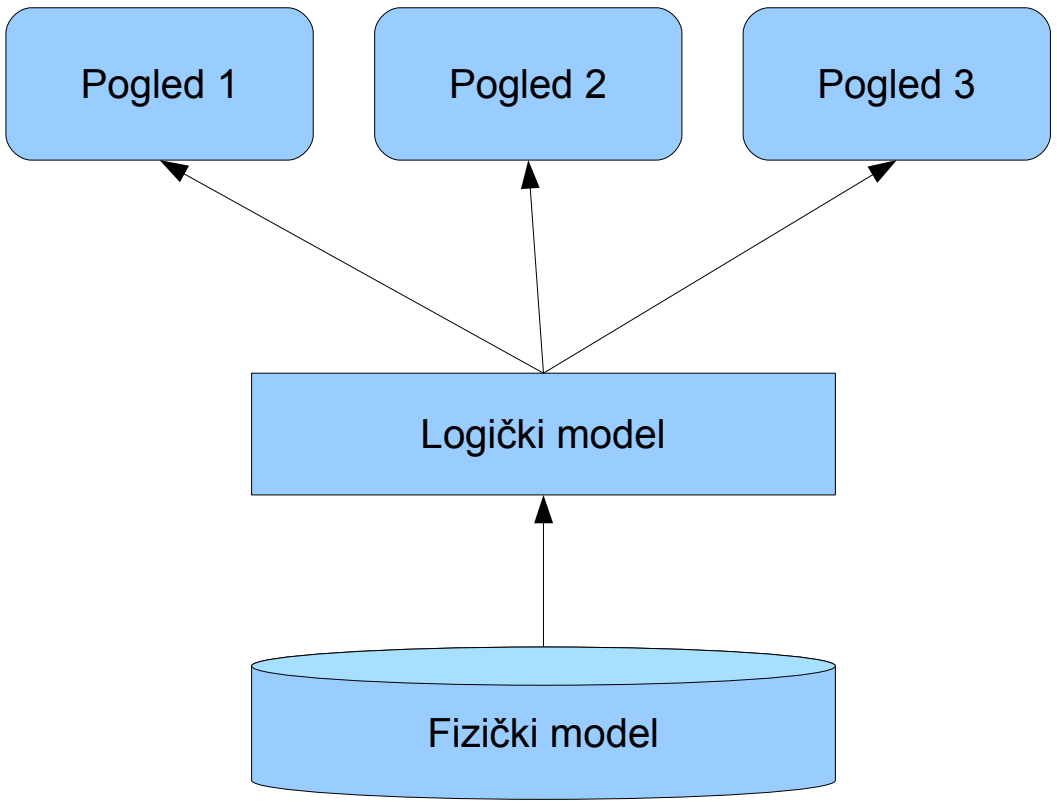
## DBMS - Database Management System

SUBP je programski sustav koji osigurava osnovne funkcije odabranog modela podataka u postupku kreiranja i korištenja baze podataka.

SUBP se sastoji od integrirane kolekcije programske podrške koja omogućava:

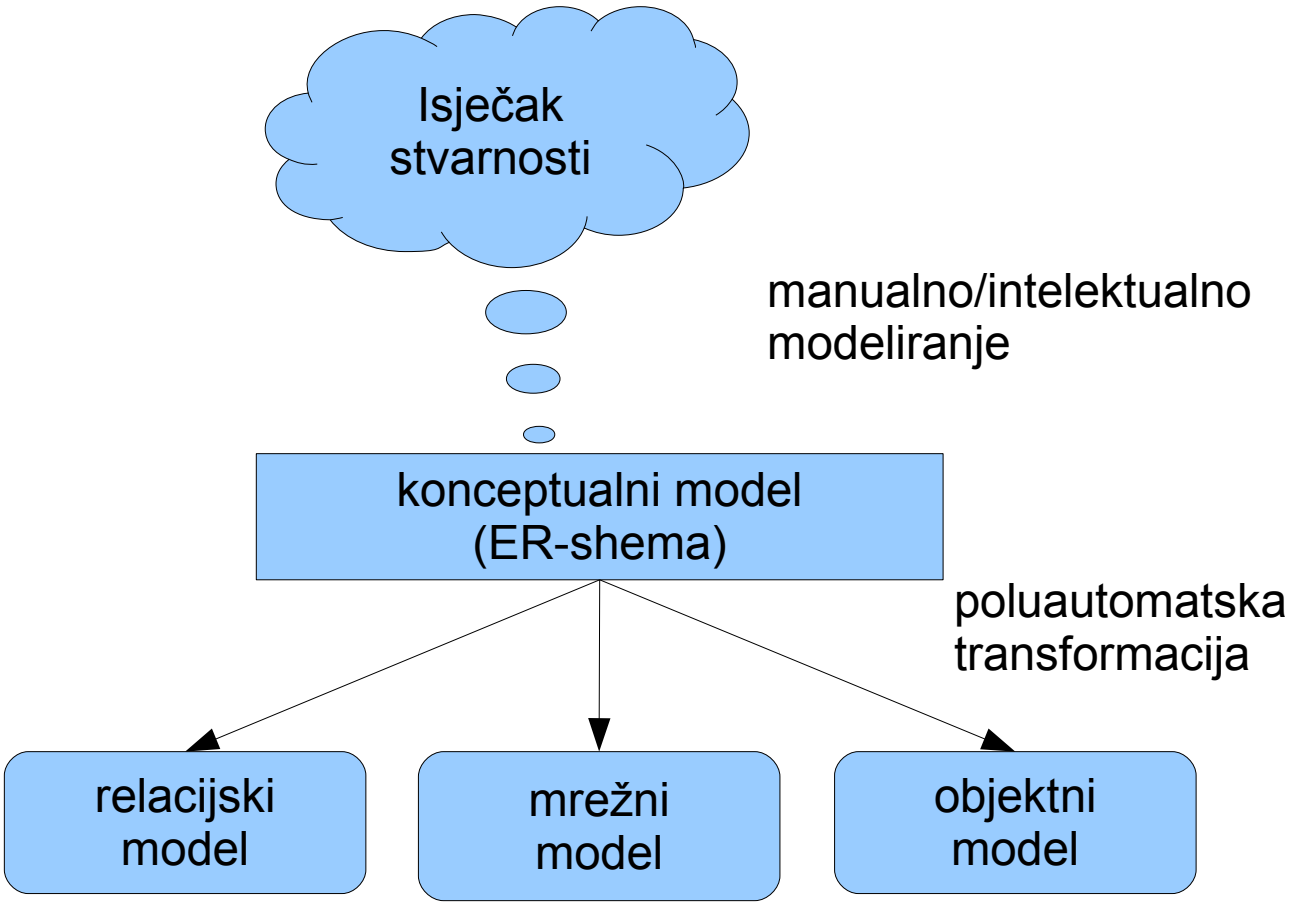
- opis i manipulaciju podacima pomoću posebnog jezika ili posebnih jezika
- visoki nivo sučelja prema podacima neovisan o strukturi podataka u računalu
- efikasno korištenje i razumijevanje informacija pohranjenih u bazi podataka, zahvaljujući skupu programskih alata (pomagala).

# Razine apstrakcije

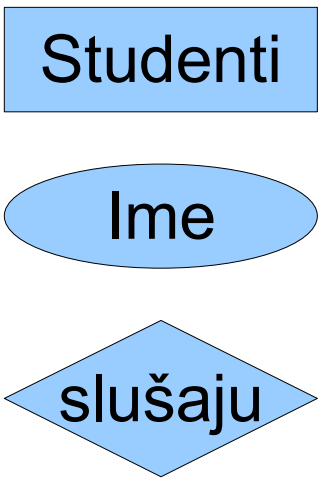



- fizička neovisnost podataka
- logička neovisnost podataka

# Modeliranje podataka



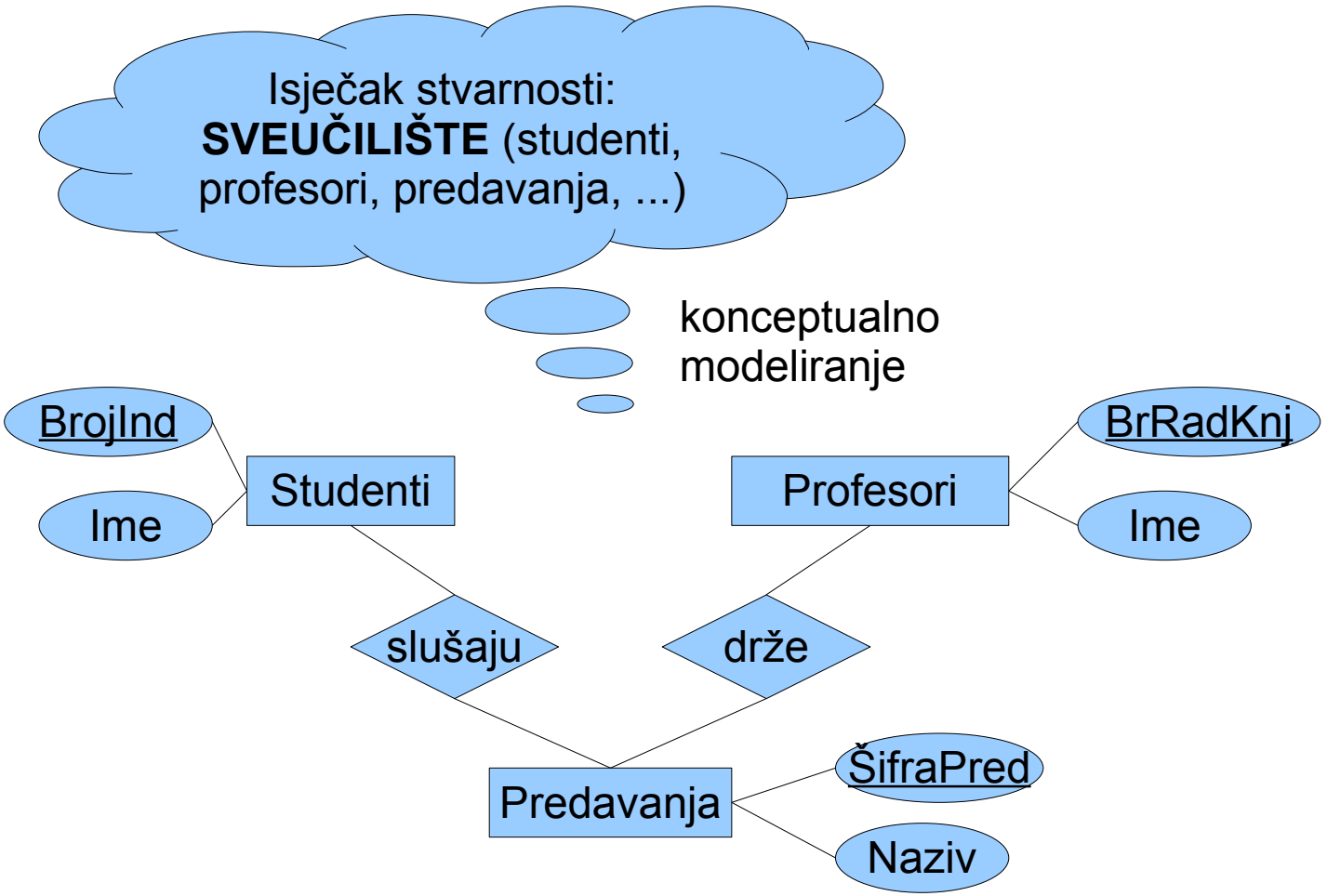
# ER-shema: Model entiteta i veza




Entiteti, veze i atributi čine ER ili ERA model. Simboli u ER-dijagramima su:

- entitet (eng. entity): pravokutnik
- atribut (eng. attribute): oval (elipsa)
- veza (eng. relationship): romb

# Model entiteta i veza



# Logički modeli podataka

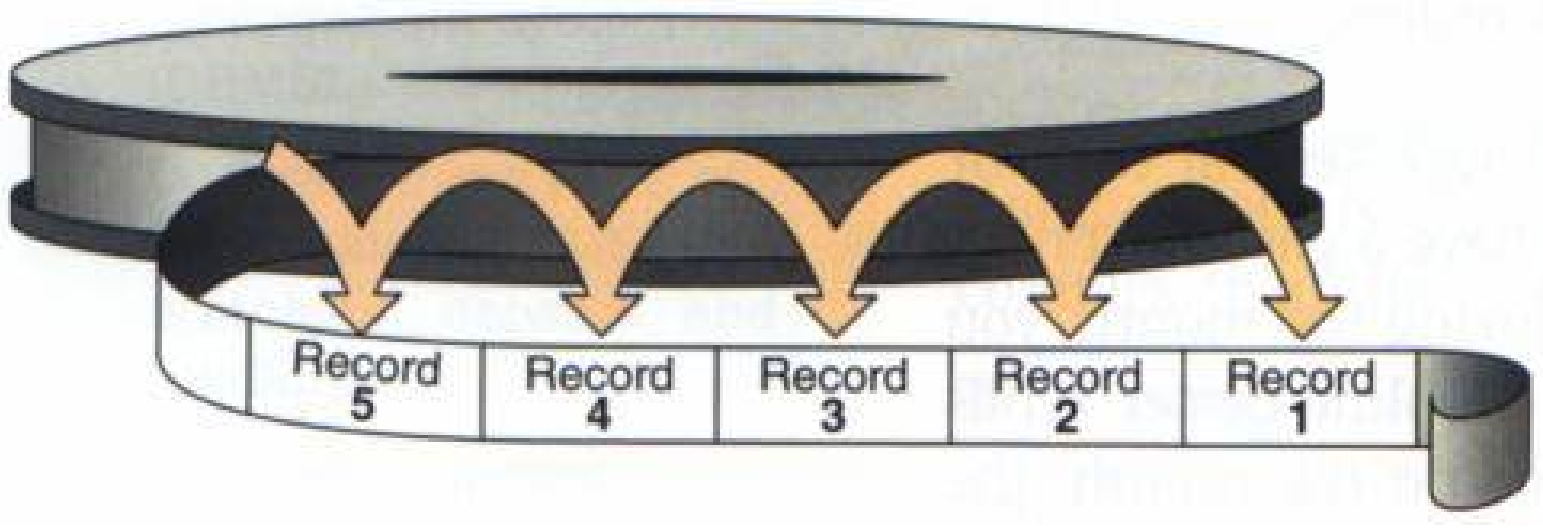
- mrežni model
- hijerarhijski model
- relacijski model
- objektni model
- objektno-relacijski model
- deduktivni model

# Logička organizacija podataka

- **Zapis** *(record)*: niz podataka povezan s jednim entitetom (npr. podaci o jednom studentu)
- **Polje** *(field)*: mjesto rezervirano za jedan element zapisa (npr. mjesto za prezime studenta)
  - polje **varijabilne** duljine (npr. komentar, popis ocjena)
  - polje **fiksne** duljine (npr. godina upisa na studij)
  - **ključ** jednoznačno definira zapis (npr. broj indeksa, JMBG)
- **Datoteka** *(file)*: niz zapisa obično istog tipa (npr. popis studenata Geodetskog fakulteta)

# Fizička organizacija podataka

- Datoteke su fizički spremljene na disk tako da blokovi na disku čuvaju zapise.
- Ako je kapacitet bloka manji od veličine zapisa, zapis će zauzeti dva ili više blokova.
- Ako je kapacitet bloka veći od veličine zapisa, blok će sadržavati više zapisa (najčešći slučaj).

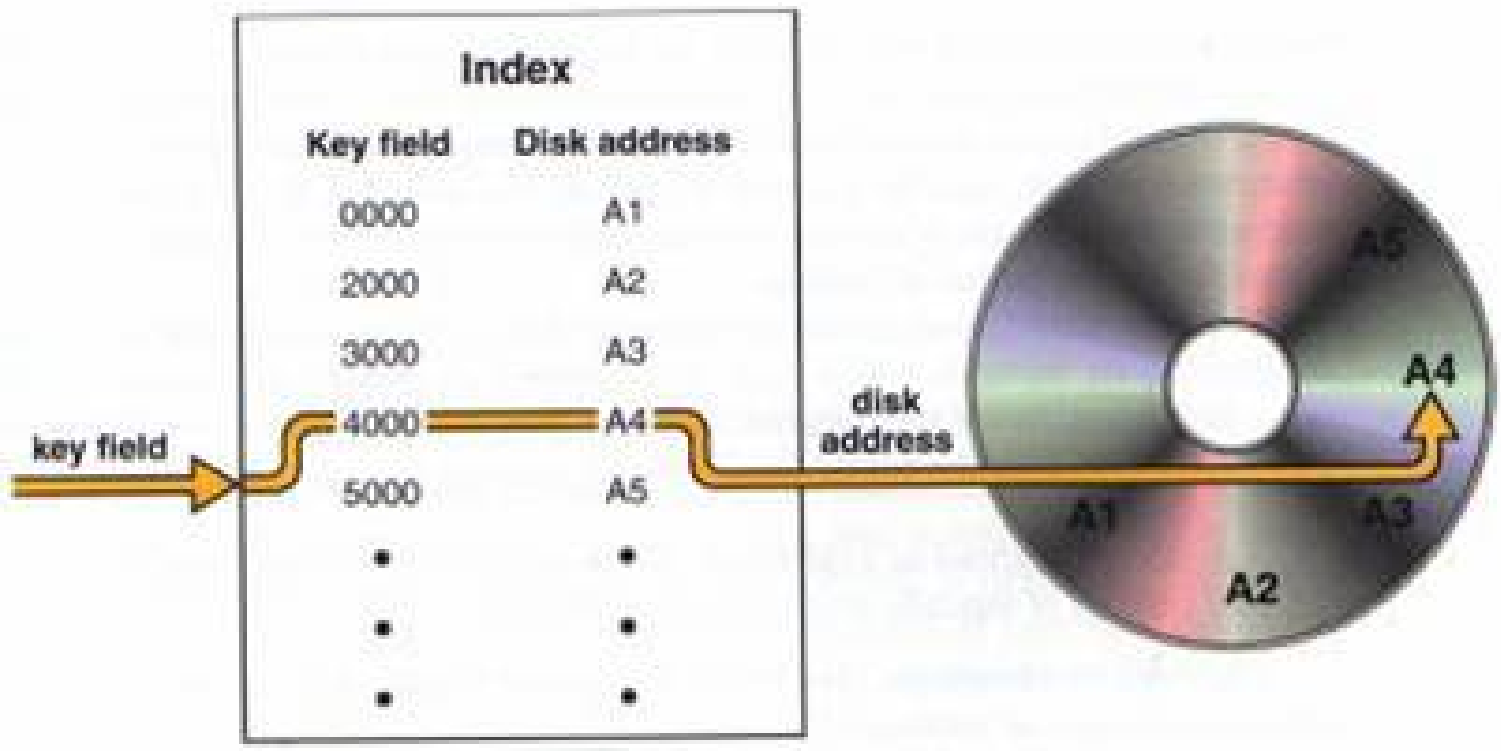


# Metode pristupa podacima

- **Otvaranje** *(open)*: priprema datoteku za pristup
- **Traženje** *(find)*: lociranje bloka na disku koji sadrži zapis koji zadovoljava postavljeni uvjet
- **Čitanje** *(read)*: kopira zapis u radnu memoriju
- **Brisanje** *(delete)*: briše zapis i sprema rezultat na disk
- **Izmjena** *(modify)*: mijenja polje u zapisu i sprema rezultat na disk
- **Dodavanje** *(insert)*: čita blok u radnu memoriju, dodaje novi zapis i sprema rezultat na disk
- **Zatvaranje** *(close)*: prekida pristup datoteci

# Pretraživanje podataka

- jednostavno i brzo pretraživanje (pronalaženje) - jedna od temeljnih funkcija baze podataka



# Linearno pretraživanje

- neuređene datoteke
- jednostavno dodavanje novih podataka
- složen direktni pristup (npr. naći ocjene studenta prema imenu: potrebno pretraživanje svih zapisa dok se ne pronade upravo onaj zapis u kojem se polje slaže s uvjetom),
- za n podataka prosjecno je potrebno ucitati n/2 zapisa
- linearno pretraživanje se još naziva brute-force (sirova snaga)

# Binarno pretraživanje

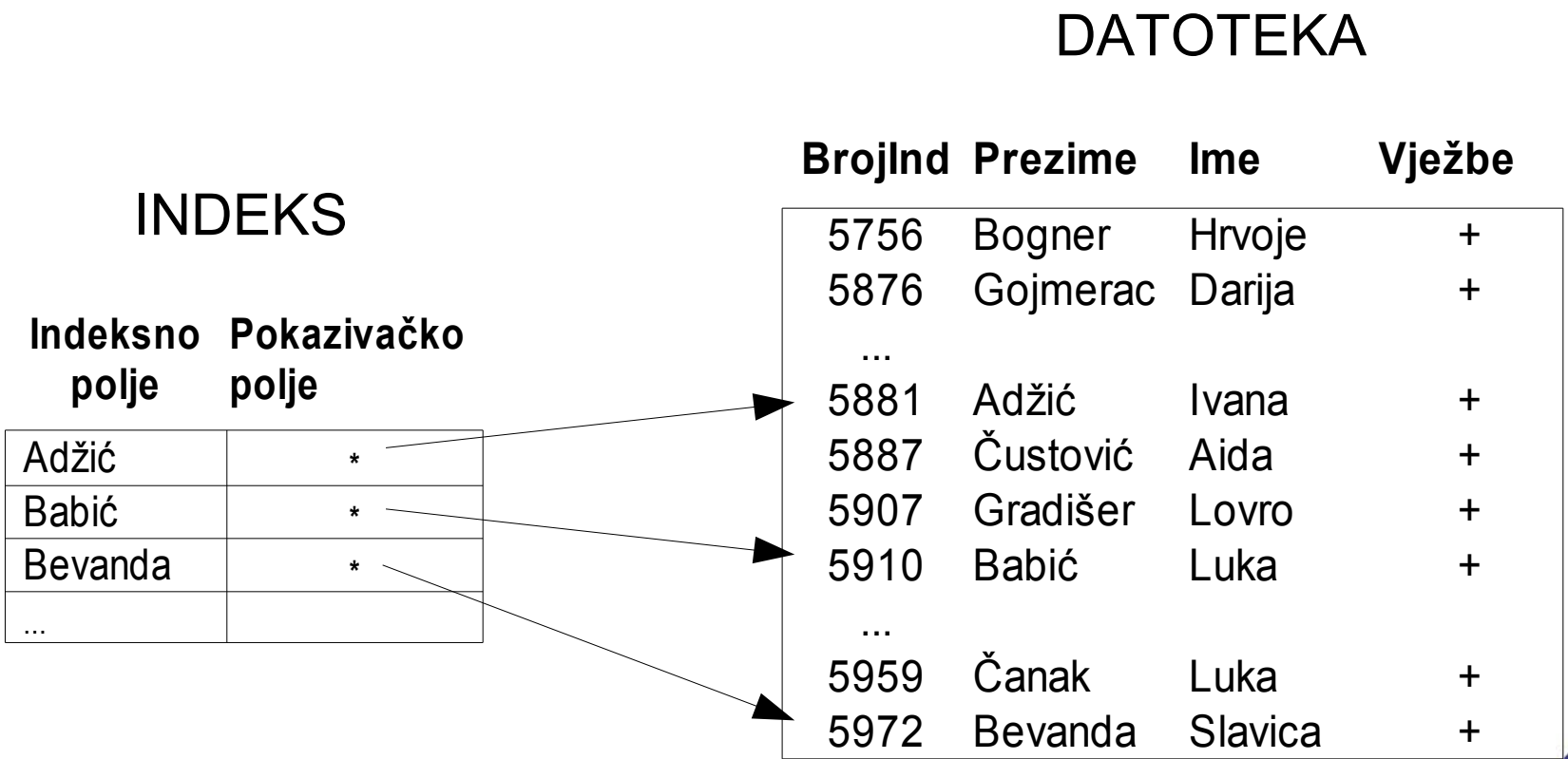
- uređene datoteke
- broj zapisa se u svakom koraku prepolovi
- za n podataka potrebno je maksimalno $log_2 n$ koraka, npr. za 1000 studenata poredanih po prezimenu dovoljno je 10 pristupa.

# Indeksiranje podataka

- Dobra strana: ubrzava pristup zapisima
- Loša strana: potreban dodatan prostor za spremanje indeksa
- Usporedba: knjigu bez indeksa pojmova bi trebalo svaki put pretražiti od prve do zadnje stranice
- **Indeks** je uređena datoteka ciji zapisi sadrže dva polja:
  - polje indeksa (sadrži sortirane vrijednosti indeksnog polja)
  - polje pokazivača (sadrži adrese blokova na disku koje imaju vrijednost indeksa)
- Pretraživanje indeksne datoteke je binarno.

# Indeksiranje podataka

- Pronalazak traženog indeksa vodi nas do zapisa koji sadrži pokazivač na zapis s ostalim poljima o traženom entitetu.



# Entiteti i atributi

## Entitet

**Entitet** je individualni primjer elementa stvarnog svijeta ili njegova isječka. Entitet je bilo što o čemu želimo prikupljati i spremati informacije. **Tip entiteta** je skup entiteta s istim svojstvima.

## Atribut

**Atribut** je svojstvo koje posjeduje neki entitet.

## Primjer

Najjednostavniji geometrijski element je točka opisana s barem dvije (nerazdvojne) koordinate (y i x).

# Odnos između entiteta

- ovisan o odabranom modelu podataka
- odnos može biti i entitet!
- binarni odnosi: 1:1, 1:m, m:n

| | | | | |
|---|---|---|---|---|
| Država | 1 | $\leftrightarrow$ | 1 | Glavni grad |
| Država | 1 | $\leftrightarrow$ | m | Županija |
| Država | m | $\leftrightarrow$ | n | Jezik |

# Primarni ključ

- **Primarni ključ** je minimalni skup atributa potreban za jednoznačno određivanje zapisa (npr. JMBG).
- Primarni ključ kojeg čine dva ili više atributa zovemo **kompozitni ključ** (npr. Fakultet i BrojIndeksa jednoznačno određuju studenta na Sveučilištu).
- **Sekundarni ključ** je bilo koji atribut koji nije kandidat za primarni ključ.
- Primarni ključ je s entitetom kojeg opisuje uvijek povezan asocijacijom 1:1 ili m:1.

## Primjer

Ime države može biti napisano na više različitih jezika (Germany, Deutschland, Njemačka), a ipak se odnosi na jedan te isti zapis u bazi podataka.

# Strani ključ

- **Strani ključ** *(foreign key)* je atribut koji služi kao primarni ključ drugog entiteta (npr. kratica države kao atribut grada).
- Strani ključ važan je kod povezivanja tablica: ako zapis u vezanoj tablici ne sadrži na mjestu primarnog ključa vrijednost koja se želi unijeti kao strani ključ, unos nije moguć.

## Primjer

Ako u tablici Student postoji uvjet da je poštanski broj strani ključ (Student.Grad), koji je ujedno primarni ključ u tablici Grad (Grad.GradID). U polje Student.Grad ne možemo unijeti vrijednost 20000, ako vrijednost 2000 nije prethodno unesena u polje Grad.GradID.

# Interna navigacija baza podataka

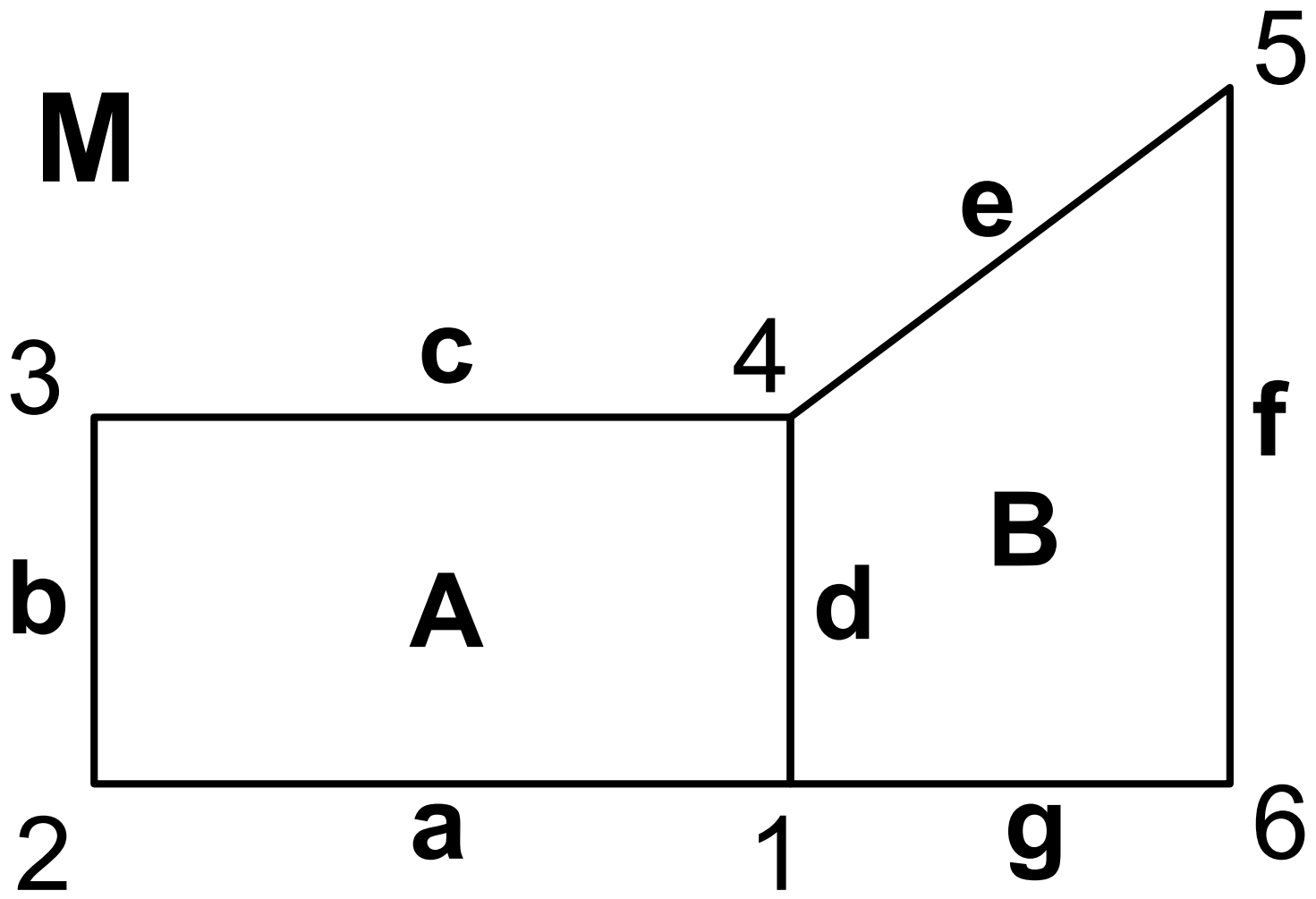
- Iako je sustav za upravljanje bazom podataka (DBMS) obično vrlo napredan program, za pojedine namjene obično se rade posebna, dodatna programska rješenja (aplikacije).
- Što manje pojedina aplikacija "zna" o internoj navigaciji baze - to bolje!
- Općenito, aplikacije bi se trebale baviti pitanjem:

  **ŠTO** je potrebno?

  dok DBMS (SUBP) odgovara na pitanje:

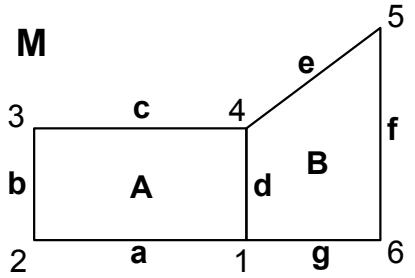
  **KAKO** realizirati zahtjev?

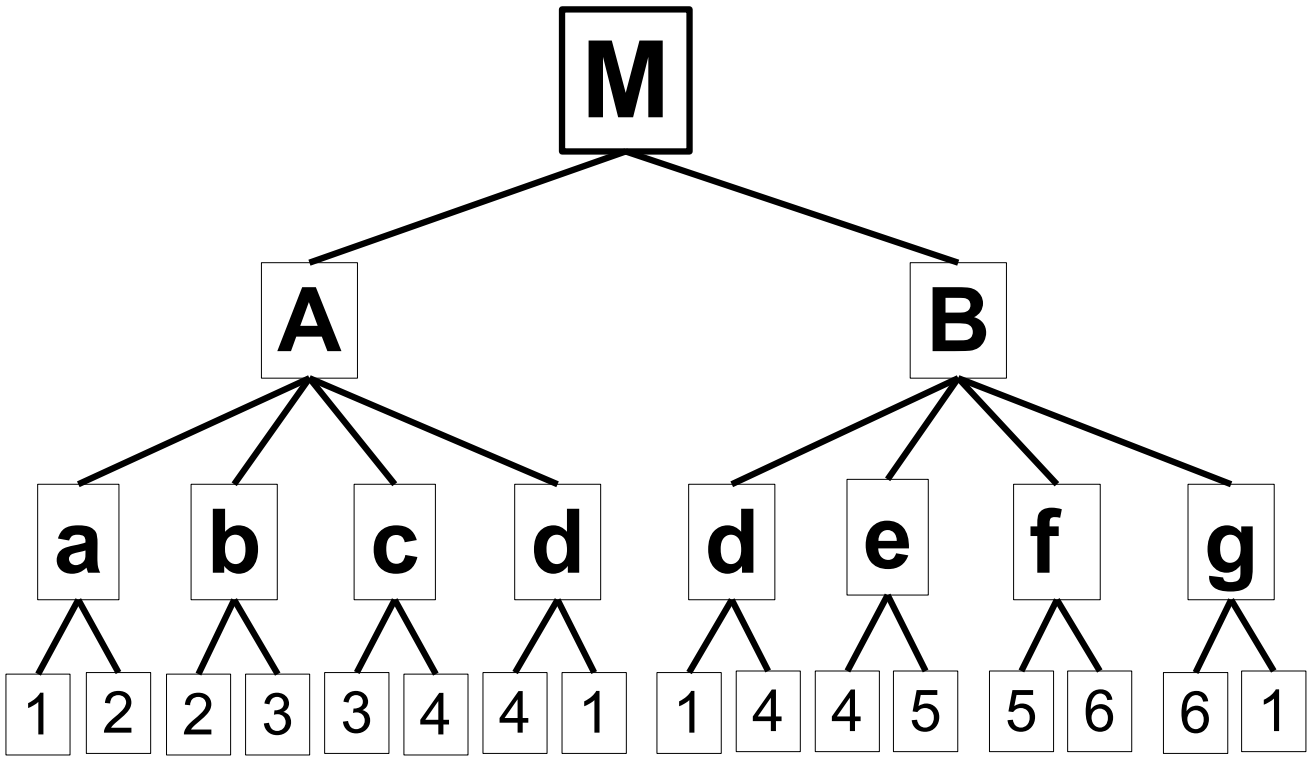
# Geometrijski primjer: karta M s dva poligona A i B



# Hijerarhijski model podataka

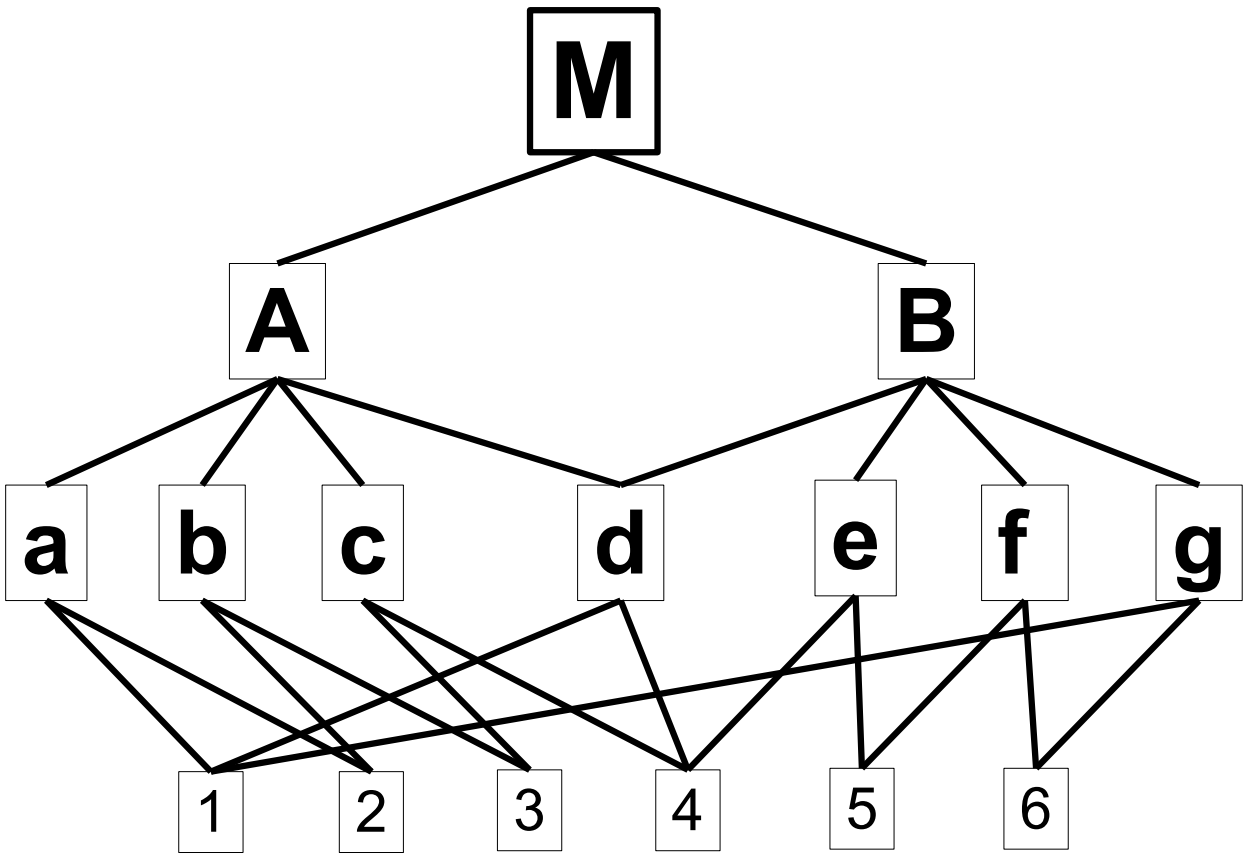
- "vlasnici" i "članovi": jedan član može imati samo jednog vlasnika
- putovi pretraživanja su fiksni



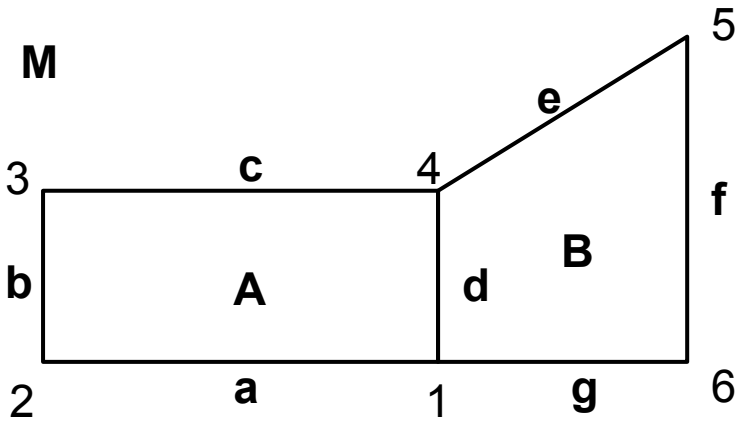



# Mrežni model podataka

- opći slučaj hijerarhijskog modela
- odnosi definirani eksplicitno
- aplikacija mora poznavati interni model baze podataka



# Relacijski model podataka




**Karta**

| M | A | B |
|---|---|---|

**Poligoni**

| A | a | b | c | d |
|---|---|---|---|---|
| B | d | e | f | g |

**Linije**

| a | 1 | 2 |
|---|---|---|
| b | 2 | 3 |
| c | 3 | 4 |
| d | 4 | 1 |
| e | 4 | 5 |
| f | 5 | 6 |
| g | 6 | 1 |

**Točke**

| 1 | x1 | y1 |
|---|---|---|
| 2 | x2 | y2 |
| 3 | x3 | y3 |
| 4 | x4 | y4 |
| 5 | x5 | y5 |
| 6 | x6 | y6 |

# Relacijske baze podataka

- Svi su podaci spremljeni u tabličnom obliku.
- Svakom tipu podataka odgovara jedna tablica.
- Zapisima odgovaraju reci u tablici.
- Poljima odgovaraju stupci u tablici.

| **Grad_ID** | **Ime** | **Geometrija** |
|---|---|---|
| ZAG | Zagreb | (y1,x1),... |
| MUC | München | (ym,xm),... |
| WIE | Beč | (yn,xn),... |
| GRZ | Graz | (yo,xo),... |
| BER | Berlin | (yp,xp),... |
| ... | ... | ... |

| **Država_ID** | **Ime** | **Geometrija** |
|---|---|---|
| GER | Njemačka | (yr,xr),... |
| AUT | Austrija | ... |
| CRO | Hrvatska | ... |
| SUI | Švicarska | ... |
| SLO | Slovenija | ... |
| ... | ... | ... |

# Relacije

- Temelj relacijskih baza podataka: **povezivanje entiteta različitog tipa.**
- Odnos grada i države moguće je i implicitno definirati: gradovi kao točke, države kao poligoni, provjeriti da li je točka u poligonu.

- Druga mogućnost je definicija predmetnog odnosa (relacije) uz pomoć nove tablice.

| Grad_ID | Država_ID |
|---|---|
| ZAG | CRO |
| MUC | GER |
| WIE | AUT |
| GRZ | AUT |
| BER | GER |
| ... | ... |

# Posebnosti relacijskih baza podataka

- Niti **redoslijed zapisa** niti **redoslijed polja** u tablici **nije** važan.
- Položaj (redoslijed) identičnih vrijednosti nije važan.
- Relacija se uspostavlja preko **primarnih** ključeva entitetnih tablica.
- Ako je primarni ključ atribut druge tablice, onda je to **strani ključ**.

# Posljedice (1)

- **Interna organizacija** neke tablice potpuno je **neovisna** od drugih tablica (npr. sortiranje).
- Korisnikov upit se bavi samo pitanjem **ŠTO?**, dok sustav za upravljanje relacijskim bazama podataka (RDBMS) rješava problem **KAKO?**.
- primjena je **neovisna** o *unaprijed zadanim putevima pristupa podacima* (hijerarhijski i mrežni model).
- tablice su bliske strukturama koje susrećemo u svakodnevnom životu (računi, kalendar, ...)

# Posljedice (2)

- Tablice je moguće jednostavno kombinirati, mijenjati i pretraživati: **relacijska algebra** je matematički precizno definirana.
- Upitni jezik sadrži **malen broj elemenata** i **lagano ga je naučiti** (SQL).
- Tablice se lako **proširuju** novim podacima.
- Tablice dozvoljavaju **prikaze specifične za pojedinog korisnika** (eksterni model).
- Postoje jednostavna pravila za spremanje podataka bez redundancije (normalne forme).

# Normalne forme

- Realizacija relacijskih baza podataka ostavlja veliki prostor za izbor entiteta, atributa i relacija.
- Ciljevi: što jednostavnije održavanje, konzistentnost, stabilna struktura podataka.
- Nepotrebno spremljeni podaci - nestabilna struktura - anomalije baza podataka.
- Opasnost: pri izmjeni podataka dolazi do neželjenih sporednih efekata.
- Relacijski model: jednostavna provjera pridržavanja normalnih formi.

# 1. normalna forma

Uvjeti 1. normalne forme:

1. veza između podataka na **logičkoj**, a ne na fizičkoj razini (adrese na disku),
2. za svaki tip entiteta postoji **jedan primarni ključ**,
3. svako **polje** unutar jednog entiteta ima **jednoznačno ime** koje se ne ponavlja.

Prva dva uvjeta su za relacijske baze neminovna, dok treći uvjet ne zadovoljava sljedeći primjer (jer se polje županija ponavlja):

| entitet: Država |
| --- |
| DržavaID, Naziv, GlavniGrad, Županija1, Županija2, ... |

# 2. normalna forma

Uvjeti 2. normalne forme:

1. moraju biti ispunjeni **uvjeti 1. normalne forme**,
2. svako **polje** unutar jednog entiteta koje nije dio primarnog ključa funkcionalno potpuno ovisi o **cijelom** (kompozitnom) ključu.

Druga normalna forma dolazi do izražaja kod primarnih ključeva sastavljenih od dva ili više polja (kompozitni primarni ključ).

| entitet: Student |
| --- |
| **ŠifraFakulteta**, **BrInd**, Ime, Prezime, AdresaFakulteta |

Polje AdresaFakulteta ovisi o dijelu kompozitnog primarnog ključa kojeg čine polja ŠifraFakulteta i BrInd.

# 3. normalna forma

Uvjeti 3. normalne forme:

1. moraju biti ispunjeni **uvjeti 2. normalne forme**,
2. ne postoji tranzitivna ovisnost **nijednog polja o nijednom ključu**.

Tranzitivnost: ako $a\ R\ b$ i $b\ R\ c$, slijedi $a\ R\ c$.

**entitet: Županija**

**Županija**, Država, GlavniGradDržave

Polje GlavniGradDržave ovisi o polju Država, a polje Država o polju Županija. Relacija nije u 3. normalnoj formi pa ju treba rastaviti na dvije relacije: Županija, Država, odnosno Država, GlavniGradDržave.

# Normalne forme - posljedice

- Postoje i viši stupnjevi normalnih formi.
- Normalne forme su skup pravila korisnih za modeliranje općih slučajeva baza podataka.
- Kod kompleksnih tipova podataka ponekad su neophodni kompromisi: odstupanje od idealnog rješenja.
- Takvi kompleksni tipovi podataka posebno se često javljaju u geoinformatici.
- Rezultat: objektno-relacijske baze podataka.

# Relacijska algebra

- Relacije opisuju i podatke i njihove odnose.
- Rezultat operacije nad relacijama je ponovo relacija (slično kao i kod brojeva: rezultat zbrajanja u skupu prirodnih brojeva je uvijek prirodan broj. Da li je isto s oduzimanjem?).
- Relacijska algebra je zatvoreni i konzistentni skup pravila koja su primjenjiva na relacije.

# Elementarne operacije relacijske algebre

- $\cup$ unija relacija
- $-$ razlika relacija
- $\cap$ presjek relacija
- $\pi$ projekcija
- $\sigma$ selekcija
- $\times$ Kartezijev produkt
- $\bowtie$ pridruživanje ili *join*

# Unija relacija: $C = A \cup B$

- koncept sukladan uniji skupova
- rezultirajuća relacija sadrži sve zapise iz obje tablice pri čemu se zapisi koji se ponavljaju spremaju samo jedanput
- UVJET: ulazne relacije (tablice) moraju biti kompatibilne, tj. moraju imati identična polja
- Primjer: pri udruživanju geodetskih točaka koordinate istih točaka u različitim tablicama mogu biti različite

# Razlika relacija: $C = A - B$

- rezultat ove operacije su svi zapisi sadržani u relaciji A, a koji nisu u relaciji B
- UVJET: kompatibilnost relacija
- Primjer: baza geodetskih točaka, tablica A sadrži sve točke, a tablica B samo izjednačene tocke; razlikom A-B dobijemo točke koje tek treba izjednačiti
- operacija u standardnom programskom jeziku vrlo složena, u RDBMS vrlo jednostavna

# Presjek relacija: $C = A \cap B$

- rezultat ove operacije sadrži samo zapise koji postoje i u relaciji A i u relaciji B.
- Primjer:
  - A: tablica zemalja u kojima se govori engleski jezik
  - B: tablica europskih zemalja
  - $C = A \cap B$ : tablica europskih zemalja u kojima se govori engleski jezik

# Projekcija: $B = \pi A$

- unarna operacija (samo jedan argument)
- izbor polja neke relacije (izbor stupaca neke tablice)
- Primjer: tablica s podacima o geodetskim točkama (n, y, x, H, naziv, tip točke, ...), a želimo samo broj točke i koordinate (n, y, x)
- $\pi_{(n,y,x)} Tocke$
- izbor podataka za eksterni model gdje isti podaci ne moraju ili ne smiju biti vidljivi svim korisnicima baze podataka

# Selekcija: $B = \sigma A$

- izbor zapisa koji zadovoljavaju postavljene uvjete
- Primjer: relacija Student ima polja: BrInd, Ime, Prezime, ProsjekOcjena
- Selekcijom se mogu izabrati samo oni studenti koji imaju prosjek ocjena veći od 3,5:
- $\sigma_{(ProsjekOcjena>3,5)} Student$
- ponekad nas zanimaju samo pojedini stupci iz zapisa koji zadovoljavaju uvjete pa se izraz selekcija primjenjuje i za kombinaciju selekcije i projekcije

# Kartezijev produkt: $C = A \times B$

- Rezultat je tablica u kojoj je svaki zapis iz prve tablice kombiniran sa svakim zapisom iz druge tablice.

**Primjer: Student $\times$ Predmet**

Student (BrInd,Ime,Prezime,Slusa)
Predmet (Sifra,Naziv,Satnica)
Student$\times$Predmet (BrInd,Ime,Prezime,Slusa,Sifra,Naziv,Satnica)

pri čemu je broj zapisa u rezultatu jednak umnošku broja zapisa u tablici Student s brojem zapisa u tablici Predmet.

# Pridruživanje: $C = A \bowtie B$

- poseban oblik Kartezijevog produkta
- razlikujemo unutarnji, vanjski (lijevi i desni) *join*
- u svakoj se tablici (relaciji) bira stupac (polje) preko čijih se vrijednosti uspostavlja veza

## Primjer: Student ⋈ Predmet

Student (BrInd, Ime, Prezime, Slusa)
Predmet (Sifra, Naziv, Satnica)
Student ⋈ Predmet (BrInd, Ime, Prezime, Slusa, Sifra, Naziv, Satnica) on (Slusa = Sifra)
U rezultatu će biti samo kombinacije zapisa iz tablica Student i Predmet u kojima polje Slusa ima jednaku vrijednost polju Sifra.

# Relacijska algebra - zaključak

- Opisane operacije relacijske algebre primjenjuju se uvijek u kombinaciji:
  1. iz više tablica **selektiramo** samo one zapise koji udovoljavaju uvjetima
  2. napravimo pridruživanje (**join**)
  3. **projekcijom** odaberemo željena polja
- Na taj se način iz vrlo malog broja osnovnih operacija relacijske algebre može izvesti veliki broj kombinacija za obradu i analizu podataka.
- Prijevod operacija relacijske algebre u jezik za definiciju i manipulaciju podacima koji danas predstavlja osnovni standard za relacijske baze podataka: SQL.

# Povijest relacijskih upitnih jezika

- 1970-ih: IBM – DBMS System – upitni jezik SEQUEL (Structured English Query Language)
- 1980-ih: SEQUEL preimenovan u SQL (Structured Query Language), komercijalni sustav SQL/DS, pojava novih relacijskih DBMS-a (Oracle, Informix), potreba za standardizacijom upitnog jezika
- 1986: prvi SQL-standard ANSI-komisije
- 1989: prva revizija standarda
- **1992**: proširenje na **SQL-92** (SQL 2)
- 2003: objavljena prva verzija SQL 3 standarda

# SQL-92

- upitni jezik zasnovan na relacijskoj algebri
- teorijske postavke pojednostavljene zbog jednostavnije uporabe i veće brzine
- deklarativni jezik (što? a ne kako?)
- jezik nije samo "upitni", nego služi i za definiciju i manipulaciju podacima: DDL (Data Definition Language) i DML (Data Manipulation Language)
- SQL ne obraduje relacije u matematičkom smislu, nego tablice
- zapis i polje postaju redak i stupac
- osim manipulacije tablicama, SQL omogućuje definiciju integriteta podataka, prava pristupa i kontrolu transakcija

# Osnovni tipovi podataka u SQL-92

- Tri temeljna tipa podataka:
  - brojevi (numeric)
  - znakovni nizovi (character)
  - datum/vrijeme (date, time)
- Brojevi mogu biti cjelobrojni (integer) i decimalni (float).
- Znakovni nizovi mogu biti fiksne duljine (character) ili varijabilne duljine (varchar).
- Datum i vrijeme mogu biti definirani pojedinačno ili kao intervali.
- Osim navedenih, postoji i BLOB (Binary Large OBject) odnosno RAW za binarne tipove podataka (npr. u CAD-primjenama).

# Definiranje sheme baze podataka: CREATE TABLE

```
CREATE TABLE Nastavnik
 (BrRadKnj INTEGER NOT NULL,
  Ime      VARCHAR(10) NOT NULL,
  Titula   CHARACTER(4));
```

- popis atributa i tipa podatka za svaki atribut
- NOT NULL: primjer definicije uvjeta integriteta podataka (nije moguce zadati nastavnika bez imena)

# Izmjena sheme baze podataka: ALTER TABLE

- Ako bismo htjeli dodati broj sobe za svakog nastavnika:

```
ALTER TABLE Nastavnik
  ADD COLUMN Soba INTEGER;
```

- Ako je 10 znakova bila pogrešna procjena za duljinu imena nastavnika:

```
ALTER TABLE Nastavnik
  MODIFY COLUMN Ime VARCHAR(30);
```

- Prethodna naredba ne isključuje ranije zadani uvjet integriteta (not null).

# Manipulacija podacima

- dodavanje redaka (punjenje podacima):

```
INSERT INTO Nastavnik VALUES (21366, 'Medak', 'prof', 75)
```

- izbor podataka na temelju kriterija (selekcija):

```
SELECT BrRadKnj, Ime
  FROM Nastavnik
  WHERE  Titula = 'prof';
```

- **select** određuje atribute koji će se pojaviti u rezultatu
- **from** određuje iz koje se relacije traže podaci
- **where** određuje uvjet kojeg reci u rezultatu moraju zadovoljiti

# Manipulacija podacima: sortiranje

- snaga SQL-a je u bliskosti naredbi s razgovornim jezikom: najčešće je dovoljan prijevod naredbe s engleskog (“izaberi broj i ime nastavnika čija je titula prof”)
- rezultat prethodnog upita bila je lista bez utvrđenog redoslijeda
- SQL omogućuje sortiranje prema bilo kojem atributu:

```
SELECT BrRadKnj, Ime, Titula
  FROM Nastavnik
  ORDER BY Titula DESC, Name ASC;
```

pri čemu je ascending uzlazno sortiranje (a-z, 1-9), a descending silazno sortiranje (z-a, 9-1).

# Višetablični upiti

Povezivanje tablica: Tko predaje "Baze podataka"?

```
SELECT Ime, Titula
  FROM Nastavnik, Predmet
  WHERE BrRadKnj=Drzi and Naziv="Baze podataka";
```

- prijevod: "Izaberi ime i titulu iz kombinacije nastavnika i predavanja pri čemu je vrijednost stupca **Drži** u tablici **Predmet** jedanaka vrijednosti stupca **BrRadKnj** u tablici **Nastavnik**, a naziv predmeta je "Baze podataka" "
- Obrada upita se odvija u tri faze:
  1. Kartezijev produkt tablica Nastavnik i Predavanje
  2. za svaki redak produkta se ispituje uvjet postavljen pod WHERE
  3. projekcijom se biraju pod SELECT zadani stupci (atributi)

# Višetablični upiti - pridruživanje (join)

Povezivanje tablica: Koji studenti slušaju koje predmete?

```
SELECT Ime, Naziv
  FROM Student, Slusa, Predmet
  WHERE Student.BrInd=Slusa.BrInd AND
        Slusa.Sifra=Predmet.Sifra;
```

- Elegantniji zapis uz pomoć varijabli (aliasa):

```
SELECT s.Ime, p.Naziv
  FROM Student as s, Slusa as h, Predmet as p
  WHERE s.BrInd=h.BrInd AND
        h.Sifra=p.Sifra;
```

# Višetablični upiti - eksplicitni join

Povezivanje tablica: Koji studenti dolaze iz kojeg grada?

STUDENT:

| BrInd | Ime | Prezime | Grad |
|---|---|---|---|
| 4350 | Ivana | Ivic | 10000 |
| 4230 | Marko | Maric | 20000 |
| 5140 | Ana | Anic | 21000 |
| 4760 | Mijo | Mijic | 21000 |
| 5145 | Pero | Peric | |

GRAD:

| GradID | Naziv |
|---|---|
| 10000 | Zagreb |
| 20000 | Dubrovnik |
| 10410 | Velika Gorica |
| 21000 | Split |
| 44330 | Novska |
| 34000 | Pozega |

```
SELECT Ime, Prezime, Naziv
  FROM Student JOIN Grad on (Grad=GradID);
```

# Višetablični upiti - eksplicitni join

Rezultat: samo oni zapisi za koje postoje identične vrijednosti za strani ključ Grad u tablici Student i primarni ključ GradID u tablici Grad.

```
STUDENT JOIN GRAD:
Ime   Prezime Naziv
----- ------- -----------
Ivana Ivic    Zagreb
Marko Maric   Dubrovnik
Ana   Anic    Split
Mijo  Mijic   Split
```

# Višetablični upiti - left join

Povezivanje tablica: Svi studenti, pa i oni koji nisu iz grada?

```
SELECT Ime, Prezime, Naziv
  FROM Student LEFT JOIN Grad on (Grad=GradID);
```

- U rezultatu se pojavljuju svi zapisi iz tablice Student bez obzira postoji li odgovarajuća vrijednost u tablici Grad.

```
STUDENT LEFT JOIN GRAD:
Ime   Prezime Naziv
----- ------- -----------
Ivana Ivic    Zagreb
Marko Maric   Dubrovnik
Ana   Anic    Split
Mijo  Mijic   Split
Pero  Peric
```

# Agregiranje u SQL-u

- Agregativne funkcije u SQL-u izvode operaciju na skupu zapisa i svode skup vrijednosti na jednu vrijednost.
- Primjeri:
  1. **avg** (prosjek skupa brojeva)
  2. **max, min** (najveci, odnosno najmanji element skupa)
  3. **sum** (zbroj svih vrijednosti)
  4. **count** (broj vrijednosti)
- Srednji broj semestara svih studenata je tako:

```
SELECT AVG(Semestar)
  FROM Student;
```

# Grupiranje u SQL-u...

- Naredba **group by** određuje podgrupe na koje će agregativne funkcije biti primijenjene.
- Tako je npr. iz tablice Predmet (Sifra, Naziv, Satnica, Drzi) moguće izračunati satnicu pojedinih nastavnika:

```
SELECT Drzi, SUM(Satnica)
  FROM Predmet
  GROUP BY Drzi;
```

- Objašnjenje: u rezultatu su svi zapisi (predmeti) u tablici koje imaju jednaku vrijednost za atribut Drzi grupirani u jedan, te je za svaki izračunat zbroj satnica.

# ... Grupiranje u SQL-u

- Uz pomoć naredbe **having** moguće je postaviti dodatan uvjet kojeg mora ispuniti grupa definirana s **group by**
- Primjer: grupirati predmete prema nastavnicima i za svaku grupu (svakog nastavnika) izračunati prosjek satnice, te uzeti u obzir samo one nastavnike koji imaju više od tri sata tjedno :

```
SELECT Drzi, SUM(Satnica)
  FROM Predmet
  GROUP BY Drzi
    HAVING AVG(Satnica)>3;
```

- **having** ima slično značenje kao i **where** u uobičajenim upitima: razlika je u tome što se **having** izvršava nakon grupiranja pa je u uvjetu moguće koristiti agregatne funkcije, što kod **where** nije moguće.

# SQL - zaključne napomene

- SQL predstavlja standardizirani oblik komunikacije korisnika s relacijskim sustavom za upravljanje bazom podataka (RDBMS).
- Naredbe napisane u SQL-u su daleko moćnije od grafičkih korisničkih sučelja za upisivanje upita prema primjeru (Query By Example).
- SQL-92 je podržan od velike većine komercijalnih (Oracle, Informix, DB2) i slobodnih proizvoda (PostgreSQL)
- SQL 3 ce predstavljati bitno proširenje standarda novim funkcijama koje proširuju relacijski model u smjeru objektno-relacijskog modela.

# Transakcije

## Transakcija

Transakcija je skup operacija koji prevodi bazu podataka iz jednog konzistentnog stanja u drugo konzistentno stanje.

- Primjer: plaćanje s jednog računa uključuje barem dvije operacije: umanji iznos na jednom računu i uvećaj iznos na drugom. Ako je prva operacija izvršena, a druga ne, novac bi mogao nestati, čime bi bilo narušeno konzistentno stanje.
- Sustav za upravljanje bazom podataka mora zadovoljiti četiri tzv. ACID uvjeta da bi omogućio pravilan rad s transakcijama.

# ACID uvjeti

**atomicity** atomičnost transakcije: transakciju nije moguće dijeliti na elementarnije transakcije. Ili su izvršene sve operacije unutar transakcije ili nijedna.

**consistency** konzistentnost: baza podataka mora uvijek zadržati konzistentno stanje.

**isolation** izolacija: dok transakcija nije izvršena, njezine operacije ne utječu na druge, istodobno izvršavane transakcije.

**durability** trajnost: nakon što je transakcija izvršena, promjene ostaju trajno pohranjene, čak i u slučaju pada sustava.

# Konzistentnost baza podataka

- Za pojedine atribute u tablicama moguće je u SQL-u definirati točno određene uvjete (*constraints*) koje vrijednosti atributa moraju zadovoljiti.
- Primjer prvi: NOT NULL onemogućuje prazno polje u zapisu.
- Primjer drugi:

```
CREATE TABLE Posao (
  ...
  min_placa NUMERIC NOT NULL,
  max_placa NUMERIC NOT NULL,
  ...
  CHECK (min_placa < max_placa);
```

# Sigurnost baza podataka ...

- Fizička sigurnost: više kopija na različitim mjestima (u različitim zgradama), naredba SHADOW kreira istovjetnu kopiju
- Logička sigurnost: osigurati autorizirani pristup bazi podataka (korisničko ime + lozinka)
- U SQL-u moguće definirati autorizaciju pristupa svakog korisnika za pravo izvođenja pretraživanja odabranih tablica:

```
GRANT SELECT ON Student TO medak;
```

# ... Sigurnost baza podataka

- Za naredbe koje mijenjaju vrijednosti u tablicama (INSERT, UPDATE, REFERENCE) moguće je izabrati pojedine stupce za koje se dozvoljava pristup:

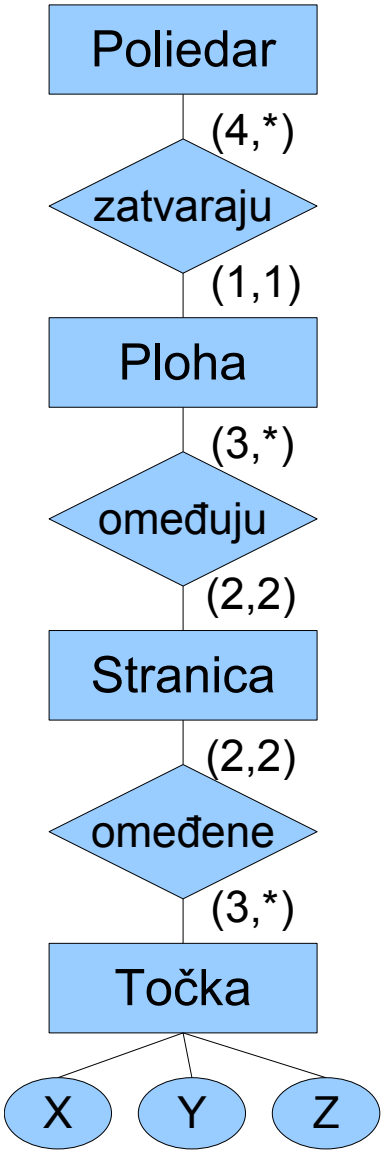
```
GRANT UPDATE (Datum_ispita, Ocjena)
  ON Ispit
  TO medak;
```

- Sve je dozvole pristupa moguće povući naredbom REVOKE.
- Svrstavanje korisnika u grupe olakšava administraciju baze podataka jer svaki korisnik 'nasljeđuje' prava grupe kojoj pripada.

# Objektne baze podataka

- Relacijske baze podataka imaju dvije velike prednosti:
  - upitni jezik je standardiziran (SQL-92)
  - relacijski model ima čvrst teorijski temelj: relacijsku algebru
- Relacijski model posebno je pogodan za podatke koji se obično prikazuju tablicama (bankovni računi, trgovina, računovodstvo, ...).
- Geometrijski podaci su kompleksniji, npr. tijela u 3D prostoru (vidi sliku na sljedećoj stranici):
  - poliedar zatvaraju 4 ili više ploha,
  - svaka ploha je omeđena s 3 ili više stranica,
  - svaka stranica ima točno jednu početnu i jednu završnu točku,
  - svaka točka ima tri koordinate.

# 3D - ER i relacijski model




**Poliedar**

| PolyID | Masa | Volumen | Materijal |
|---|---|---|---|
| **tetra01** | 25,7 | 11,9 | Željezo |
| kocka04 | 45,9 | 15,3 | Staklo |

**Ploha**

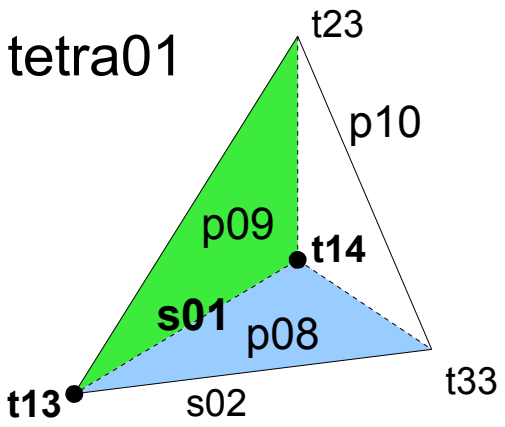
| PlohaID | PolyID | Boja | Šrafura |
|---|---|---|---|
| **p08** | tetra01 | plava | ne |
| **p09** | tetra01 | zelena | ne |
| ... | | | |

**Stranica**

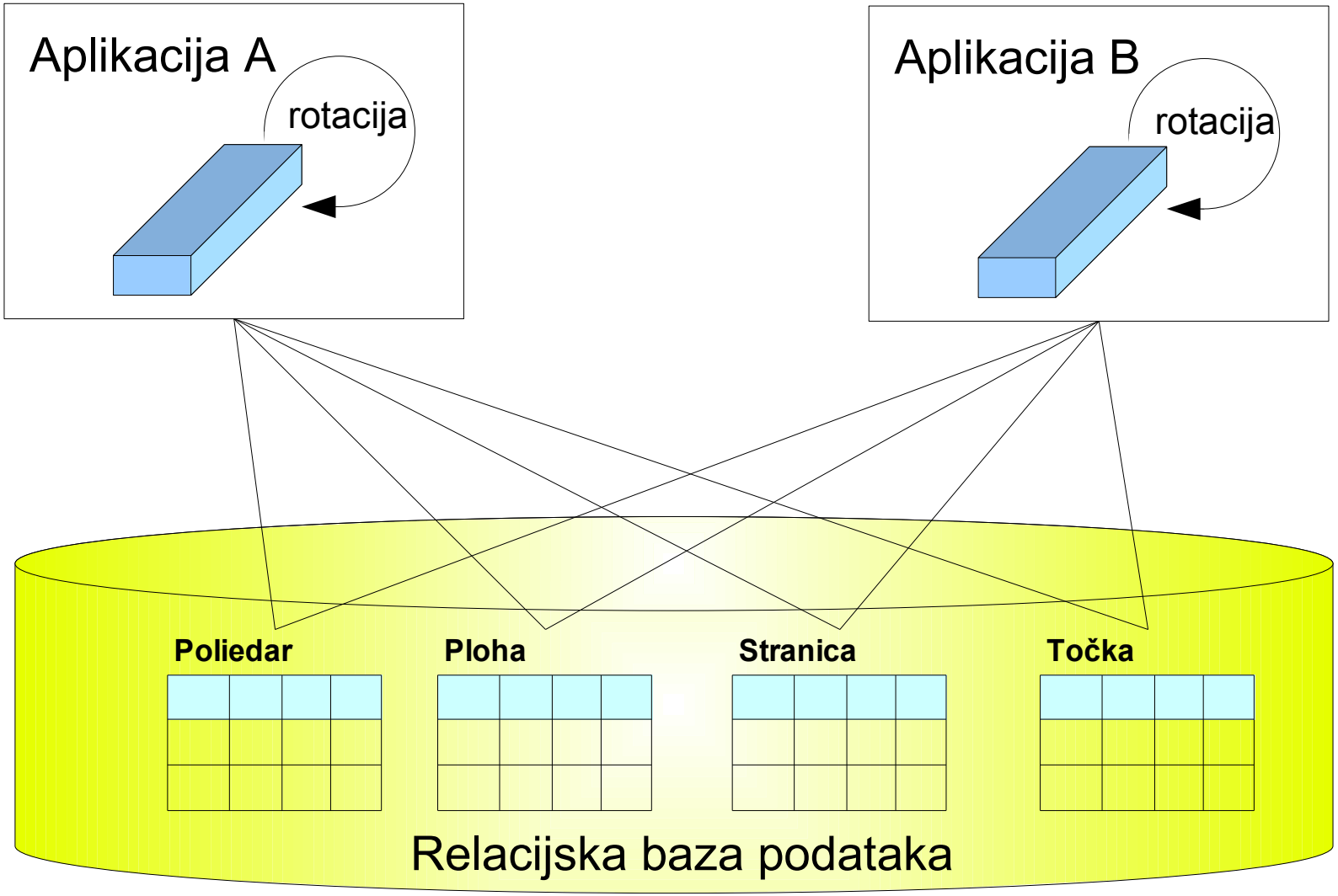
| StranicaID | PlohaL | PlohaD | TočkaPoč | TočkaKraj |
|---|---|---|---|---|
| **s01** | **p09** | **p08** | **t13** | **t14** |
| s02 | p08 | p10 | t13 | t33 |
| ... | ... | ... | ... | ... |
| s78 | p31 | p67 | t87 | t89 |

**Točka**

| TockaID | X | Y | Z |
|---|---|---|---|
| **t13** | 0,00 | 0,00 | 0,00 |
| **t14** | 3,00 | 4,00 | 0,00 |
| ... | | | |



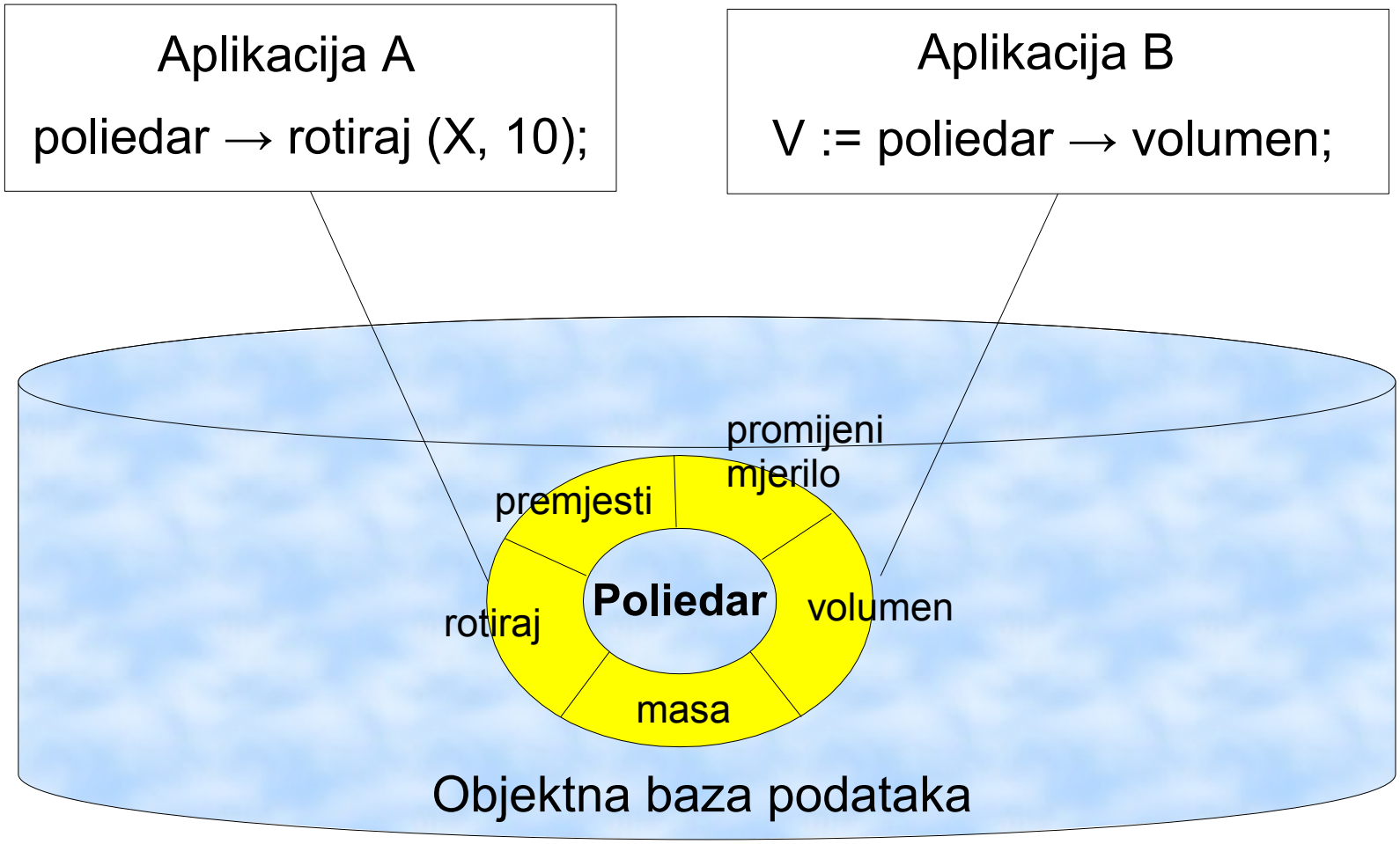
# Rotacija definirana izvan RDBMS-a



# Nedostaci relacijskog modela

- Segmentiranje: jedan objekt se obično segmentira na više relacija, čijim se pridruživanjem (join) ponovno uspostavlja cjelina - kompleksna i dugotrajna operacija.
- Umjetni ključevi: nužno uvođenje dodatnih atributa kao ključeva (xxxID)
- Nemoguće je definirati ponašanje (eng. *behavior*), npr. promjena mjerila, rotacija i translacija, volumen, masa nisu definirani u logičkom modelu, nego ovise o konkretnoj aplikaciji.
- Eksterno programersko sučelje: nužno povezivanje relacijskog jezika (SQL) s nekim drugim programskim jezikom (C, Java), što vodi dupliciranju rješenja istih operacija (npr. rotacija poliedra)

# Rotacija definirana u OODBMS-u



# Prednosti objektno-orijentiranog modela

- Prikrivanje informacija (*encapsulation*): pristup atributima isključivo preko operacija.
- Ponovna upotrebljivost (*reusability*): definicije u natklasi vrijede za sve potklase.
- Operacije su realizirane direktno u jeziku objektnog modela.
- Nasljeđivanje (*inheritance*)
- Višeobličnost (*polymorphism*): operacija '+' zbraja i cijele i decimalne brojeve.

# Obilježja objekata

- Relacijski model:
  - entiteti u obliku zapisa,
  - zapisi se sastoje iz konačnog broja polja,
  - vrijednosti u poljima su elementarne i nepromjenjive (npr. broj 2)
- U objektnom modelu objekt ima tri dijela:

  **Identitet:** svaki objekt ima jedinistveni identifikator, koji se za vrijeme životnog ciklusa objekta ne mijenja.

  **Tip (klasa):** određuje strukturu i ponašanje objekta. Konkretni objekti nastaju kao instance tipa objekta.

  **Vrijednost (stanje):** u nekom trenutku životnog ciklusa objekt ima određeno stanje dato vrijednostima atributa i postojećih veza s drugim objektima.

# Ukrali identitet 30.000 ljudi (VL, 27.11.2002.)

NEW YORK – Državno tužiteljstvo u New Yorku optužilo je jučer trojicu ljudi za **krađu identiteta** oko 30 tisuća osoba kojima je nanesena šteta od gotovo 3 milijuna dolara. Trojka je tijekom tri godine krala podatke iz baza podataka kreditnog odjela tvrtke Ford, kreditne agencije Experian i još nekih tvrtki. Podatke o osobama preprodavali su za 60 dolara. "Kupci" novog identiteta mogli su, lažno se predstavljajući, nesmetano tražiti duplikate kreditnih ili bankovnih kartica, podizati kredite. Državni tužitelj na Manhattanu kaže da su na takav način pribavili oko 15 tisuća kreditnih kartica, te da je to **najveća krađa identiteta u povijesti SAD-a**.

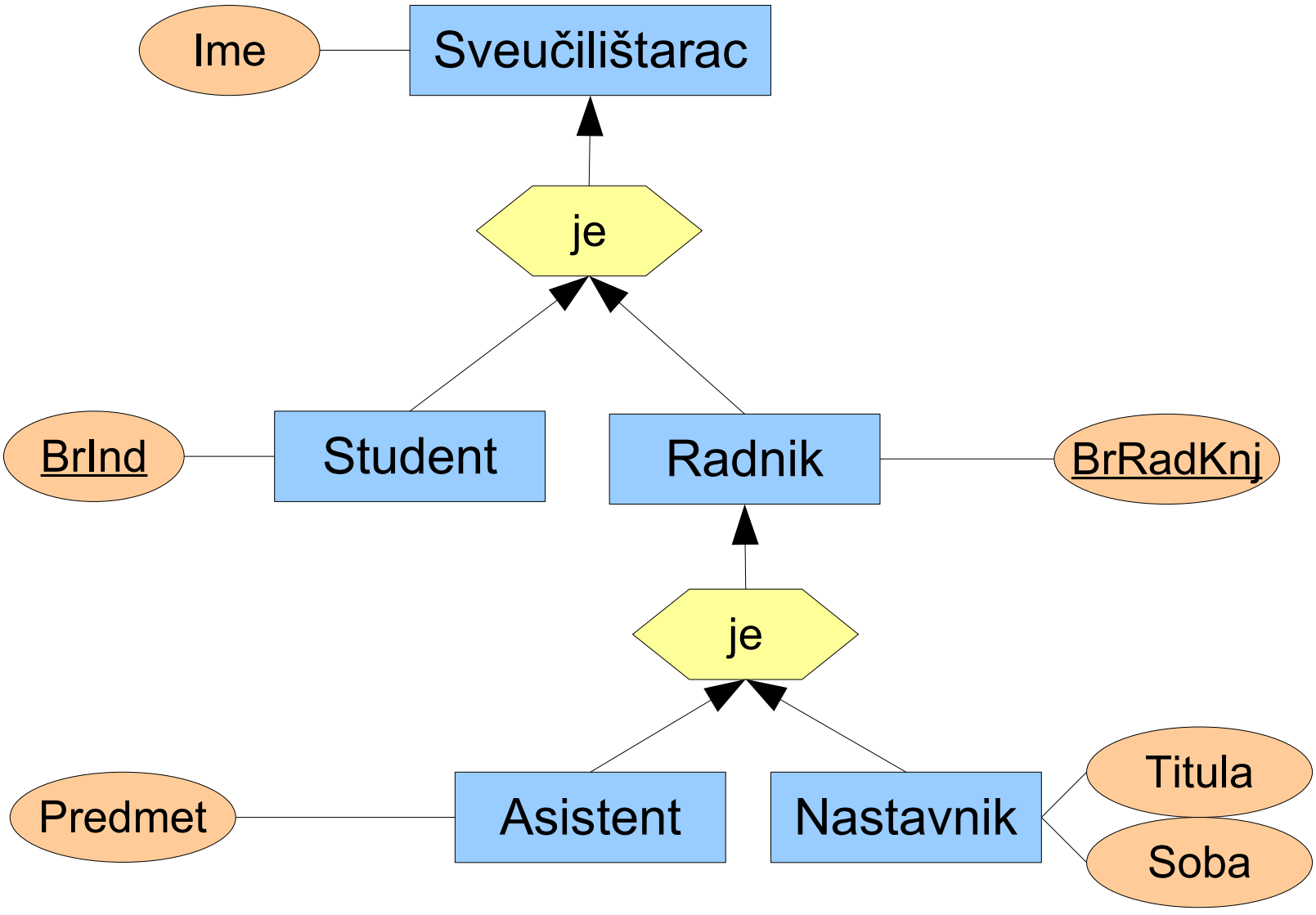
# JMBG - što znači trinaesta znamenka?

- Trinaesta znamenka je kontrola da li je prvih 12 ispravno upisano.
- Dobiva se računskim putem - neka su znamenke DDMMGGGRRBBBK date kao niz *abcdefghijklm*, tada je:

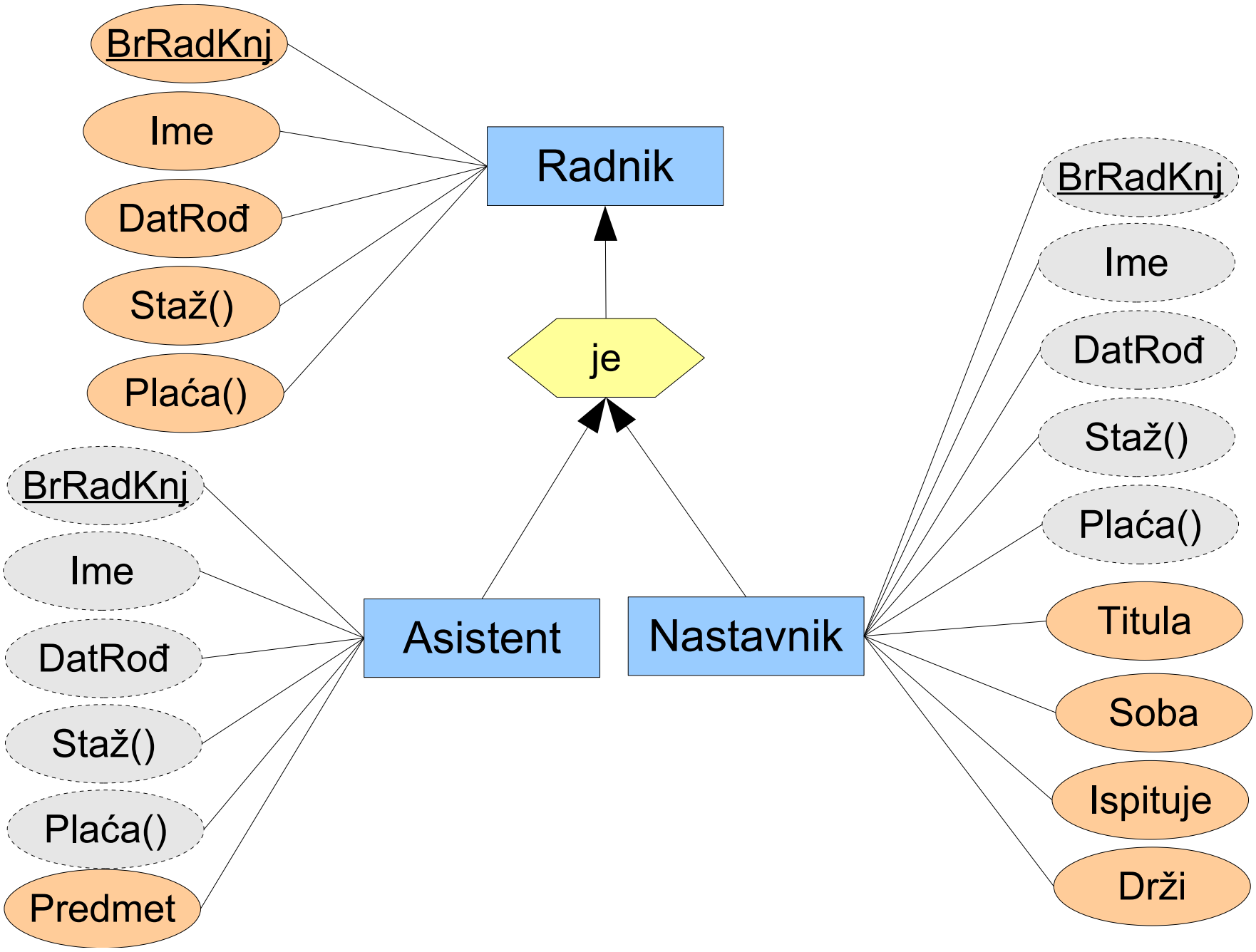
$$m = 11 - ((7 * (a + g) + 6 * (b + h) + 5 * (c + i) + 4 * (d + j) + 3 * (e + k) + 2 * (f + l)) \bmod 11)$$

- Ako je $m$ između 1 i 9, onda je K = m.
- Ako je $m = 10$, K se ne može izračunati, tj. neki od prvih 12 brojeva su pogrešni.
- Ako je $m = 11$, onda je K=0.

# Nasljeđivanje



# Nasljeđivanje atributa - ponovna upotrebljivost



# Objektne baze podataka - zaključne napomene

- Objektna orijentacija obećavajuća metodologija na polju programskih jezika i konceptualnog modeliranja (ljudi u svojoj okolini opažaju objekte, a ne tablice).
- Trenutačni nedostatak standardiziranog upitnog jezika glavna prepreka nedovoljno brzom širenju objektne paradigme u svijet kojim dominiraju relacijske baze podataka.
- Postoji veliki broj OODBMS-ova na tržištu (ObjectStore, Poet, Versant, Illustra, ...).
- Velik potencijal upravo kod kompleksnih podataka i operacija kakve su uobičajene u geoinformatici.
- Usvajanjem naprednih osobina objektno-orijentiranog modela, relacijske baze podataka postaju **objektno-relacijske baze podataka**.

# Funkcionalno proširenje relacijskog modela

- **Skupovni atributi:** dozvoljen atribut koji sadrži više istovrsnih podataka (ocjene studenta)
- **Definiranje tipova podataka** od strane korisnika
- **Reference na pojedine zapise** (umjesto stranih ključeva), npr. reference za predavanja za svakog studenta
- **Identitet objekata**
- **Nasljeđivanje**
- **Operacije** (u SQL se podacima ne mogu pridružiti operacije)

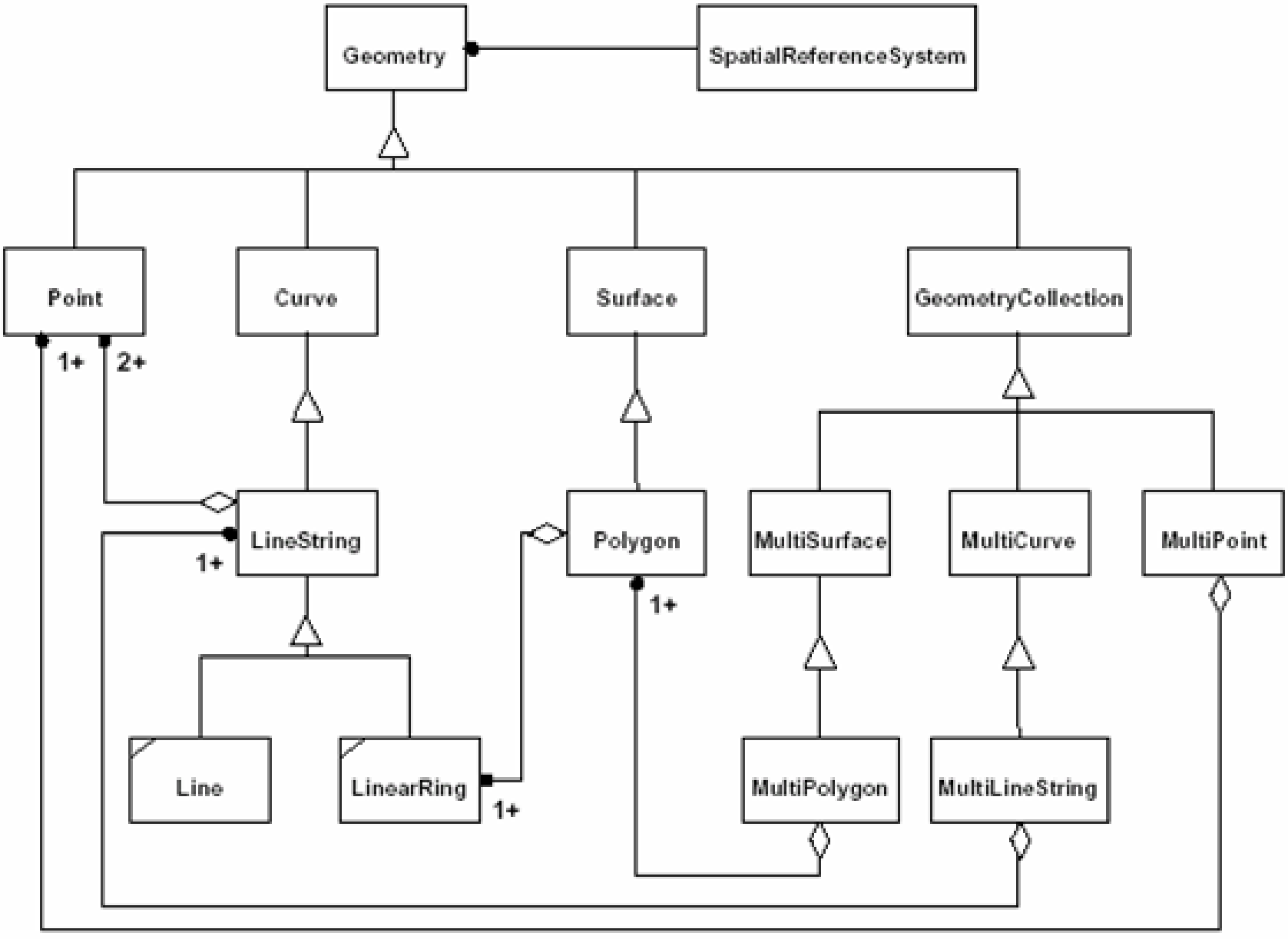
# Proširenja za prostorne podatke

- Operacije nad prostornim podacima u SQL-92 vrlo složene (površina poligona)
- Tipovi prostornih podataka: problem standarda (*Open Geodata Interoperability Specification*)
- Operacije nad prostornim podacima: problem kompleksnosti (ažuriranje baze) i efikasnosti (brzina obrade upita)
- Primjeri:
  - Oracle + Spatial (komercijalna baza podataka)
  - PostgreSQL + PostGIS (slobodna baza podataka, *open source*)

# Standardi za prostorne podatke

- OpenGIS: *Open Geodata Interoperability Specification* (Otvorena specifikacija za interoperabilnost geopodataka)
- UML (Unified Modeling Language)
- JDBC (Java Database Connectivity)
- OGC: Open Geospatial Consortium
- ISO: Međunarodna organizacija za standardizaciju
- ISO/TC 211 Geographic information / Geomatics

# UML-hijerarhija geometrijskih klasa (OGC)



# PostgreSQL: geometrijski tipovi podataka

| naziv (eng) | oblik | naziv (hr) |
|---|---|---|
| line | ((x1, y1), (x2, y2)) | pravac |
| lseg | ((x1, y1), (x2, y2)) | dužina |
| box | ((x1, y1), (x2, y2)) | pravokutnik |
| path | ((x1, y1), ...) | zatvorena polilinija |
| open path | [(x1, y1), ...] | otvorena polilinija |
| polygon | ((x1, y1), ...) | poligon |
| circle | < (x, y), r > | krug |

# PostgreSQL: operacije nad geometrijskim podacima

| simbol | naziv | specifikacija |
|---|---|---|
| $+$ | translacija pravokutnika | $box \times point \mapsto box$ |
| $*$ | promjena mjerila | $box \times integer \mapsto box$ |
| $\#$ | presjek dužina | $lseg \times lseg \mapsto point$ |
| $\#$ | broj vrhova poligona | $polygon \mapsto integer$ |
| && | preklapanje poligona | $polygon \times polygon \mapsto bool$ |
| $<<$ | da li je s lijeve strane? | $geom \times geom \mapsto bool$ |
| $>>$ | da li je s desne strane? | $geom \times geom \mapsto bool$ |
| $<<$ | da li je ispod? | $geom \times geom \mapsto bool$ |
| ?# | presjek | $path \times geom \mapsto bool$ |

# PostgreSQL - primjer geometrijskog upita

- Naći ceste koje prolaze općinom X:

```
SELECT  Cesta.naziv
FROM    Opcine, Ceste
WHERE   Opcine.opcinaID = 'X'
AND     Opcine.geometry ?# Ceste.geometry;
```

# Objektno-relacijske baze podataka - zaključne napomene

- Objektno-relacijski koncepti prisutni kod većine najnovijih verzija komercijalnih (Oracle 10g, Informix), i slobodnih baza podataka (Postgresql 8.0)
- Za geodetske i geoinformatičke primjene standardizacija prostornih podataka ključni je preduvjet efikasne primjene baza podataka u svakodnevnom radu.
- Napori ISO/TC211 i OGC-a vrijedni pažnje geodeta i geoinformatičara i prije samog službenog donošenja standarda.

# Baze znanja i logika

- Znanje nije jednostavno definirati.
- Znanje predstavlja sljedeću kariku u nizu:
  podaci $\longrightarrow$ informacije $\longrightarrow$ znanje
- Stara definicija: "istinito vjerovanje koje je moguće opravdati", tj. znanje je moguće potkrijepiti argumentom (npr. Zagreb je glavni grad Republike Hrvatske.).
- Koncept *vjerovanja* problematičan za računala.
- Razvoj **formalne logike** omogucio računalni pristup problematici izvođenja zaključaka iz poznatih činjenica.

# Činjenice i pravila

- **Činjenica** (eng. *fact*) u formalnoj logici je atomarna izjava koja se ne može dijeliti, tj. prikazati kao logička kombinacija jednostavnijih izjava.
- Činjenice su specifične, a ne općenite:
  - "Sokrat je smrtnik" je činjenica
  - "Svi ljudi su smrtnici" nije činjenica
- **Pravilo** (eng. *rule*) je uvjetna izjava koja uključuje *kvantificiranje*: $\forall$ "za svaki", $\exists$ "postoji"
- Logički simboli: $\Rightarrow$ "implikacija", $\wedge$ "i", $\vee$ "ili", $\neg$ "negacija".

# "Tradicionalni" pristup

- Činjenice su tip znanja kojim se bave tradicionalne baze podataka (npr. relacija GLAVNIGRAD bi imala polja GRAD i DRZAVA)
- Tradicionalne baze podataka se bave jednostavnim činjenicama, ali je vrlo teško definirati pravila tipa: "Svi letovi za Dubrovnik idu preko Zagreba".
- Kod baza prostornih podataka, važnih za geodeziju i geoinformatiku, velik broj informacija je implicitno pohranjen.

# Deduktivne baze podataka

- **Dedukcija**: izvođenje zaključaka iz premisa u logičkim sustavima.

## Primjer

- Svi ljudi su smrtni.
- Sokrat je čovjek.
- $\Rightarrow$ Sokrat je smrtnik

# Pravilo za presjedanje - zračni promet

- Neka je predikatom **let** zadan niz činjenica koji povezuje polazišta i odredišta u zračnom prometu: let(ZG,FR), let(ZG,ST), let(ZG,DU), let(ZG,PU).
- Ako želimo definirati predikat **jednoPresjedanje**, kojim želimo provjeriti da li se iz polazišta $X$ može stići u odredište $Y$ uz točno jedno presjedanje, logička definicija glasi:

$$\forall X \forall Y (jednoPresjedanje) \Leftrightarrow \exists Z (let(X, Z) \wedge let(Z, Y))$$

## Primjena pravila **jednoPresjedanje**

- jednoPresjedanje(ZG,ST) - No
- jednoPresjedanje(FR,ST) - Yes

# Usporedba relacijskih i deduktivnih baza podataka

- Relacijski model
  - SQL manipulira složenom strukturom podataka, čijim pretraživanjem dobivamo odgovor,
  - jezik za manipulaciju je odvojen od podataka.
- Deduktivni model
  - izračunava vrijednost logičkih izraza (istina ili neistina) ili traži vrijednosti za koje je logički izraz istinit,
  - činjenice i pravila su dio jezika, tj. nema razlike između podataka i jezika za njihovu manipulaciju.
- Problemi deduktivnog modela
  - osiguravanje trajnosti zapisa u deduktivnoj bazi
  - konstrukcija logičkog jezika koji ce imati mogućnosti prelaziti okvire radne memorije

# PROLOG

- "PROgramming with LOGic", logički sustav, odnedavno *open source*
- iznimno popularan alat u akademskoj zajednici, primjenjiv za rješenja brojnih logičkih problema, pa i onih vezanih uz prostorne podatke.
- `http://www.swi-prolog.org/`

## Primjer: obiteljsko stablo

- Činjenice: zensko, musko, roditelj
- Pravila: majka, otac, kcer, sin, baka, djed, ...
- `majka(X,Y):- zensko(X), roditelj(X,Y).`

# Deduktivne baze podataka - zaključne napomene

- Deduktivne baze podataka su obećavajuća tehnologija za baze podataka utemeljene na znanju.
- Kompaktan format zapisa pravila koja zamjenjuju tisuće redaka.
- Jedinstvenost podataka i jezika za manipulaciju.
- Problemi s trajnošću podataka.