BAZE PODATAKA 1 - UVOD

Prof Dr. Borivoje Milošević

- **UVOD**

- Od samog početka korišćenja računara, obrada različitih vrsta podataka, bila je jedan odosnovnih zadataka. Podaci i informacije su postali pokretačka snaga modernog poslovanja u celom svetu. Kada želimo da imamo kvalitetne informacije o svim segmentima našeg poslovnog ili čak i privatnog života najbolje je da na određeni način organizujemo sve podatke koje mogu da nam pruže informacije koje su od velike važnosti u trenutku kada su nam potrebne. Pogotovo se to odnosi na situacije kada u kratkom roku moramo doneti neku kvalitetnu ili sudbonosnu odluku.

- Tada bi bilo najbolje da podaci za svaki pojedini element budu organizovani tako da se mogu smestiti u tabele sa jedinstvenim zaglavljem. Može, a veoma često i mora da bude više tabela koje bi obuhvatile sve segmente našeg interesovanja. Svi ti segmenti se neretko zbog svoje prirode moraju organizovati u posebne tabele, a te tabele se mogu povezivati preko određenih zajedničkih elemenata. ***Skup više tih tabela koje služe jednom zajedničkom cilju, zajedno sa njihovim veznim elelmentima naziva se*** BAZOM PODATAKA. Njihov zajednički cilj se odnosi na svođenje veoma brze i uspešne infrormacije o svim događajima koji se dešavaju unutar jedne celine.

- *Motivacija*
- Da bi se stekla precizna slika o bazama

  podataka, nije dovoljno samo definisati

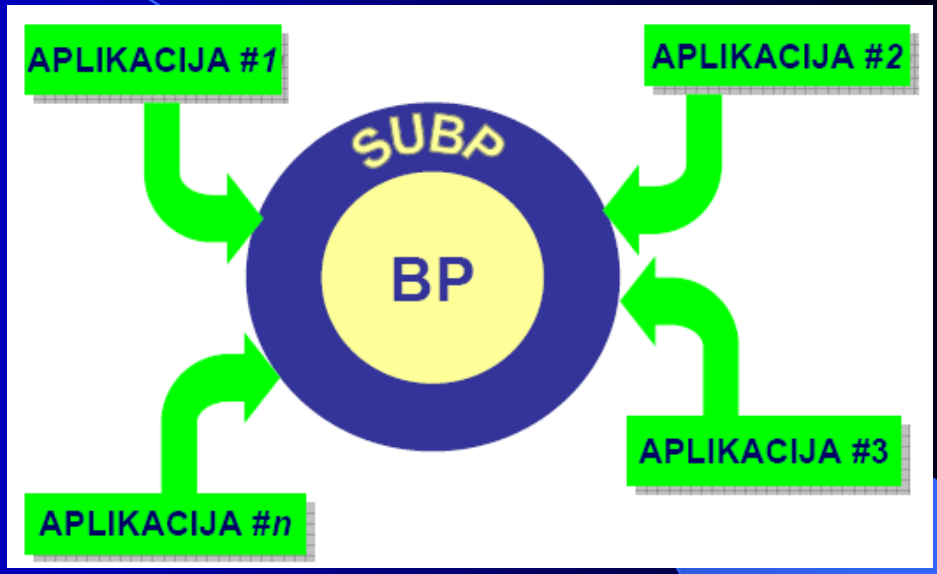
  pojam baze podataka
- Potrebno je prvo baze podataka sagledati u kontekstu njihovog istorijskog razvoja

Baza podataka je organizovana zbirka podataka. Termin je izvorno nastao u računarskoj industriji, a njegovo se značenje proširilo popularnom upotrebom toliko da Evropska direkcija za baze podataka (koja za baze podataka donosi prava za intelektualno vlasništvo) uključuje i neelektronske baze podataka unutar svoje definicije.

- Jedna od mogućih definicija baze podataka glasi da je to: ***Zbirka zapisa zapamćenih u računaru na sistemski način, takav da joj se računarski program može obratiti prilikom odgovaranja na problem*.**
- Svaki se zapis za bolji odgovor i razvrstavanje obično prepoznaje kao skup elemenata (činjenica) podataka. Predmeti vraćeni u odgovoru na upite postaju informacije koje se mogu koristiti za stvaranje odluka koje bi inače mogle biti mnogo teže ili nemoguće za stvaranje.

- Računarski program korišćen za upravljanje i ispitivanje baze podataka nazvan je ***sistem upravljanja bazom podataka*** (SUBP) ili DBMS (Data Base Management System). Svojstva i dizajn sistema baze podataka uključeni su u proučavanje informatičke nauke.

- Naziv *baza podataka* se strogo govoreći odnosi na zbirku zapisa, a na softver bi se trebalo odnositi kao na *sistem upravljanja bazom podataka* ili SUBP. Kada je kontekst jedinstven, mnogi administratori za baze podataka i programeri ipak koriste termin *baza podataka* da pokriju oba značenja.



Informacija – Definicija:

Informacija je rezultat obrade, manipulacije i organizacije podataka na način koji daje znanje korisniku. Drugim rečima, to je kontekst u kojem su podaci uzeti.

Značenja informacija

- 'Informacija' kao koncept ima mnoštvo značenja, od svakodnevnih pa do tehničkih upotreba. Opšte govoreći, koncept informacije je usko povezan sa notacijama ograničenja, komunikacije, upravljanja, podataka, oblika, instrukcije, znanja, značenja, mentalnog podražaja, uzroka, opažaja i predstavljanja.

Informacijsko doba

- Mnogi ljudi govore o informacijskom dobu kao uvodu za **doba znanja** ili društvo znanja, informacijskom društvu, informacijskoj tehnologiji pa iako su informatika, nauka o informaciji i računarstvo često u središtu pogleda, reč "informacija" je često korišćena bez obraćanja odgovarajuće pažnje na različita značenja koja je poprimila.

Informacija kao poruka

- S pojmom **informacija** susrećemo se u najraznovrsnijim situacijama, od upotrebe u svakodnevnom životu do one u specijalizovanim naučnim područjima. Ona predstavlja osnovno obeležje infromacionog doba, informacione nauke, tehnologije i društva. Od mnoštva značenja koja posjeduje, u ovom delu obrađen je onaj aspekt informacije koji je povezuje sa konceptom poruke. Budući da se podatak i informacija neretko koriste kao sinonimi, važno je napraviti distinkciju između njih. Naime, definicija informacije glasi: ***Da su to podaci stavljeni u značenje konteksta, dok je podatak izvan konteksta.***

- Drugim ričima, podatak je beskoristan sve dok ne prenosi neku informaciju. Prema sledećoj definiciji informacija je: *Skup znakova koji korisniku nešto znače, odnosno otkrivaju nešto novo.* Informacija je pojam sa mnogo značenja zavisno od konteksta, ali je kao pravilo usko povezana s konceptima kao što su značenje, znanje, percepcija, instrukcija, komunikacija i razni mentalni procesi. **Jednostavno rečeno, informacija je primljena i shvaćena poruka.** Ali pre svega, ona je rezultat procesiranja, manipulisanja i organizovanja podataka na načina da isti nadograđuju znanje osobe koja informaciju prima.

Informacioni sistem

- **Informacioni sistem** je u najopštijem smislu svaki sistem razvijen u cilju sakupljanja, kreiranja, čuvanja, obrade i interpretacije informacije. Informacijom se ovde smatra znanje ili dopuna znanja a predstavlja se podacima (uređene kolekcije simbola, znakova i signala). Informacioni sistemi se mogu bazirati i na samo pisanoj ili usmenoj komunikaciji, ali je verovatnije da će zahtevati upotrebu računara za upravljanje sakupljanjem, čuvanjem, obradom i distribucijom informacija. Informacioni sistem je uglavnom jedan od podsistema poslovnog sistema i njegove kategorije su:

- **1. Sistemi za obradu podataka** - vrše automatsku obradu velikog broja podataka, npr. kontrola inventara, obračuni bankovnih računa, popisi, itd.

- **2. Upravljački informacioni sistemi** - vrše automatsku analizu podataka i proizvode informacije korisniku za upravljanje poslom, npr. mesečna analiza prodaje, žalbe klijenata.

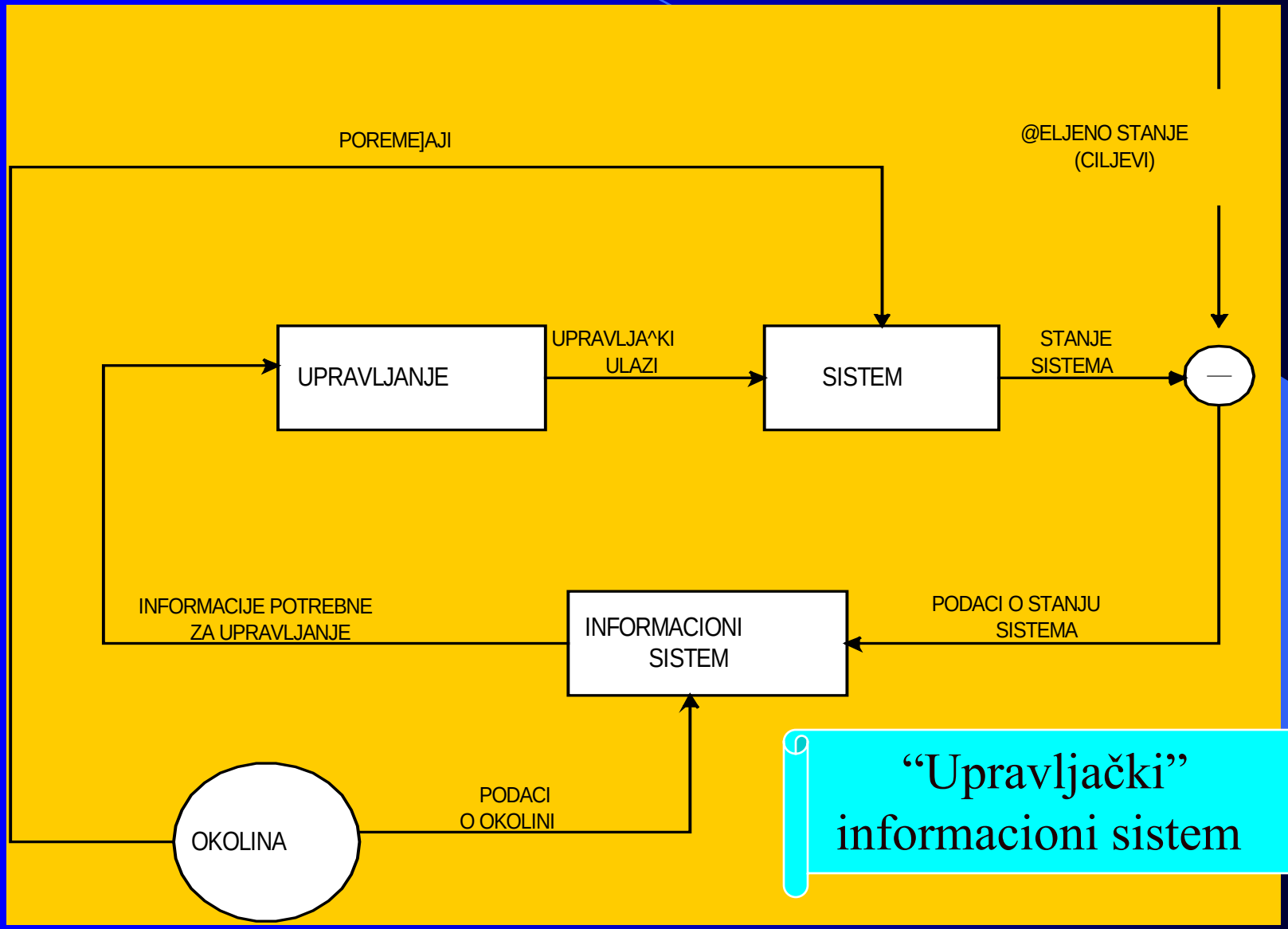
- **3. Sistemi za podršku odlučivanju** - vrše specijalne analize koje se tiču strateških problema odlučivanja u upravljanju, npr. dugoročno planiranje nabavke ili izgradnje fabrike ili izbora snabdevača sirovina sa najboljim balansom kvalitete - cena, pouzdanost snabdevanja.

Baze podataka i metodologija razvoja informacionih sistema

- Tehnologija baza podataka je znatno izmenila i unapredila metodologiju razvoja informacionih sistema (IS). Konvencionalne metode razvoja IS polaze od toga da je uloga IS da zatvori "povratnu spregu" u procesu upravljanja nekim realnim sistemom. Zato se terminu "informacioni sistem" obično i dodavao atribut "upravljački". Na osnovu ovakvog položaja IS u pocesu odlučivanja, problem projektovanja IS, u užem smislu, može se postaviti na sledeći način:

- Dati su podaci o stanju sistema i okolini.
- Date su informacije potrebne za upravljanje.
- Projektovati IS koji će podatke o stanju sistema i okolini transformisati u informacije potrebne za upravljanje.

Baze podataka i metodologija razvoja informacionih sistema



- Ovako postavljen zadatak, svodi se na projektovanje u užem (tehničkom) smislu, projektovanje i implementaciju programa na osnovu dobro specificiranih zahteva. Međutim, u praksi, niti su dati podaci o stanju sistema i okolini, niti informacije potrebne za upravljanje, pa se postavka problema projektovanja mora proširiti i obuhvatiti celokupan razvoj IS kroz sledeće opšte korake:

- ***Analiza i specifikacija zahteva korisnika (specifikacija podataka o stanju sistema, okolini i informacija potrebnih za upravljanje).***

- ***Projektovanje IS u užem smislu.***

Razvoj informacionog sistema

- Razvoj informacionog sistema (RIS)izvodi kroz četiri sledeće faze :

- **Aktivnost 1.** Funkcionalno modeliranje,
- **Aktivnost 2.** Informaciono modeliranje,
- **Aktivnost 3.** Aplikativno modeliranje i
- **Aktivnost 4.** Implementacija.

Funkcionalno modeliranje

treba da omogući postavljanje modela, tj.definisanje studije koja koncipira reinženjering poslovnih procesa u širinu.Ova aktivnost se definiše kroz tri podaktivnosti:

1. Funkcionalna dekompozicija	Polazi se od svesti o potrebi razvoja informacionog sistema, i to povezane, pre svega, sa donošenjem strategijske odluke rukovodećeg menadžmenta o sprovođenju reinženjeringa poslovnih procesa. Kao rezultat treba dobiti stablo poslovnih procesa kako ih vidi vodeći menadžment.
2. Definisanje zahteva korisnika	Treba da omogući da se analizom dokumenata i sprovođenjem intervjua može identifikovati okvir reinženjeringa poslovnih procesa.
3. Tehnički preduslovi	Rezultat ove faze rada su Tehnički preduslovi i "studija " ili "idejni projekat" kao dokument za dalju aktivnost.

Informaciono modeliranje

je ključni momenat gde do izražaja dolazesposobnost i znanje visokostručnog kadra iz oblasti menadžmenta i informatike. U ovoj fazi poželjno je angažovanje i spoljnih eksperata.Ova aktivnost se definiše kroz sledeće četiri podaktivnosti:

1. Definisanje detaljnih zahteva	Kao rezultat ovog rada treba da bude definisano detaljno stablo aktivnosti sa odgovarajućim detaljnim dekompozicionim dijagramima
2. Kreiranje ER modela	Ova aktivnost otvara "crnu kutiju", koja je budućim korisnicima uvek bila nepoznata, jer nisu mogli da prate razmišljanja projektanata informacionog sistema. Prvi put korisnici uzimaju aktivno učešće i u ovom delu i prvi put projektanti informacionog sistema crtajuono što predstavlja njihovo iskustvo i saznanje o poslovanju konkretnog preduzeća i što su oni osmislili u svojoj glavi.
3. Kreiranje atributa i	Treba da da opis osobina u prethodno definisanim entitetima. Osobine entiteta se definišu kroz identifikaciju atributa za svaki entitet,definisanje odgovarajućih ključeva i sprovođenja postupka normalizacije.
4. Definisanje poslovnih pravila	Predstavlja sintezu prethodne dve aktivnosti itreba da definiše poslovna ograničenja i pravila ponašanja.

Aplikativno modeliranje

posmatra se sa stanovišta izabranog sistemaza upravljanje bazama podataka (SUBP).Ova aktivnost se posmatra kroz sledeće tri podaktivnosti:

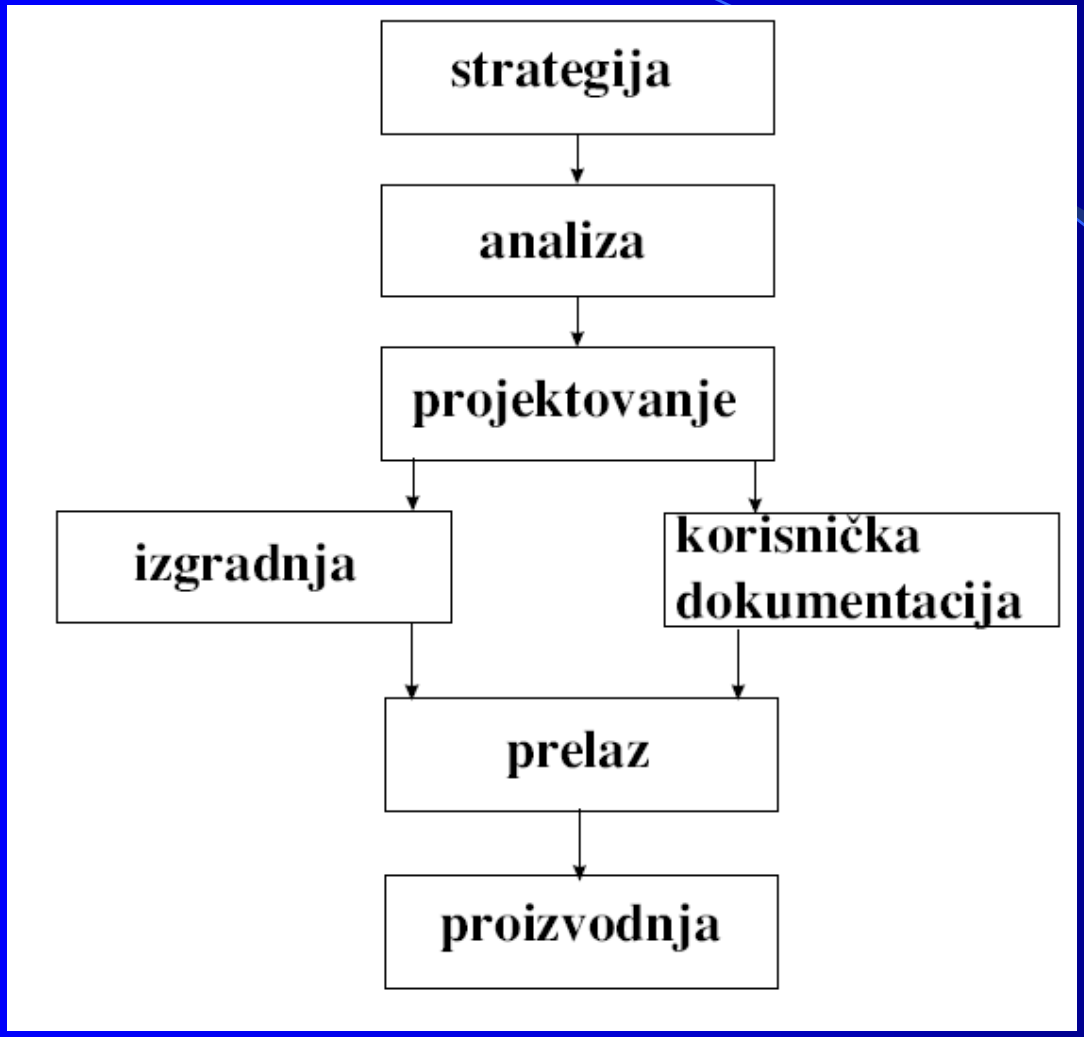
1. Definisanje fizičkog dizajna	Razmatra problematiku vezanu za izgradnjusistema za upravljanje bazama podataka (SUBP).
2. Generisanje šeme baze podataka	Definiše se za izabranu ciljnu platformu,gde se definišu fizičke tabele, kolone i relacije.
3. Izrada aplikacije	Treba da se realizuje korisnički pogled na podatke, tj. dase definišu meniji, forme, upiti i izveštaji.

Implementacija

omogućuje izvođenje promena vezanih za na čin rukovođenja i primene informacionih tehnologija.Ova aktivnost se posmatra kroz sledeće tri podaktivnosti:

1. Uvođenje	Treba da da ocenu urađene korisničke aplikacije, omogući izmene u toku uvođenja, izradi uputstva za korisnike i obuči same korisnike.
2. Testiranje	Treba da omogući testiranje sprovedenih aktivnosti u okviru postavljenog SUBP gde se ocenjuju performanse tog sistema.
3. Održavanje	Izvodi se kad se pređe u fazu eksploatacije novopostavljenog sistema. Ovde se drastično pokazuju sve manjkavosti i neregularnosti vezane za sprovedeni reinženjering poslovnih procesa i definišu odgovarajuće korektivne akcije.

APLIKACIJA=Šta utaditi+U kom domenu+Kako+Kojom veštinom



- *Šta uraditi:* To je pitanje na koje korisnik daje odgovor opisujući svoja očekivanja, način interakcije i različite aktere.
- *U kom domenu:* je pitanje koje treba da sadrži opis domena ili okruženja u kome aplikacija treba da egzistira i bitne po aplikaciju – elemente domena.
- *Kako:* dobija odgovor u fazi projektovanja – dizajna. Odgovor je rezultat projektantskog iskustva i veštine.
- *Kojom veštinom:* je pitanje o neophodnim znanjima i veštinama za izgradnju aplikacije.

1. Analiza informacionog sistema je proces u kome se opisuje funkcija postojećeg ili planiranog IS. Projektovanje IS je proces u kome se opisuje struktura planiranog sistema koja je najpogodnija za automatizovani rad.

2. Analiza IS počinje proceduralnim opisom transakcija tog sistema, a zatim se iz tog opisa izvodi opis podataka pojedinih transakcija sistema i globalna funkcija svake od transakcija sistema. Projektovanje IS polazi od globalne funkcije transakcija sistema uočenih u analizi sistema, koje zatim širi u praktično primenljivi i optimalni, sa gledišta posla, korisnika i cene sistema, opis podataka i procedura koje se implementiraju.

3. Analiza predstavlja opis mogućeg. Projektovanje predstavlja opis praktičnog. Pritom, procedure koje opisuju funkciju sistema, od kojih se polazi u analizi sistema, u slučaju većih IS, po pravilu se ne poklapaju sa procedurama sistema kojima završava proces projektovanja (zbog zahteva koje nalaže efikasnost i praktičnost realizacije).

4. Analiza sistema je pronalazački posao koji zahteva prikupljanje informacija, istraživanje, ispitivanje i uvid u sistem koji treba da se razvije. Projektovanje je kreativni proces koji na osnovu informacije pronađene u vreme analize, kreira strukturu novog informacionog sistema.

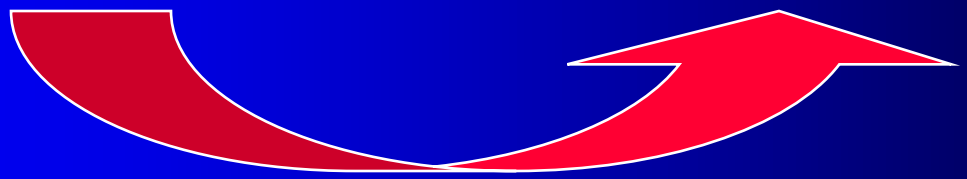
Linearni životni ciklus

1.1.1 Linearni životni ciklus:

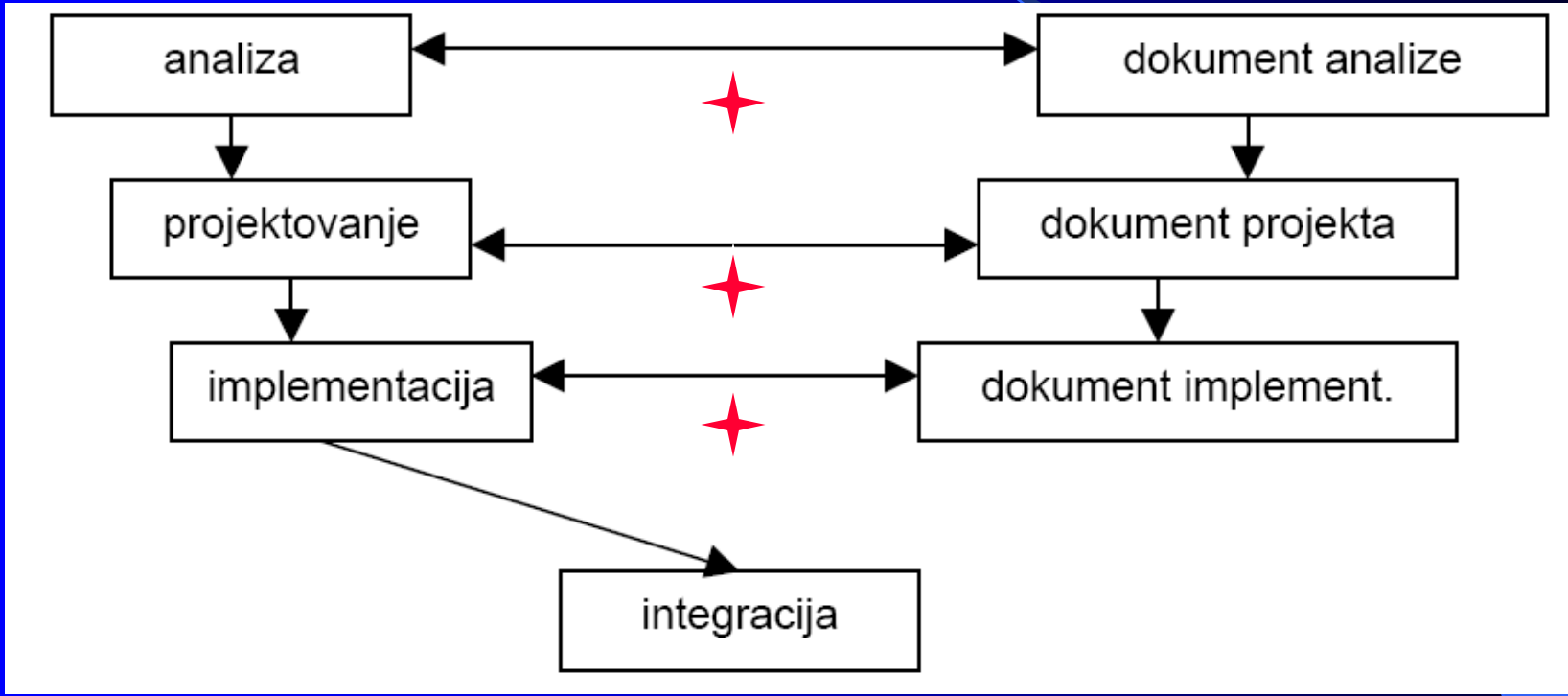
Se odnosi na pristup razvoju softvera koji je baziran na sukcesiji koraka, od zahteva do implementacije. Postoji nekoliko varijanti pristupa.

Model tunela : Ovo je ilustrativni način da se izrazi nepostojanje modela razvoja. U projektima sa tunelskim pristupom nemoguće je znati šta se dešava. Razvoj napreduje, ljudi rade - često vrlo naporno, ali nema pouzdane informacije o tome kako softver napreduje ili kakvog su kvaliteta razvijeni elementi. Ovaj model razvoja odgovara samo malim projektima sa vrlo ograničenim brojem učesnika.

Model vodopada : Ovaj model podržava opšti opis aktivnosti u razvoju softvera. Prema ovom modelu, razvoj softvera je niz faza povezanih u linearno izvršenje, od specifikacije i analiza zahteva do isporuke proizvoda. Svaka faza odgovara jednoj aktivnosti. Ovakav razvoj je praćen izradom dokumentacije koja predstavlja podršku za validaciju svake faze.



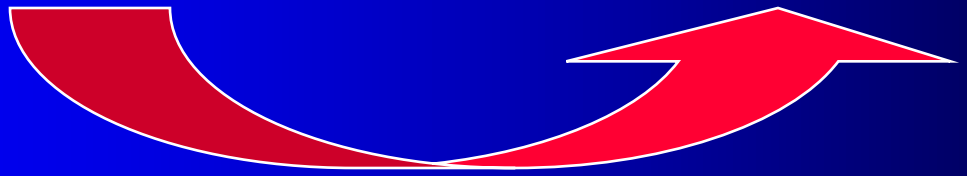
Linearni životni ciklus



Iterativni životni ciklus

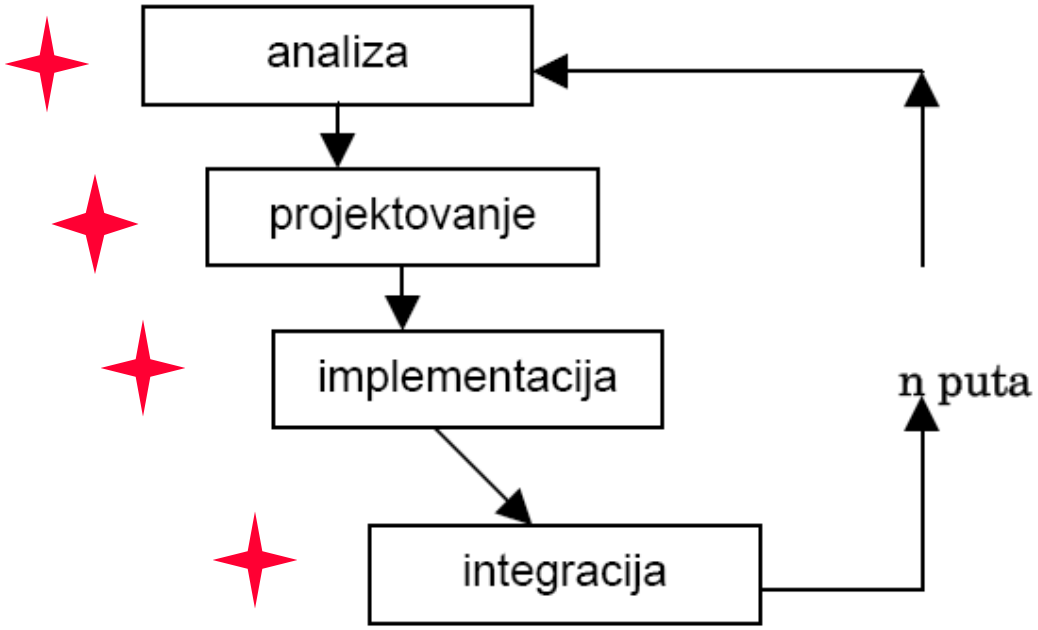
1.1.2 Iterativni životni ciklus:

Se bazira na jednostavnoj ideji: kada je sistem suviše složen da se razume, projektuje i implementira u jednom pokušaju, bolje je implementirati ga iterativno, evolucijom. I u prirodi su funkcionalno složeni sistemi uvek evoluirali od jednostavnijih sistema. Implementacija ove ideje u razvoju softvera nije uvek jednostavna. Softver je osetljiv na promene i modifikacije. Softversko inženjerstvo nas uči da je neophodno, da bi softver bio robustan:



Iterativni životni ciklus

- segmentirati prostor mogućih stanja
- redukovati kopčanje korišćenjem nivoa apstrakcije (strukturno projektovanje)
- razdvojiti specifikaciju od implementacije.



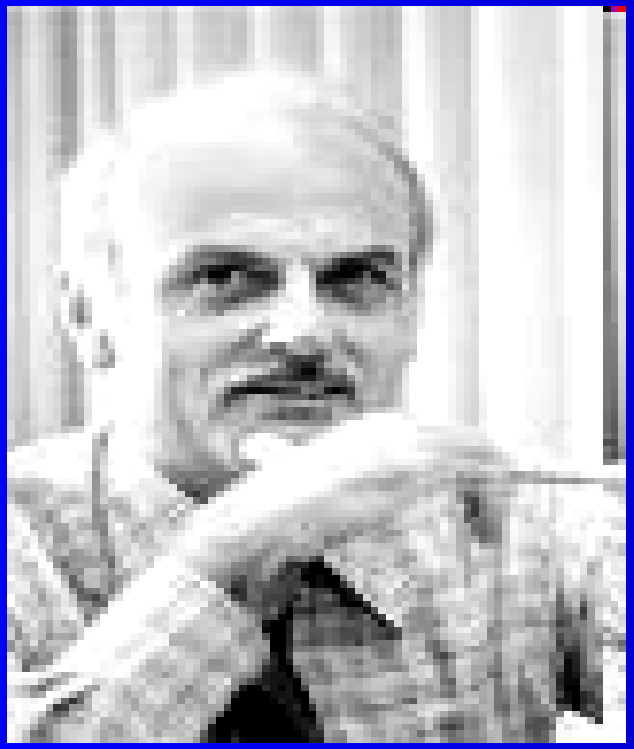
ISTORIJAT BAZA PODATAKA:

- Najranija poznata upotreba termina **baza podataka** potiče iz 1963. god. kada je Društvo za razvoj sistema uzelo pod pokroviteljstvo simpozijum pod naslovom *Razvoj i upravljanje računarsko centriranom bazom podataka*. **Baza podataka** kao jedinstvena reč postala je uobičajena u Evropi u ranim 1970-ima, a krajem decenije koristila se u glavnim američkim novinama. (**Banka podataka, sinonimni termin, koristio se vrlo rano u novinama Washington Post, 1966.)**

ISTORIJAT BAZA PODATAKA:

- Prvi sistemi upravljanja bazom podataka razvijeni su u 1960-ima. Začetnik u tom polju bio je Charles Bachman. Bachmanovi rani radovi pokazuju da je njegov cilj bio stvaranje delotvornije upotrebe novih uređaja s trenutnim pristupom memoriji koji su postali dostupni: do tada se obrada podataka temeljila na bušenim karticama i magnetskoj traci, pa je tako serijska obrada bila dominantna aktivnost. Dva su se ključna modela podataka pojavila u to vreme: **CODASYL** je razvio ***mrežni model*** baziran na Bachmanovim idejama, pa se (očigledno nezavisno) ***hijerarhijski model*** koristio u sistem koji je razvio North American Rockwell, a koga je kasnije prihvatio IBM kao kamen temeljac svoje proizvode.

ISTORIJAT BAZA PODATAKA:



Relacioni model je predložio E. F. Codd 1970. godine. On je kritikovao postojeće modele zbog zbrke apstraktnih opisa informacijskih struktura sa opisima mehanizama fizičkog pristupa. Ipak je dugo vremena relacioni model ostao samo u području akademskog interesa. Dok su CODASYL sistemi i SUI bili zamišljeni kao rešenja praktičnog inženjerstva, uzimajući u obzir tehnologiju koja je postojala u ono vreme, relacioni model je zauzeo mnogo veću teoretsku perspektivu, smatrajući (ispravno) da će hardverska i softverska tehnologija uhvatiti korak s vremenom. Među prvim prototipovima bili su Stonebrakerov Ingres na Berkeley-u, te projekt Sistem R u IBM-u. Oba navedena su bili istraživački prototipovi objavljeni tokom 1976. Prvi komercijalni proizvodi, Oracle i DB2, nisu se pojavili sve do oko 1980.

ISTORIJAT BAZA PODATAKA:

- Tokom 1980-ih istraživačka aktivnost se usredsredila na sisteme distribuiranih baza podataka i na sisteme baza podataka, međutim taj je napredak imao mali učinak na tržište. Druga važna teoretska zamisao bio je funkcionalni model podataka, ali bez obzira na neke specijalizovane primene u genetici, molekularnoj biologiji itd. svet nije na njega obratio veliku pažnju.

- U 1990-im pažnja se prebacila na baze podataka orijentisane prema objektu. To je izazvalo nekakav uspeh u poljima gdje je bilo potrebno rukovati kompleksnijim podacima nego što bi se mogli komotno nositi opisani sistemi: prostorne baze podataka, inženjerski podaci (uključujući odlagališta softverskog inženjerstva), multimedijski podaci. Neke od tih ideja prihvatili su komercijalisti, koji su kao posledicu integrisali nove osobine u svoje proizvode.

- U 2000-im pomodno područje za inovacije postale su XML baze podataka. XML baze podataka pokušavaju ukoniti tradicionalnu podelu između dokumenata i podataka, dopuštajući svim organizacijskim informacijskim resursima da se drže na jednom mestu bez obzira da li su visoko strukturirani ili ne.

PRETEČA BAZA - Klasična organizacija datoteka

- IS je sačinjavao skup nezavisnih aplikacija
- Svaka aplikacija - sopstvene datoteke
  "Skladište podataka" - skup datoteka
- Podaci o istom entitetu u različitim
- datotekama
- Vremenom, takav IS dolazi u
  kontradikciju sa samim sobom

Klasična organizacija datoteka

5.

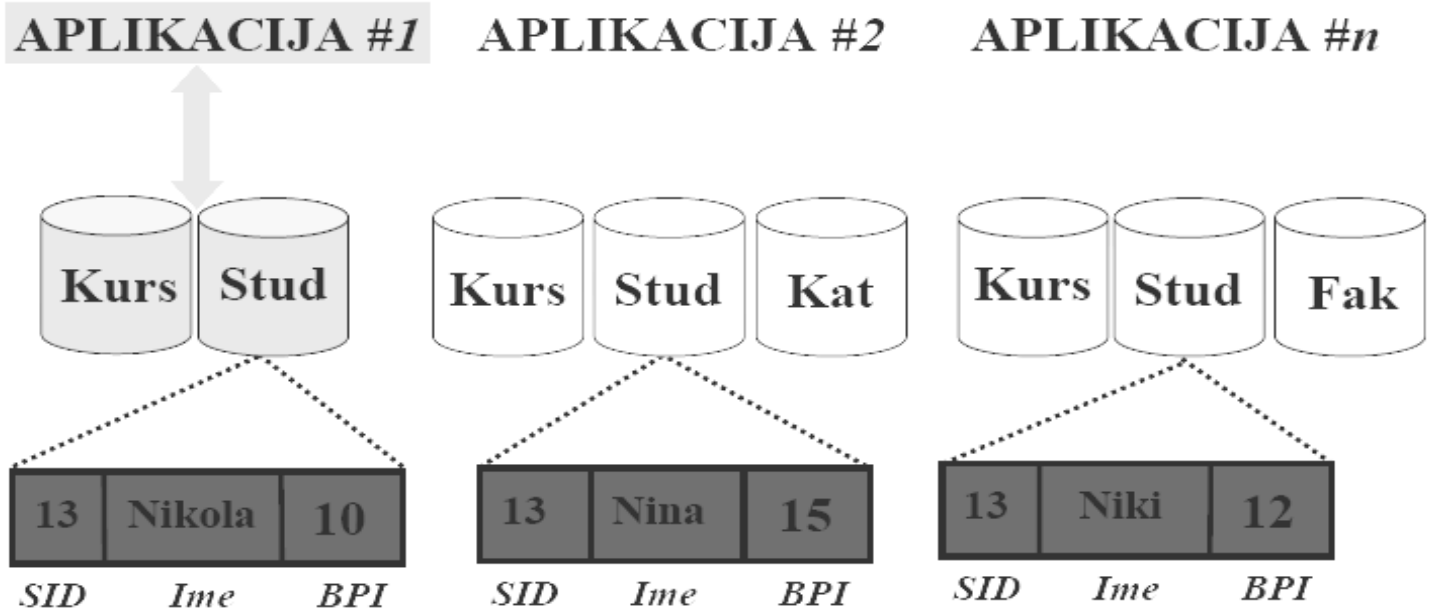
Klasična organizacija datoteka

- Osnovni nedostaci
  - nepovezanost aplikacija
  - redundantnost podataka
  - čvrsta povezanost programa i podataka
    - program vodi računa o FSP datoteke, kako u opisu, tako i u proceduri
- Posledice
  - otežano održavanje IS-a
  - otežan dalji razvoj IS-a

6.

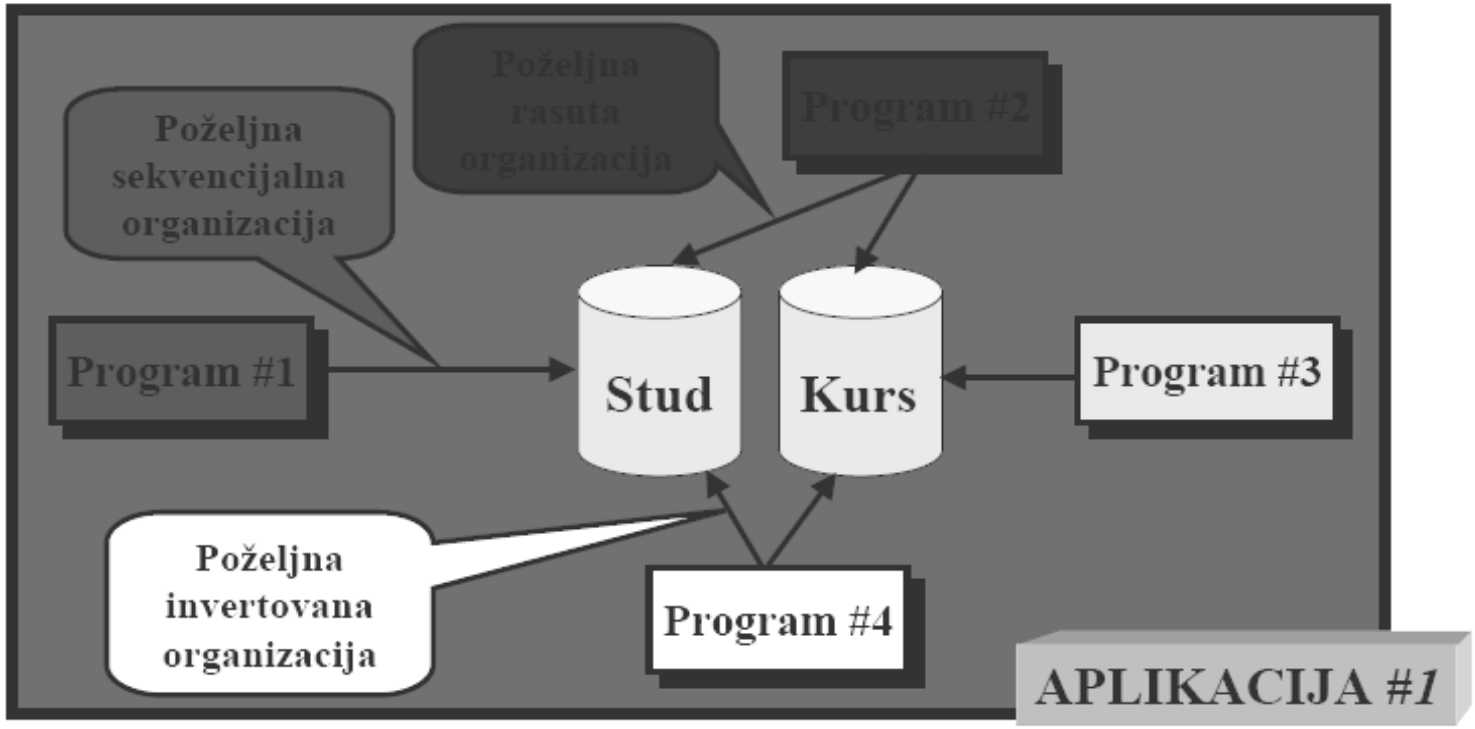
Klasična organizacija datoteka

- Primer - nepovezanost i redundantnost



Klasična organizacija datoteka

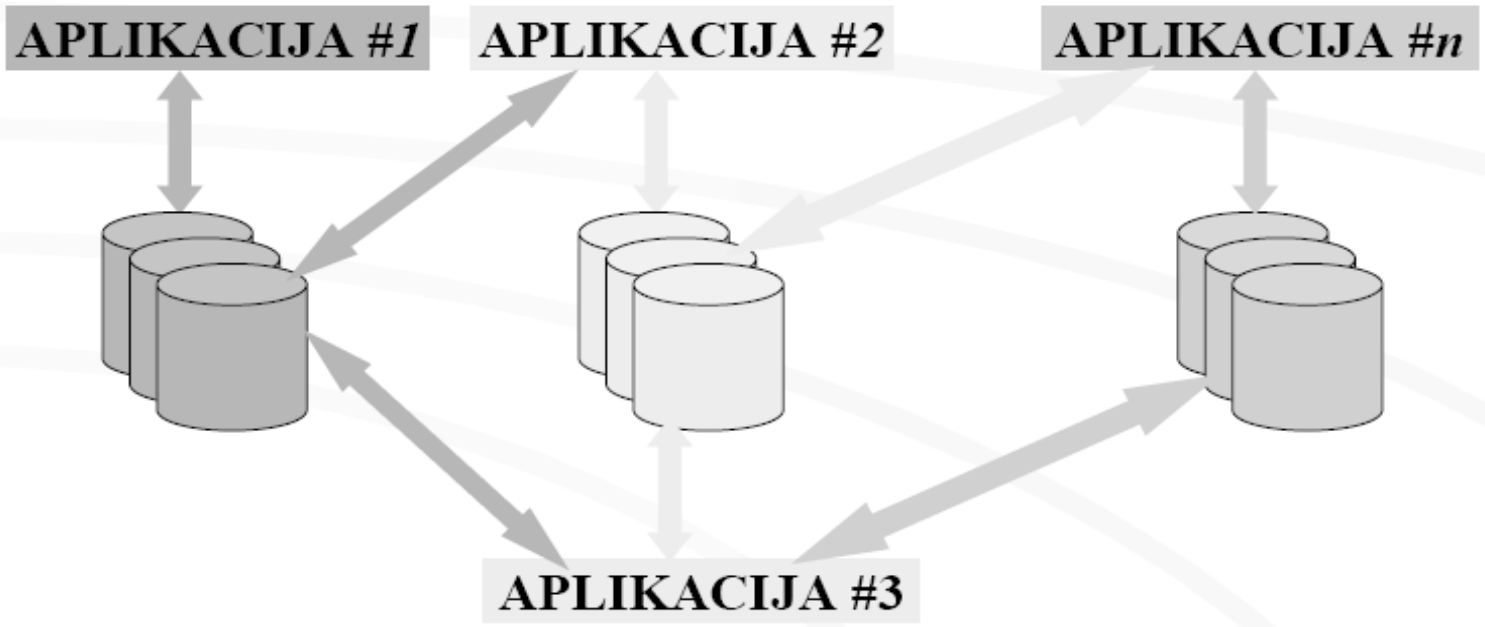
- Čvrsta povezanost programa i podataka

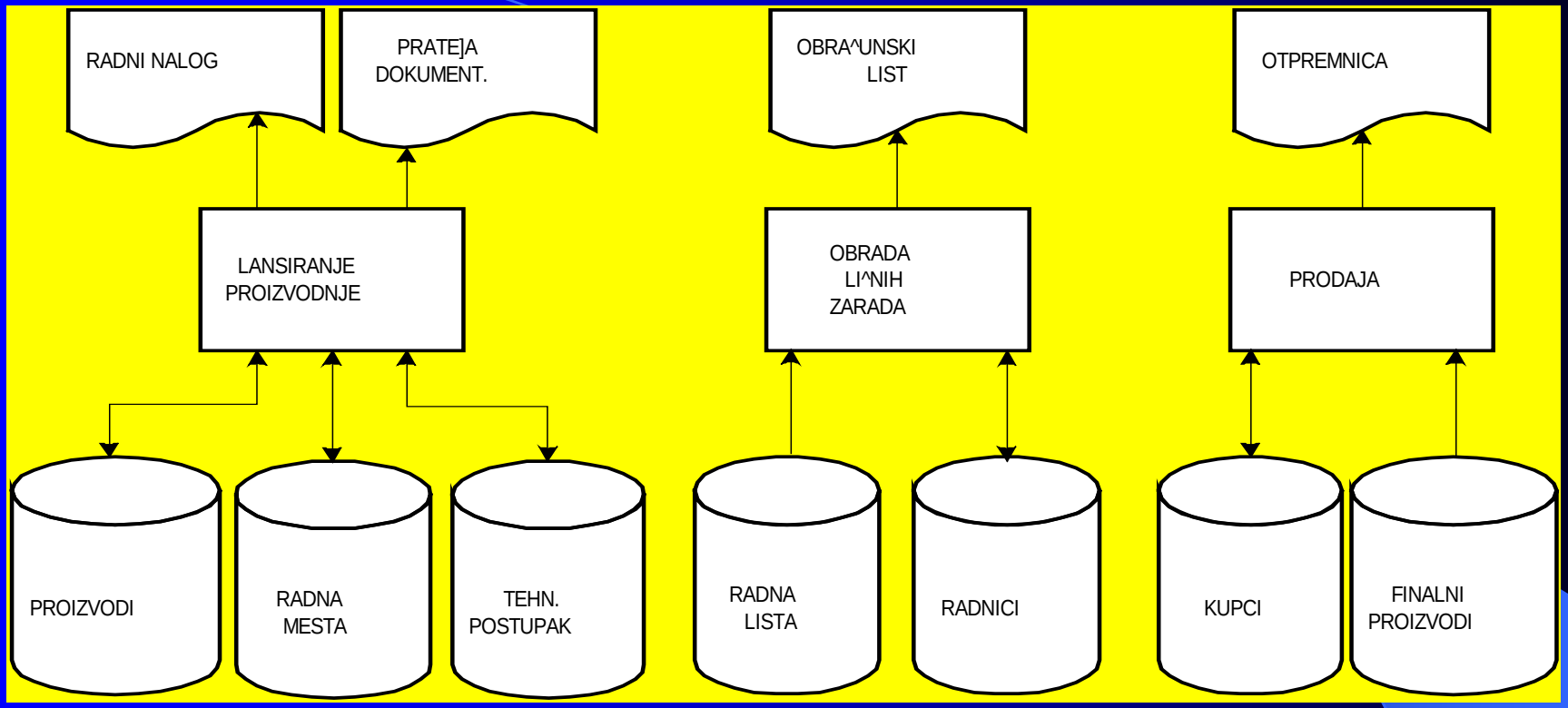


Klasična organizacija datoteka

- Problemi koji se mogu ublažiti, ili čak i razrešiti i u klasičnoj organizaciji
  - nepovezanost aplikacija
  - redundantnost
- Problem koji je gotovo nemoguće ublažiti ili razrešiti u klasičnoj organizaciji
  - čvrsta povezanost programa i podataka

Klasična organizacija datoteka





“Sistemski dijagram" jednog skupa "aplikacija" u nekom informacionom sistemu.

- Svaka aplikacija je razvijana posebno
- Koristi svoje "privatne" datoteke sa fizičkom strukturom podataka pogodnom za tu aplikaciju.

Osnovni nedostaci ovakve obrade podataka su:

- *Redundansa podataka*, odnosno *višestruko pamćenje* istih podataka je neminovno. Na primer, isti podaci o proizvodima se, u nekoj proizvodnoj radnoj organizaciji pamte i nekoliko desetina puta, ili isti podaci o građanima u nekom komunalnom informacionom sistemu i stotinu puta. Višestruko skladištenje istih podataka dovodi do nepremostivih problema pri njihovom ažuriranju. Kada se neki podatak promeni, to se mora učiniti na svim mestima na kojima se on čuva. To, ne samo da značajno povećava troškove obrade podataka, nego je organizaciono praktično neizvodljivo, pa se, u raznim izveštajima, pojavljuju različite verzije istog podatka.

Osnovni nedostaci ovakve obrade podataka su:

- ***Zavisnost programa od organizacije podataka***. Programi su zavisni i od logičke i od fizičke strukture podataka. Fizička struktura podataka definiše način memorisanja podataka na spoljnim memorijama.

  Logička struktura je struktura podataka koja je predstavljena programeru.

  U klasičnim datotekama razlika fizičke i logičke strukture podataka je mala. Zavisnost programa od logičke strukture se ogleda u tome što program zavisi, na primer, od naziva i redosleda polja u zapisu, što ubacivanje novog polja u zapis ili bilo kakvo drugo restruktuiranje zapisa, koje ne menja sadržaj podataka koje program koristi, ipak zahteva i izmenu samog programa. Fizička zavisnost se ogleda u tome što program zavisi od izabrane fizičke organizacije datoteke i izabrane metode pristupa, načina sortiranja i slično.

Osnovni nedostaci ovakve obrade podataka su:

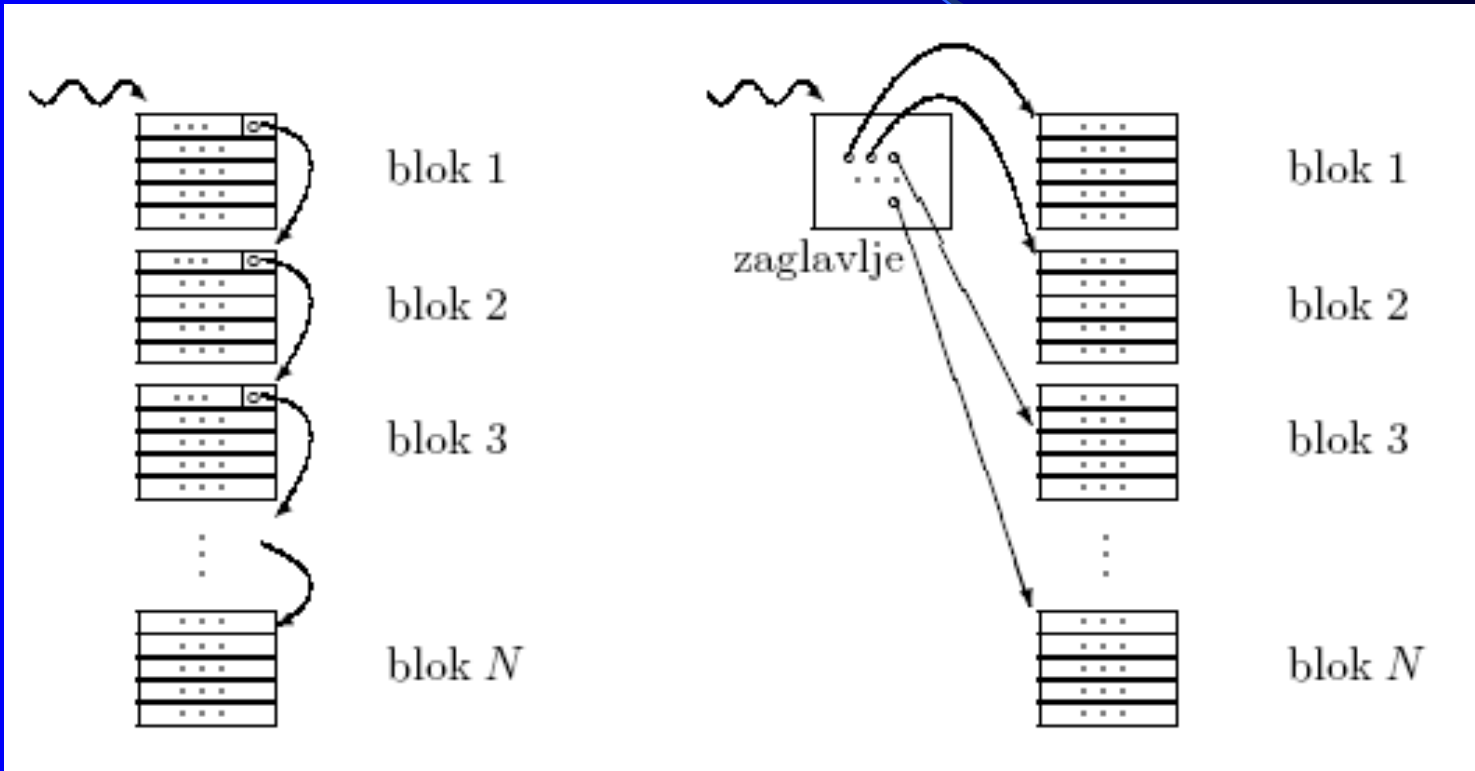
- U osnovi i logička i fizička organizacija podataka su prilagođene konkretnom programu. **Novi zahtevi za informacijama, iz podataka koji već postoje u datotekama, mogli bi zahtevati izmenu fizičke i logičke strukture podataka, a to bi zahtevalo izmene u ranije razvijenim programima.** Umesto toga, obično se za novi program stvara nova datoteka, a to dalje povećava redundansu podataka.

Osnovni nedostaci ovakve obrade podataka su:

- *Niska produktivnost u razvoju informacionog sistema (IS)*. Struktuiranje podataka u nezavisan skup datoteka je jedan od uzroka veoma niske produktivnosti u razvoju IS. Na primer, čak i kada postoje svi podaci koji se u nekoj aplikaciji zahtevaju, ali se oni nalaze u različitim datotekama, na raznim medijumima, sa različitim fizičkim organizacijama, zadovoljenje i nekog jednostavnog informacionog zahteva iziskuje značajne programerske napore. Nad strukturom podataka predstavljenom velikim brojem nezavisnih datoteka veoma je teško razviti softverske alate za brz i visoko produktivan razvoj aplikacija i za neposrednu komunikaciju korisnika sa sistemom.

Jednostavna – sekvencijalna datoteka

Zapravo se radi o odsustvu bilo kakve organizacije. Zapise datoteke ređamo u onoliko blokova koliko je potrebno. Blokovi koji čine datoteku mogu biti povezani u vezanu listu (svaki blok sadrži adresu idućeg bloka) ili može postojati tablica adresa svih blokova.



Jednostavna datoteka (dve varijante)

Jednostavna datoteka

- Traženje zapisa sa zadanom vrednošću ključa zahteva sekvencijalno čitanje cele datoteke (ili bar pola datoteke u proeku), sve dok ne nađemo na traženi zapis. Ako je datoteka velika, moramo učitati mnogo blokova, pa pristup traje dugo. Da bi ubacili novi zapis, stavljamo ga na prvo slobodno mesto u prvom nepopunjenom bloku, ili priključujemo novi blok ukoliko su svi postojeći blokovi popunjeni.

Baze podataka i sistemi za upravljanje bazama podataka

- Osnovne ideje:
  - da se svi podaci jednog IS integrišu u jednu veliku "datoteku" - bazu podataka
  - nereduntantno memorisanje podataka
  - da svi programi koriste podatke iz baze podataka, ili je ažuriraju koristeći usluge posebnog softverskog proizvoda - sistema za upravljanje bazama podataka

Tipovi baza podataka

- Objektno-relacione baze podataka
- Objektno orijentisane baze podataka
- XML baze podataka
- Distribuirane baze podataka
- Aktivne baze podataka
- Prostorne baze podataka
- Deduktivne baze podataka
- Data warehose
- Data mining
- Mobilne baze podataka
- Embedded baze podataka

Baze otvorenog koda

- MySQL
- PostgreSQL
- Cloudscape
- Firebird
- HSQLDB
- Ingres
- MaxDB
- MonetDB
- SQLite
- tdbengine

Komercijalne baze podataka

Oracle
IBM DB2
Informix
InterBase
Progress
Microsoft SQL Server
DBase
Foxpro
InterBase

SAVREMENE BAZE PODATAKA

Mainframe

- Do nedavno su se velike baze podataka mogle pokretati samo na mainframe kompjuterima. Do kraja 80-tih i početka 90-tih godina 20. veka obrada podataka na mejnfrejm kompjuterima je bio jedini izbor za organizacije kojima je bio potrebano obimno procesiranje podataka kao i podrška za veliki broj korisnika. Mejnfrejmovi su bili u upotrebi preko 20 godina a glavni razlog za njihovu dugovečnost je bila njihova pouzdanost. Sposobnost mejnfrejmova da istovremeno podržavaju veliki broj korisnika dok istovremeno vrši brzo prikupljanje informacija i njihovo prosleđivanje doprineo je njegovoj prihvaćenosti u oblasti velikih firmi i korporacija.
- Obrada podataka na mejnfrejmovima je označivala da se sva procesiranja obavljaju na glavnom računaru – mejnfrejmu. Mejnfrejm je zadužen za:

  - pokretanje sistema za upravljanje relacionim bazama podataka (RDBMS),

  -da upravlja aplikacijama koje pristupaju sistemu za upravljanje relacionom bazom podataka i

  -da održava komunikacije između mejnfrejma i udaljenih terminala.

  Inteligentni terminal je kao što mu i samo ime implicira bio ograničen na prikazivanje teksta i prihvatanje podataka koje je korisnik na njemu unosio.

SAVREMENE BAZE PODATAKA

- Najveća mana obrade podataka na mejnfrejmovima je bila to što su bili veoma skupi.

- Kupovina i održavanje mejnfrejmova je mogla da košta i po nekoliko miliona dolara. Mejnfrejmovi su skupi za korišćenje jer još zahtevaju posebne uslove instalacije (posebne prostorije sa posebnim napajenjem i kontrolom temperature) , zahtevao je ekstenzivnu podršku i nisu koristili uobičajene kompjuterske komponente. Mejnfrejmovi su , umesto korišćenja standardnih kompjuterskih komponenata, koristili hardver i softver koji je proizvodio sam proizvođač kompjutera. Ovakav pristup proizvođača je ograničavao kupca kompjutera na relativno ograničen skup dodatnog hardvera kao i softverskih rešenja.

SAVREMENE BAZE PODATAKA

PC/file server:

- PC/File server je postao popularan krajem 80-tih godina 20 veka kada su poslovni korisnici počeli da se okreću PC računarima kao alternativu mejnfrejmovima.
- Korisnicima se dopalo to što su sada mogli sami i sa lakoćom da razvijaju svoje aplikacije kroz četvrtu generaciju programskih jezika (4GL).
- Jeftin i lak za korišćenje softver kao npr. Lotus 1-2-3, dBASE i Word Perfect je omogućio zaposlenima i kućnim korisnicima da kreiraju dokumenta i upotrebljavaju baze podataka brzo i precizno.
- Obrada podataka PC/File serverom je podrazumevala pokretanje i aplikacije RDBMS-a na samom PC računaru. Druga veoma važna tehnologija je bila lokalna mreža (LAN) i njena integracija u firmama širom sveta. Radilo se dakle o relacionoj bazi podataka u heterogenom mrežnom okruženju. Iako su korisnici bili naviknuti na terminalske konekcije ka firminom mainframe-u , sada su fajlovi koji su se obrađivali mogli da sačuvaju na lokalnom kompjuteru i da im se pristupa sa nekog drugog kompjutera priključenog na istu mrežu. Na PC-u se izvršavalo svo RDBMS procesiranje dok se fajl server koristio kao centralizovana oblast za skladištenje podataka. Nakon što je Apple Macintosh uveo grafički interfejs (Graphic User Interface – GUI) kompjuteri su pored jeftinosti i solidne procesorske snage postali i laki za korišćenje.

SAVREMENE BAZE PODATAKA

- Glavna negativna strana PC/File server procesiranja je što se svo procesiranje baze podataka izvršavalo na PC računaru. Kada bi se poslao upit fajl serveru on ne bi procesirao upit *već bi vratio sve podatke* koji su potrebni za obrađivanje tog upita. To je dovodilo do umanjenja performansi i drastičnom povećanju zagušenja mreže.
- Tokom ovog vremena brzih promena i napretka pojavio se novi tip sistema. Nazvan je Client/Server i može se reći da je on odgovor na negativne strane obrade podataka na mejnfrejmovima i PC/File serverima. Kombinovanjem obrađivačke snage mejnfrejma sa fleksibilnošću i cenom PC-a, obrada podataka metodom Client/Server je spojila sve prednosti prethodnih tehnologija.

SAVREMENE BAZE PODATAKA

Client/Server

- Client/Server obrada se može definisati kao logička podela korisničkog interfejsa i upravljanje bazom podataka.
- Client kompjuter, često nazivan i radna stanica, kontroliše korisnički interfejs. Na njemu se korisniku predstavlja informacija u slici i tekstu i na njemu takođe korisnik unosi podatke.
- Serverski kompjuter se stara o upravljanju bazom podataka. Na njemu se vrši manipulacija podacima, smeštaju svi podaci, i sa njega se prosleđuju podtrebni podaci određenom korisniku. Sve obrade nad podacima se obavljaju na ovom računaru.

SAVREMENE BAZE PODATAKA

Client/Server metoda obrade informacija je postala popularna zbog :

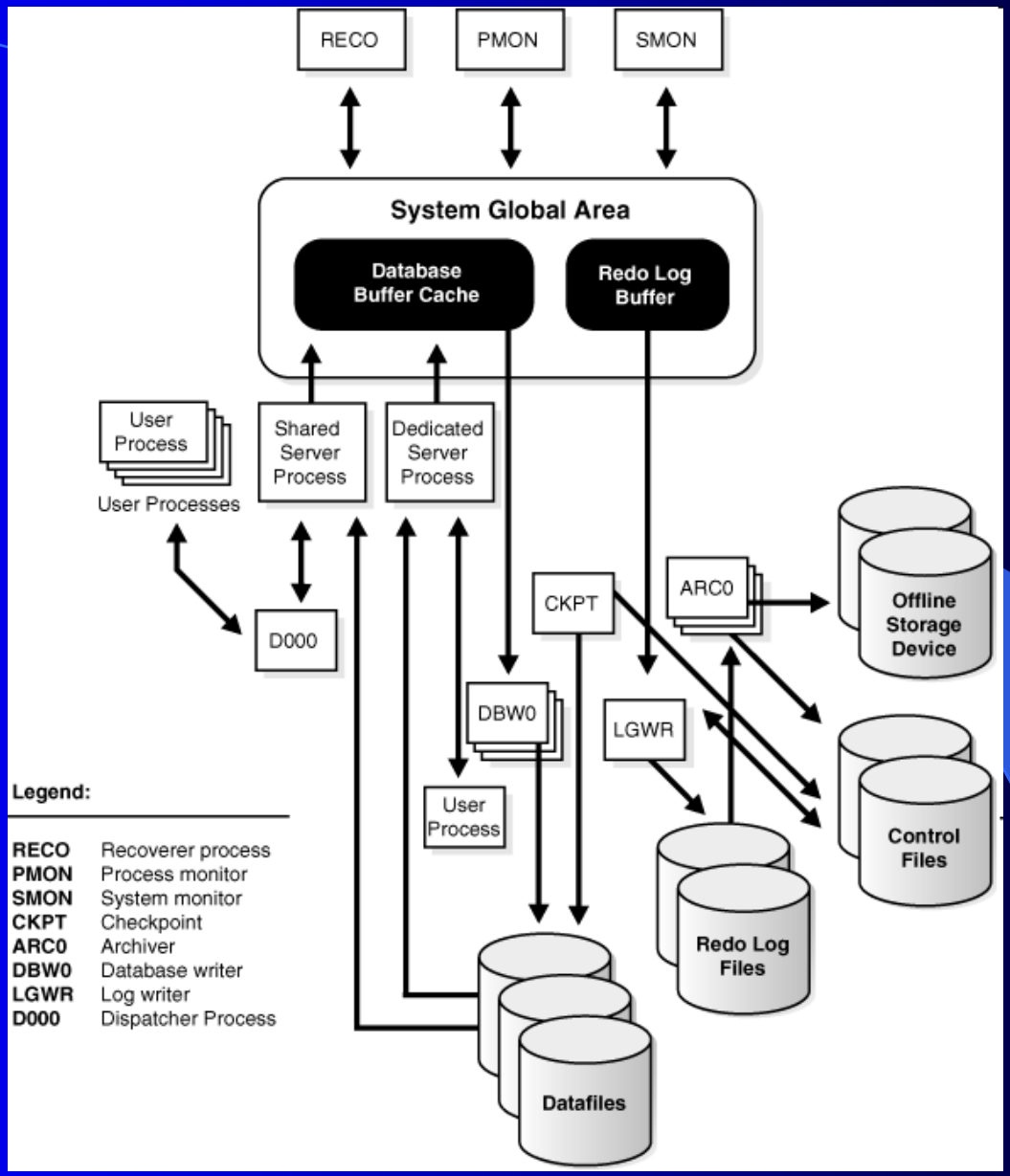
- Pristupnost – obrada podataka ovom metodom je drastično jeftinija od metode mejnfrejma. Client/Server je baziran na *otvorenoj arhitekturi* što omogućava da više proizvođača proizvodi konkurentne proizvode i time da snizi cenu za razliku od mejnfrejmova čije je elemente proizvodio proizvođač samog mejnfrejma i time diktirao cene. Client/Server je uglavnom baziran na PC računarima pa je i drastičnan pad cena računara doveo do dalje pristupačnosti celokupnog sistema.
- Brzina – razdvajanje obrade informacija između klijenta i servera smanjila je zagušenje mreže što je dovelo da *klijent/server ima performanse jednog mejnfrejma.*
- Prilagodljivost – klijent/server arhitektura je daleko otvorenija od mejnfrejm arhitekture. To je omogućilo stvaranje jednog sistema kod koga je *softver nabavljen od jednog* proizvođača sistema za upravljanje bazama podataka, *hardver od drugog* proizvođača a *softver za razvoj aplikacija od trećeg* proizvođača. Tako sada kupci mogu da odaberu komponente koje im najviše odgovaraju.
- Pojednostavljen pristup podacima – Klijent/Server povećava dostupnost podataka masama korisnika. Sa njim pristup podacima nije ograničen na one koje razumeju proceduralno programiranje. Naprotiv pristup podacima je obezbeđen softverskim paketima koji prikrivaju svu kompleksnost pristupa podacima. Softver za obradu teksta, rad sa tabelama su samo primeri uobičajenih softverskih paketa koji omogućavaju lak pristup podacima.

SAVREMENE BAZE PODATAKA

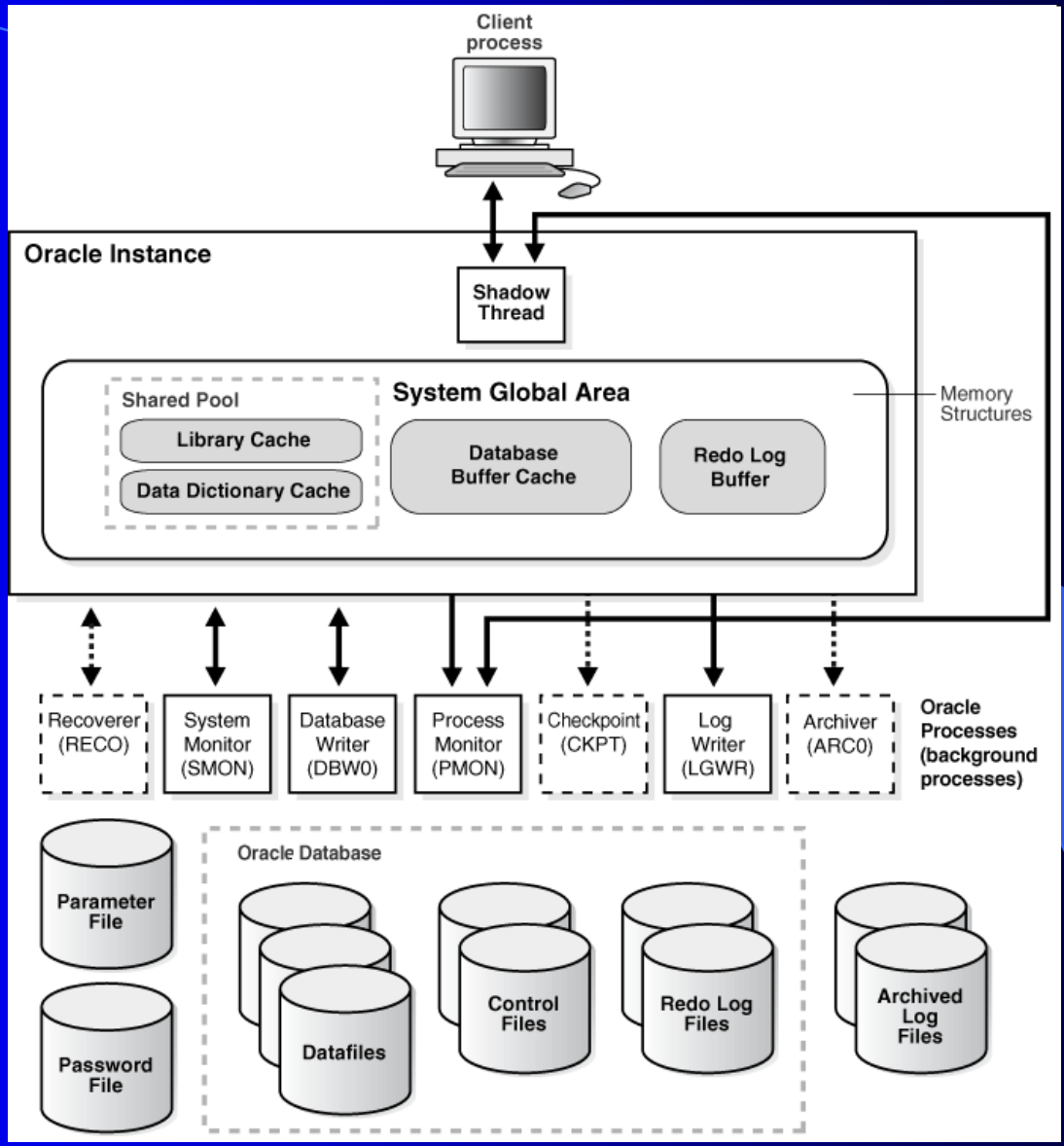
Oracle

- Oracle korporacija je nesumnjivo lider u softveru za baze podataka u oblasti velikih firmi i preduzeća. Oracle-ove baze su u širokoj upotrebi u različitim tipovima aplikacija i veoma su popularne zbog malog broja mana :
- Svestranost – Oracle u ponudi ima puno proizvoda za e-commerce koji se integriše sa njihovim bazama podataka što pomaže u procesu dizajniranja, pravljenja i korišćenja aplikacija za baze podataka.
- Stabilnost – Oracle-ovi serveri veoma retko zataje što je veoma bitno onima kojima su baze podataka potrebne 24h dnevno.
- GUI – Oracle nudi veliki broj GUI alata za upravljanje bazama podataka na taj način olakšavajući rad.
- Bezbednost – novije verzije sada imaju *toolkit* za obezbeđivanje sigurnosti tako što obezbeđuje enkripciju osetljivih podataka unutar baze podataka.
- Više platformska podrška – populane verzije Oracle-a uključuju i verzije za MS Windows kao i za sve popularniji Linux.

Oracle


RECO
PMON
SMON
System Global Area
Database Buffer Cache
Redo Log Buffer
User Process
User Processes
Shared Server Process
Dedicated Server Process
D000
CKPT
ARC0
Offline Storage Device
DBW0
LGWR
User Process
Control Files
Redo Log Files
Datafiles
Legend:
RECO Recoverer process
PMON Process monitor
SMON System monitor
CKPT Checkpoint
ARC0 Archiver
DBW0 Database writer
LGWR Log writer
D000 Dispatcher Process

Oracle


Client process
Oracle Instance
Shadow Thread
Shared Pool
Library Cache
Data Dictionary Cache
System Global Area
Database Buffer Cache
Redo Log Buffer
Memory Structures
Recoverer (RECO)
System Monitor (SMON)
Database Writer (DBW0)
Process Monitor (PMON)
Checkpoint (CKPT)
Log Writer (LGWR)
Archiver (ARC0)
Oracle Processes (background processes)
Parameter File
Password File
Oracle Database
Datafiles
Control Files
Redo Log Files
Archived Log Files

SAVREMENE BAZE PODATAKA

Oracle

- Mane:

- *Potencijalno visoki troškovi posedovanja*. (Total Cost of Ownership – TCO) – Oracle-ovi serveri baza podataka zahtevaju najmodernije hardverske resurse (najbitnije procesor i RAM memorija) da bi radili na prihvatljivom nivou.

SAVREMENE BAZE PODATAKA

Microsoft SQL Server

- Poput Oracle-a, Microsoft je ključni igrač u oblasti baza podataka, mada konstantno mora da pristiže druge pogotovu u oblasti Interneta i e-komerca. Iako je Microsoft priznati lider u oblasti desktop operativnih sistema tu prednosti nije preneo na oblast elektronske trgovine.

  Prednosti MS SQL su :

- *Visoka stabilnost.* – MS SQL Server nudi stabilnost koja je dizajnirana da bude u skladu sa Windows OS.
- *Lakoća korišćenja*. – MS SQL Server nudi prepoznatljiv GUI koji se koristi i u Windows operativnim sistemima tako da je vreme učenja korišćenja svedena na minimun.
- *Saglasnost sa ANSI SQL-92.* – Potpuno je kompatibilan sa ANSI SQL-92 standardom i čak ga u nekim stvarima poboljšava.

SAVREMENE BAZE PODATAKA

- ***Microsoft SQL Server***

  Mane :

- Zahvaljujući brojnim sigurnosnim *propustima na nivou operativnog sistema* mnogi ozbiljniji korisnici izbegavavaju da ulažu novac u MS SQL Server za potrebe firmi. Dodatno, primoravanje za restartom host računara da bi se update-ovala baza podataka ili softver odbija mnoge potencijalne korisnike kojima je potrebna 24-časovna dostupnost bazi podataka.
- *Viskoka cena posedovanja (TCO).* – Poput operativnih sistema na kojima radi MS SQL Server je veoma hardverski zahtevan. Potrebno mu je mnogo procesorske snage kao i velika količina slobodne RAM memorije. Ovaj aspekt najviše pogađa mala preduzeća, koja pored licenciranja MS SQL Servera od par hiljada dolara, računajući troškove za nabavku samog operativnog sistema kao i dodatnog softvera i hardvera da bi pokrenuli bazu podataka, imaju velike izdatke.
- *Vlasnička svojina*. – Pošto MS SQL Server *nije multi platformski*, neki potencijalni kupci se ustežu od kupovine pošto onda zavise samo od jednog dobavljača. Ako se dobavljač odluči da značajno povisi cene dodataka ili ispravki softvera morali bi da uđu u nepredviđene troškove.

SAVREMENE BAZE PODATAKA

PostgreSQL

- Novopridošlica u polju RDBMS-a, PostgreSQL se brzo uzdigao. Dosta je stabilan za tako nov proizvod ali je još dosta rada pred njim da bi mogao u značajnijoj meri da konkuriše velikim igračima.

  Prednosti :

- *Saglasnost sa SQL-92.* – PostgreSQL se najviše pridržava SQL-92 standarda.
- *Multiplatformska dostupnost.* – Distribucije PostgreSQL-a su dostupne za najkorišćenije kompjuterske platforme uključujući Windows 2000/NT i MacOS X. Kao Open Source paket stiže sa mnogim verzijama Linux operativnog sistema.
- *Niski troškovi posedovanja (TCO).* – Softver PostgreSQL baza-server je dostupan za minimalne izdatke. Softver je BESPLATAN što je potencijalno prednost u odnosu na Oracle ili Microsoft SQL Server.

SAVREMENE BAZE PODATAKA

PostgreSQL

- Mane :

- *Relativno ograničeno prihvatanje.* – Iako PostgreSQL podržava veoma važne funkcije velikih RDBMS proizvoda naročito transakcije, on je sporiji u odnosu na konkurenciju (posebno u odnosu na Oracle). To je možda i jedini razlog što PostgreSQL nije prihvaćen od mnogih velikih firmi.

SAVREMENE BAZE PODATAKA

Informix

- IBM-ova Informix serija servera je usmerena ka velikim aplikacijama za baze podataka. Informix je popularan RDBMS jer iza njega stoji jedan gigant kao što je IBM što se i odražava na njegove karakteristike.

  Prednosti :

- *Raznovrsna proizvodna linija.* – Informix u ponudi ima *širok spektar servera* u zavisnosti od potreba korisnika. Počev od on-line transakcija do paralenlnog procesiranja i dr. Informix proizvodi optimizovane servere za gotovo sve poznate potrebe.
- *Multiplatformska dostupnost.* – Informix radi na mnoštvu platforma i takođe nudi alat koji pomaže pri razvijanju aplikacija.
- *Dokumentacija i podrška.* – Za razliku od najvećih konkurenata Oracle-a i Microsoft-a IBM na svom Web sajtu nudi dokumentaciju za Informix koja je na zavidnom nivou.

SAVREMENE BAZE PODATAKA

Informix

- Mane :
- *Viskoka cena posedovanja.* – Kao i kod najvećih konkurenata potrebno je imati veoma jak hardver koji bi pokretao server kako su to zamislili programeri IBM-a što je često preveliko opterećenje za male firme pa ga one iz tog razloga najčešće izbegavaju.

SAVREMENE BAZE PODATAKA

MySQL

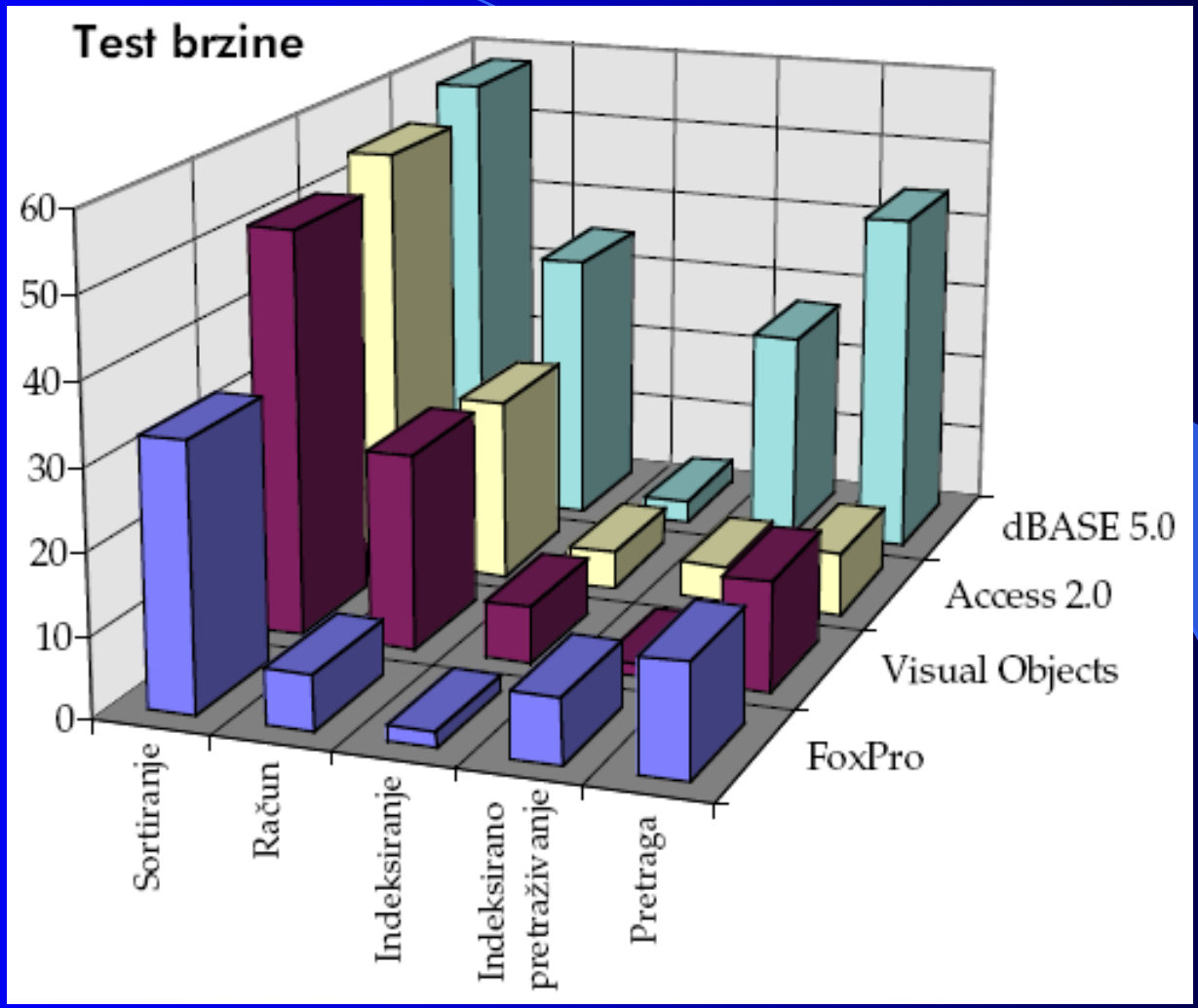
- Iako je najmlađi od gore pomenutih SQL servera MySQL daje najbolje iz sveta baza podataka. Dostupan je za ubedljivo najviše računarskih platformi. Trenutno je dostupan za Linux, Windows 95/98/2000/NT, Solaris, FreeBSD, MacOS X, HP-UX, AIX, SCO, SCI Irix, Dec OSF i BSDi. Linux verzije rade na mnoštvu arhitektura kao što su Intel libc6, Aplha, IA64, SPARC i S/390. Dostupnost na svim ovim platformama samo je još više doprinela njegovoj popularnosti i primeni. Pored običnog MySQL servera postoji i poboljšana verzija MySQL-Max. MySQL-Max pored servera sadrži i tabele za zaštitu transakcija kao što je Inno DB ili Berkeley DB.
- MySQL je dostupan kao binarni fajl ili kao source kod. Ako vam je potrebno da svojoj aplikaciji dodate MySQL osobine dovoljno je uzeti source kod MySQL-a i modifikovati ga za vaše potrebe. MySQL je pod zaštitom GNU General Public Licence (GPL) i GNU Lesser General Public Licence (LGPL) i u većini slučajeva je besplatan. MySQL takođe ima puno interfejsa za programiranje aplikacije (Application Programming Interface – API) koji programeru dozvoljavaju da pristupa i oblikuje bazu podataka. API-i su dostupni za C, C++, Tcl, Python, PHP i Perl. Neki od najpopularnijih za programiranje Web interfejsa su PHP i Perl.

SAVREMENE BAZE PODATAKA

MySQL

- Najveće prednosti :
- *Web aplikacije.* – Web aplikacije obično imaju mnogo isčitavanja i malo upisa. MySQL je dovoljno brz i može zadovoljiti zahteve Internet brzina.
- *Open Source.* – MySQL je open-source što znači da svako može da uzme source kod i da prilagodi sopstvenim potrebama.
- *Mala hardverska zahtevnost*. – MySQL radi čak i na računarima tipa Intel Pentium sa 32 Mb RAM-a pa čak i slabijim ali ipak, na takvim računarim ne treba imati velika očekivanja u pogledu performansi.
- *Stabilnost.* – Sa svakom novijom verzijom MySQL je sve stabilniji. Tome najviše doprinosi što je MySQL open-source pa u njegovoj izradi učestvuje mnogo više programera, na dobrovoljnoj osnovi, no što bi i neka velika kompanija sebi mogla da priušti da zaposli.

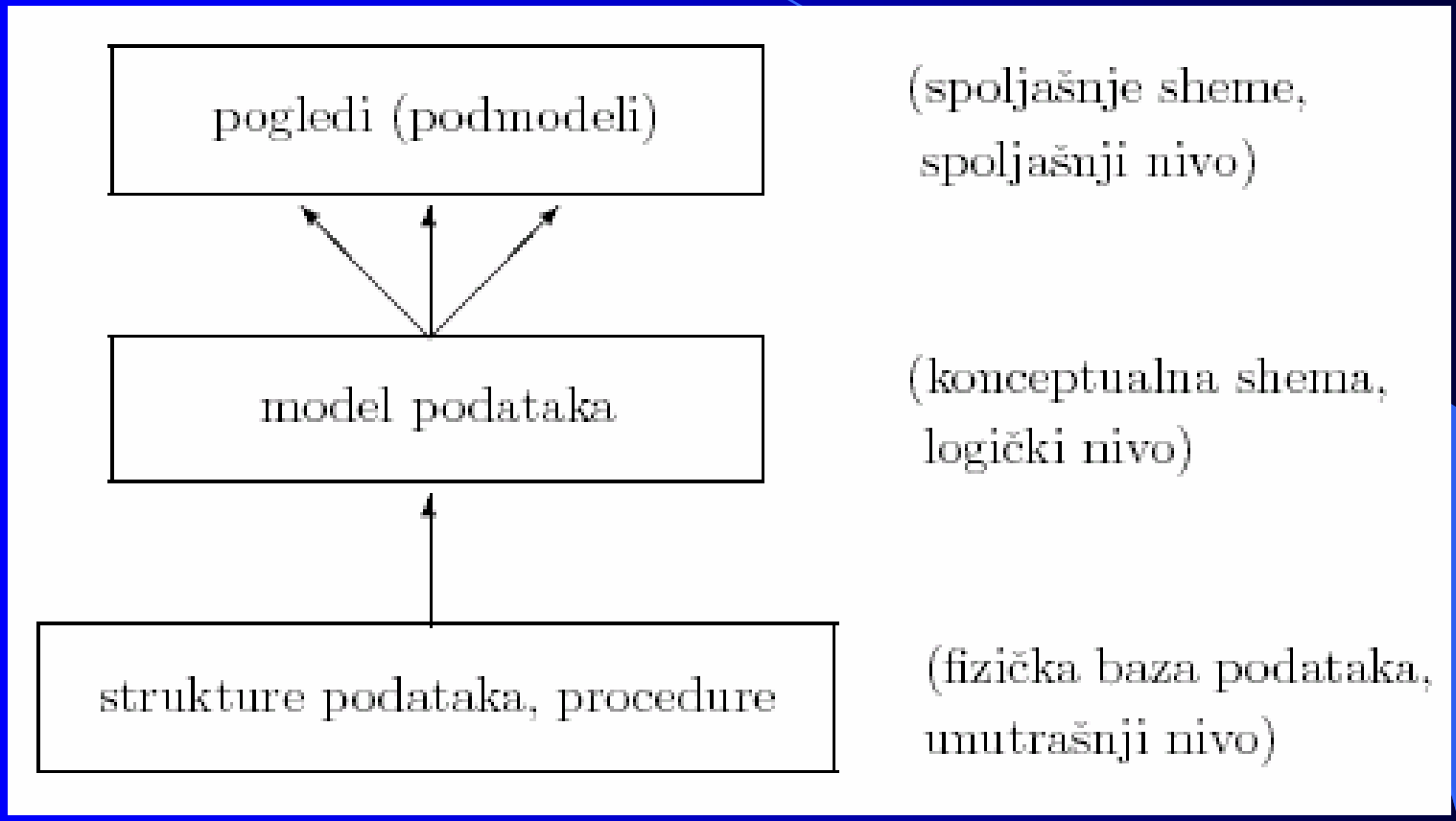
Test brzine


Test brzine
60
50
40
30
20
10
0
Sortiranje
Račun
Indeksiranje
Indeksirano pretraživanje
Pretraga
dBASE 5.0
Access 2.0
Visual Objects
FoxPro

ARHITEKTURA

- Arhitektura najvećeg broja sistema baza podataka odgovara predlogu **ANSI/SPARC** studijske grupe Američkog nacionalnog instituta za standarde, i poznata je kao ANSI arhitektura. Ova arhitektura predstavljena je hijerarhijom apstrakcija, pri čemu svaki nivo hijerarhije uključuje specifični način predstavljanja, reprezentaciju, objekata, odnosa medu objektima i operacija nad objektima. Hijerarhijska arhitektura omogućuje prirodnu dekompoziciju i efikasni razvoj sistema za upravljanje bazama podataka.

ARHITEKTURA



ANSI arhitektura sistema baza podataka

Projektovanje IS svodi na neku vrstu modeliranja realnog sistema

- Neophodna su neka intelektualna sredstva (alati) i to:
- (1) Model podataka kao intelektualno sredstvo za prikazivanje objekata sistema, njihovih atributa i njihovih međusobnih veza (statičkih arakteristika sistema) preko logičke strukture baze podataka.

- (2) Model procesa kao intelektualno sredstvo za opisivanje dinamike sistema, dejstva ulaza na stanje sistema i izlazne transformacije, preko programa nad definisanim modelom podataka.

Podatak je neka kodirana činjenica iz realnog sistema, on je nosilac informacije. *Informacija* je protumačeni (interpretirani) podatak.

- Interpretacija podataka se vrši na osnovu:
  - strukture podataka,
  - semantičkih ograničenja na njihove vrednosti, i
  - preko operacija koje se nad njima mogu izvršiti.

  Zbog toga svaki model podataka poseduje tri osnovne komponente:

- **(1) Struktura modela**, odnosno skup koncepata za opis objekata sistema, njihovih atributa i njihovih međusobnih veza.

- **(2) Ograničenja** - semantička ograničenja na vrednosti podataka koja se ne mogu predstaviti samom strukturom modela, a koja u svakom trenutku moraju biti zadovoljena. Ova ograničenja se obično nazivaju pravilima integriteta modela podataka.

- **(3) Operacije nad konceptima strukture**, pod definisanim ograničenjima, preko kojih je moguće opisati dinamiku sistema u modelima procesa.

Modeli podataka

- Na osnovu definicije, očigledno je da svaki model podataka treba da zadovolji dva bitna kriterijuma:

  -Da poseduje koncepte pogodne za modeliranje realnih sistema.

  -Da se njegovi koncepti, struktura, ograničenja i operacije mogu jednostavno implementirati na računaru.

Modeli podataka

Modeli podataka su se pojavili pedesetih godina prošlog veka a mogu se klasifikovati na:

- **modele druge generacije -hijerarhijski, mrežni, relacioni** i
- **modele treće generacije -objektni model podataka.**
- **modele prve generacije -programski jezici,**

Modeli podataka

Modeli podataka klasifikovani u “generacije” redom čine:

- Prvu generaciju čine konvencionalni programski jezici, koji se takođe mogu tretirati kao modeli podataka. Koncepti za opisivanje strukture u njima su tipovi podataka kojima raspolažu (prosti i agregirani tipovi kao što su rekord, vektor, skup, koji se međusobno ne mogu eksplicitno povezivati na željene načine, pa se zato model podataka u njima predstavlja kao skup nepovezanih datoteka). Ograničenja nad ovim tipovima obično nije moguće direktno postaviti, a same operacije nisu dovoljno moćne. Oni nisu dovoljno pogodni za modeliranje realnog sistema (zato je programiranje u njima teško), ali se model realnog sistema opisan pomoću njih može direktno implementirati na računaru.

Modeli podataka

- Drugu generaciju čine tri klasična modela baze podataka, hijerarhijski, mrežni i relacioni model. Ovi modeli poseduju semantički bogatije koncepte za opis strukture (moguće je definisati način povezivanja zapisa - rekorda, odnosno bazu podataka kao skup međusobno povezanih podataka) i znatno moćnije (makro) operacije. Zbog toga su pogodniji za opis realnog sistema, a postoje i komercijalno raspoloživi softveri, SUBP, za njihovu direktnu implementaciju na računaru.

- Treću generaciju čine tzv. semantički bogati modeli podataka i objektni modeli podataka (Model objekt i-veze, Semantic Data Model (SDM), Prošireni relacioni model, Semantičke mreže i drugi), koji poseduju semantički bogate koncepte za opis realnog sistema, ali za njih još uvek ne postoje softveri preko kojih bi se direktno mogli implementirati na računaru.

Modeli podataka

Imajući u vidu karakteristike generacije modela podataka, zadovoljavajući metodološki pristup projektovanju baza podataka bi mogao da bude sledeći:

- (1) Koristeći neki model Treće generacije, formirati semantički bogat model realnog sistema koji se analizira.
- (2) Prevesti dobijeni model u neki od modela Prve ili Druge generacije, u zavisnosti od softvera kojim se raspolaže i drugih činioca i na taj način implementirati bazu podataka na računaru. Postupak prevođenja sa semantički bogatijeg, na semantički siromašniji model može se uvek formalizovati, pa samim tim i automatizovati.

Modeli podataka

Model podataka

- Model podataka predstavlja sredstvo za izgradnju modela realnog sistema. Model podataka obuhvata reprezentaciju nekih objekata iz realnog sveta i osnosa između ovih objekata. Cilj je da se obezbedi informacija o statičkim i dinamičkim osobinama realnog sistema. Sastoji se od tri komponente:

- - strukturalne
- - integritetne
- - operacijske

- ***Strukturalna komponenta modela podataka*** definiše strukture nad skupom atributa i nad skupom podataka. Sadrži skup primitivnih koncepata i skup pravila za izgradnju koncepta.
- ***Integritetna komponenta modela podataka*** definiše ograničenja nad vrednostima atributa, veze između podataka i medusobnu uslovljenost podataka.
- ***Operacijska komponenta modela podataka*** definiše operacije nad strukturama podataka i modelira dinamičke osobine realnog sistema.

Modeli podataka

Kategorije modela podataka

- ***Konceptualni model podataka*** nudi koncepte, koji odgovaraju krajnjem korisnlku. To su entiteti, atributi i veze.
- ***Fizički model podataka*** nudi koncepte za opis detalja o fizičkom načinu memorisanja podataka u računaru. Sadrži: formate slogova, uređenost slogova i puteve pristupa.
- ***Implementacioni model podataka*** nudi koncepte koji odgovaraju krajnjem korisniku, ali nisu daleko od načina na koji su podaci organizovani u računaru.

Modeli podataka

Hijerarhijski model baze podataka:

- Hijerarhijski model baze podataka pojavio se pedesetih godina prošlog veka. Premda nam na prvi pogled hijerarhijski pristup problemu izgleda i deluje kao prihvatljiv, razumljiv i jednostavan (u svakodnevnom životu naviknuti smo na hijerarhijske strukture, od autoriteta roditelja u porodici, preko firme u kojoj radimo, vojske u kojoj smo služili vojni rok, preduzeća u kome smo zaposleni, crkve ili neke druge institucije kojoj pripadamo), potrebno je naglasiti da je sa stanovišta optimalnog projektovanja i manipulacije podacima, reč o nepovoljnom modelu. Loša strana hijerarhijskog modela je u prvom redu nedostatak egzaktne matematičke teorije koja bi omogućila njegovu punu implementaciju na računaru.

Modeli podataka

Primer1

- Pretpostavimo da je u nekoj firmi potrebno projektovati informacioni sistem (bazu podataka) internog školovanja za službenike. Preduzeće drži čitav niz različitih kurseva na različitim lokacijama u gradu. Neki od kurseva se nastavljaju jedan na drugi, pa je uspešno završen prethodni kurs, uslov za upis narednog.

- *Odgovarajuća baza podataka, nazovimo je DOKVALIFIKACIJA, treba za svaki kurs da sadrži ove informacije: broj kursa, naziv kursa, mesto i vreme održavanja, broj kursa koji je uslov za uspešno pohađanje sledećeg, detalje o predavačima (ime, prezime, adresa,..), detalje o polaznicima (ime, prezime, ocena, itd).*

- Struktura hijerarhijskog modela često se predstavlja grafički. Sa *skice je vidljivo da hijerarhijska struktura ima svoj naziv, svoju osnovu ili koren (DOKVALIFIKACIJA), koja se u zavisnosti od strukture dalje grana "prema dole".*

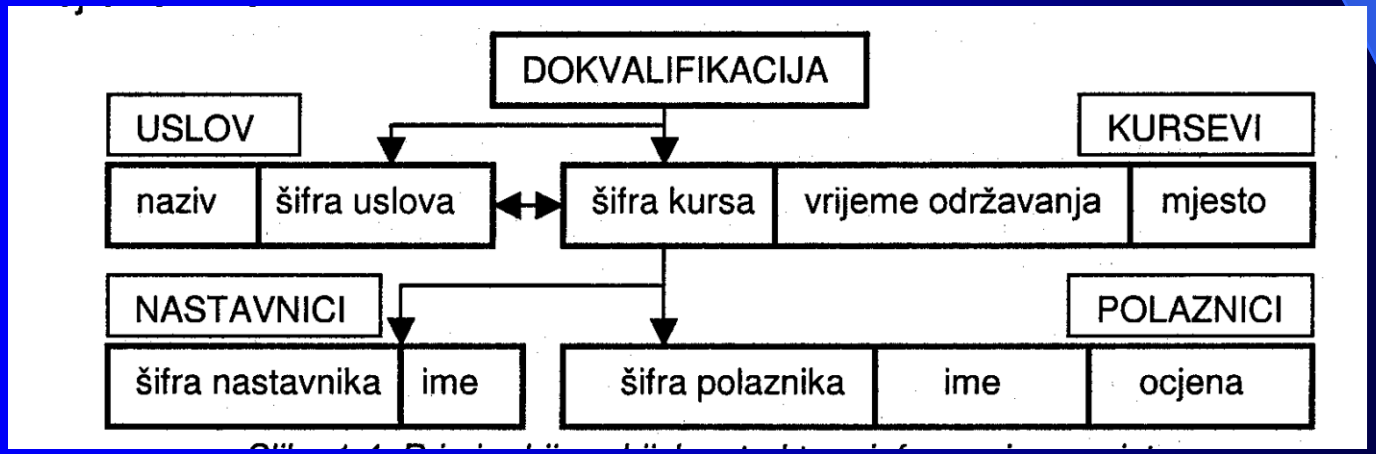
Modeli podataka

- *U konkretnom slučaju osnova, DOKVALIFIKACIJA, ima svoje dve podtabele koje je slede. To su:*

USLOV i KURSEVI

pri čemu tabela USLOV mora biti logički povezana sa *tabelom KURSEVI (mora* se *znati šta je preduslov* za *pohađanje nekog narednog kursa). Tabela USLOV nema svojih podtabela, dok tabela KURSEVI ima dve, dva "naslednika": dve podtabele i to:*

NASTAVNICI** i **POLAZNICI



Modeli podataka

- Precizno definisana međusobna povezanost atributa pojedinih tabela definiše uspostavljenu hijerarhiju kao što je to slučaj u poslednjem primeru:

KURSEVI -POLAZNICI,
KURSEVI- NASTAVNICI, ili
DOKVALIFIKACIJA -USLOV

a što je na predhodnoj slici vidljivo.

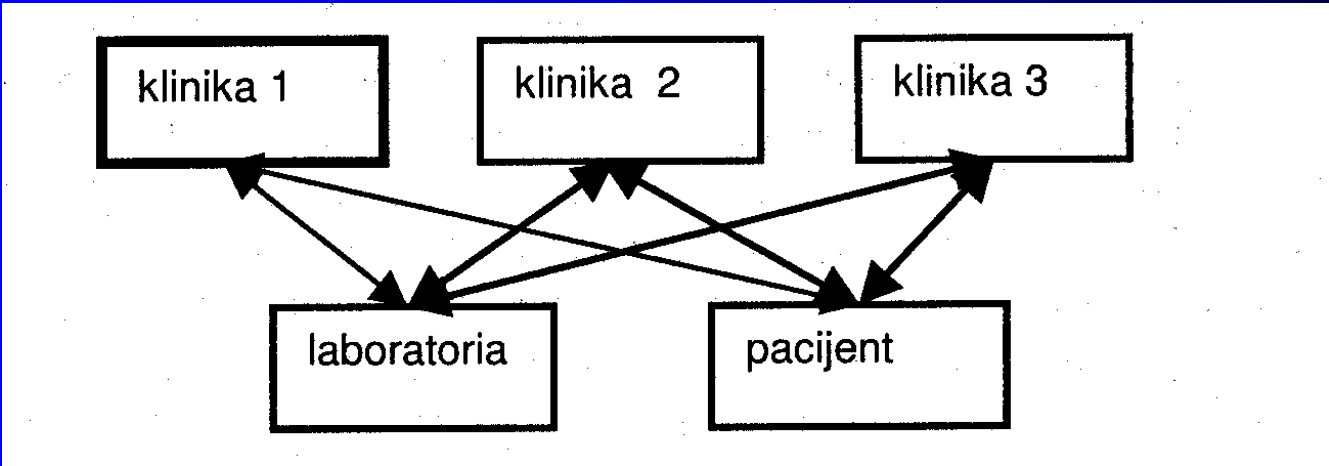
- *Hijererhijskim modelom sve veze među tabelama, odnosno među atributima pojedinih tabela, moraju biti precizno definisane.*

- Ima slučajeva za koje je u hijerarhijskom modelu teško, a nekada i nemoguće, naći rešenje. To se posebno odnosi na slučajeve kada *jedan roditelj može imati više naslednika, ali i svaki naslednik može imati više roditelja".*

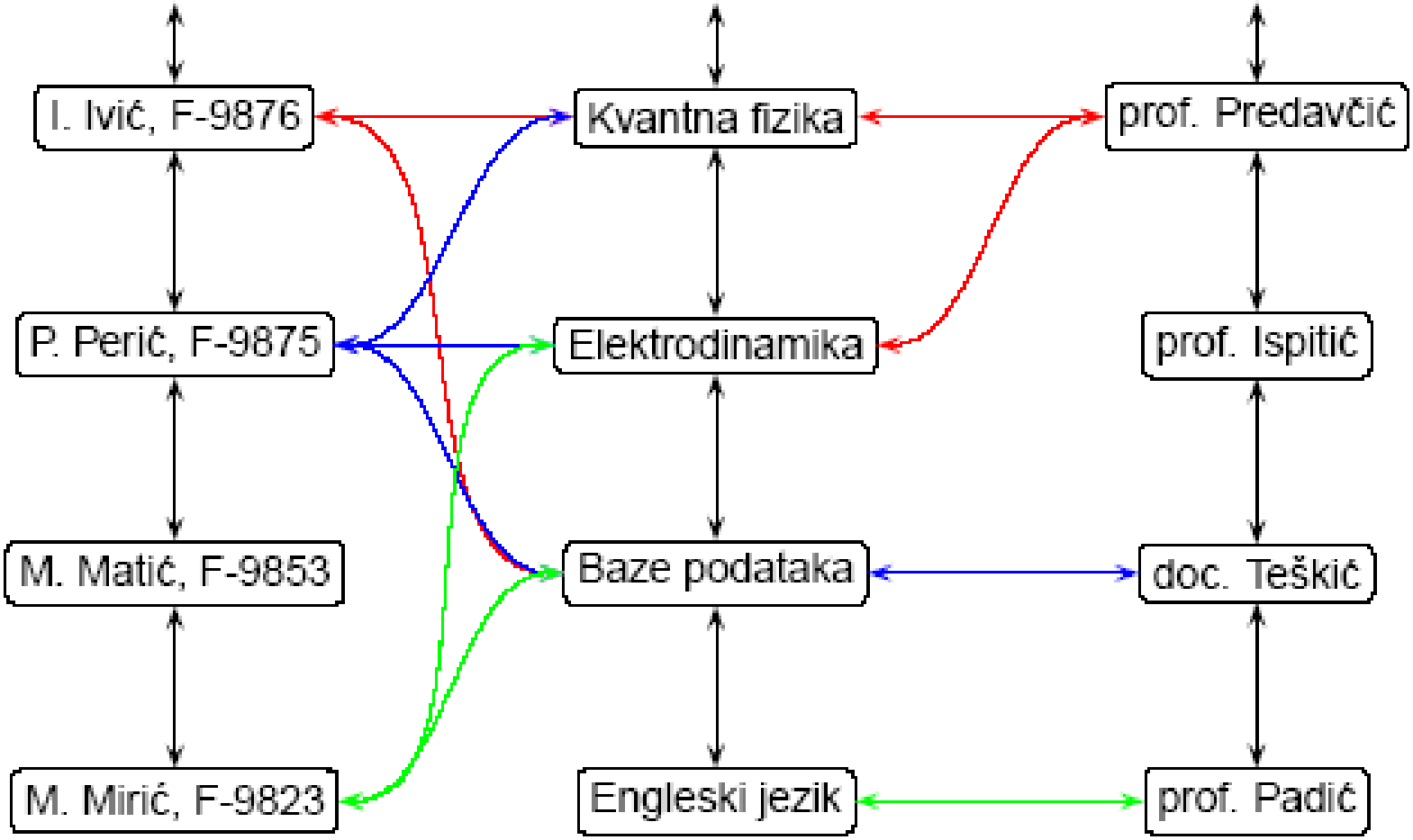
Modeli podataka

Mrežni model

- Mrežni model informacionog sistema nastao je kao proširenje hijerarhijskog. Osnovna razlika, nazovimo to poboljšanje, u odnosu na hijerarhijski model, leži u činjenici da se dozvolilo da "naslednik" ima više "roditelja", što znači da ovakav model prihvata i veze tipa **M:N.**
- Na taj način veze među slogovima pojedinih tabela, grafički interpretirane, liče na **paukovu mrežu** odakle je model i dobio ime.



Primjer mrežnog modela



Pojedini zapisi sadrže **pokazivače** na druge zapise s kojima su povezani.

Modeli podataka

Relacioni model

- Relacioni model je danas najpopularniji model baza podataka i to zahvaljujući pre svega sledećim osobinama:
  1. struktura modela je veoma jednostavna,
  2. baza podataka predstavlja skup tabela, i
  3. moguća je formalno-matematička interpretacija tabela.
- Kao što mu i samo ime govori, ovaj model se zasniva na tabelama i relacijama među njima. Kako se i relacija (veza) između tabela može u relacionom modelu opet prikazati kao tabela, u relacionom modelu sve je definisano samo tabelama.
- Obzirom da se podaci u informacionom sistemu menjaju u vremenu (prate proces) relaciona baza podataka se sastoji od vremenski promenljivih tabela (relacija) koje mogu biti *bazne* i *izvedene*.

Primjer relacijskog modela

Studenti

F-9876	I. Ivić
F-9875	P. Perić
F-9853	M. Matić
F-9823	M. Mirić

Predmeti

K-1234	Kvantna fizika
K-2345	Elektrodinamika
K-3456	Baze podataka
K-4567	Engleski jezik

Nastavnici

prof. Predavčić
doc. Ispitić
doc. Teškić
prof. Padić

Studeni i upisani predmenti

F-9876	K-1234
F-9876	K-3456
F-9875	K-1234
F-9875	K-2345
F-9875	K-3456
F-9823	K-2345
F-9823	K-3456

Nastavnici predmenta

K-1234	prof. Predavčić
K-2345	prof. Predavčić
K-3456	doc. Teškić
K-4567	prof. Padić

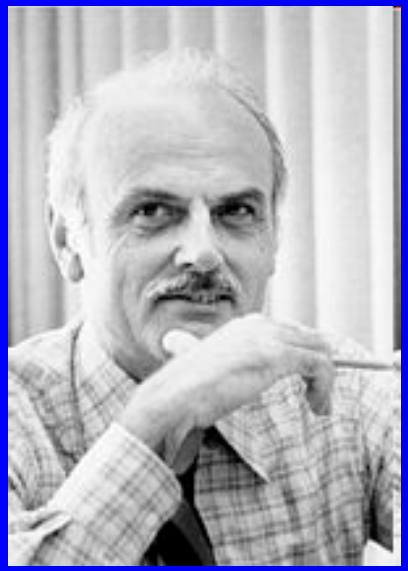
Tablica 1: U relacijskom modelu struktura podataka je jednostavna, bez **pokazivača** koji opisuju veze s nekim drugim podacima. Veze među podacima su opisane u posebnim tablicama i koje također imaju jednostavnu strukturu.

Modeli podataka

- *Bazne tabele* se još nazivaju i fizičke tabele, jer one postoje na memorijskom medijumu računara -disku, a **izvedene relacije** se mogu dobiti iz baznih relacija operacijama koje se definišu nad baznim tabelama.

- *Izvedene tabele* nastaju kao rezultat neke operacije nad fizičkim tabelama i postoje samo u operativnoj memoriji računara (ali ne i na disku) jedno izvesno vreme -dok su nam potrebne, pa se stoga nazivaju i virtualne. Ako virtualna tabela ima i svoje ime naziva se *pogled* (view) i formalno se može koristiti kao fizička tabela uz napomenu da se u njoj ne nalaze memorisani podaci nego se koriste originalni podaci iz baznih tabela. Shodno tome, podaci se preko pogleda ne mogu ni menjati, sem ako je to pogled nad podacima iz samo jedne tabele i uz još neka ograničenja.
- Na ovaj način se povećava pouzdanost podataka (izmena podataka je strogo kontrolisana) i bitno se smanjuje redundanca podataka. *Redundanca podataka je višestruko memorisanje istih podataka na više mesta u informacionom sistemu.*

KODOVA PRAVILA



IBM kompjuterski stručnjak, postavio osnove relacionog modela BP.

- ***E. F .Codd*** je u svojim radovima definisao 12 pravila od kojih najmanje 6 mora biti ispunjeno da bi se neki informacioni sistem mogao smatrati relacionim. Ta pravila su rezultat ozbiljnog teoretskog pristupa rešavanju ovoga problema. S obzirom na obim ove knjige ograničimo se samo na definiciju 6 osnovnih pravila, ne upuštajući se i u njihovo dokazivanje.
- Osnovno ili nulto pravilo koje predstavlja "conditio sine qua non" u relacionim bazama podataka glasi:
- *Svaki sistem za upravljanje bazama podataka, koji se smatra, ili koji jeste, relacioni, mora imati mogućnost upravljanja bazom podataka na relacioni način i relacionim metodama.*

KODOVA PRAVILA

- 1. **Predstavljanje informacija**

  Sve informacije u relacionoj bazi podataka moraju logički biti predstavljene na isti način: **vrednostima podataka** u **tabelama - relacijama.**

- 2. **Logička dostupnost podacima**

  Svaki podatak mora imati takozvanu "atomarnu" vrednost, koja logički mora biti dostupna korisniku uz pomoć:

  -imena relacije u kojoj se nalazi,

  -vrednosti primarnog ključa i imena atributa u toj relaciji.

- 3. **Mogućnost prikazivanja nepostojeće informacije**

  Relaciona baza podataka mora podržavati koncept nultog podatka, odnosno nulte vrednosti *(Null* vrednost). Pod pojmom nultog podatka podrazumeva se vrednost nekog atributa koja u da-tom trenutku nije poznata. To nije vrednost koja je predstavljena nulom ili nizom nula, nizom praznih (blank) mesta, ili nizom bilo kojih drugih znakova. To je jednostavno, trenutno nepoznata vrednost podataka, i predstavlja specifikum i najkontroverzniji aspekt relacionog modela baze podataka.

  -Tako atribut telefon u relaciji **službenik** *ima Null vrednost u trenutku primanja nekog službenika u radni odnos, ukoliko taj službenik u trenutku zasnivanja radnog odnosa nema svoj telefonski broj.*

KODOVA PRAVILA

- 4. **Dinamički katalog**

  Relaciona baza podataka mora biti tako prezentirana da autorizovani korisnici mogu primeniti neki od "relacionih jezika" na sve podatke u njoj. Pod ovim se podrazumevaju i sistemski katalozi, zatim relacije koje su u toku rada za stalno, ili na ograničeni vremenski rok, programski kreirane (tzv. dinamički katalozi), a koje sadrže neke nove informacije.

- 5. **Softverski paket za manipulaciju bazom podataka**

  Mnogi softverski paketi omogućavaju danas manipulisanje podacima unutar baze podataka, i oni su manje ili više prilagođeni korisniku -manipulatoru. Svaki od njih sadrži programski jezik koji pomoću precizno definisane sintakse, i na korisniku blizak način, omogućava:

  -definisanje tabela,
  -definisanje atributa, unošenje podataka,
  -definisanje proizvoljnog "pogleda" na bazu podataka,
  -manipulaciju podacima,
  -postavljanje ograničenja korisnicima, autorizaciju korisnika,te
  -upravljanje proizvoljnim transakcijama.

KODOVA PRAVILA

- **6. Nezavisnost integriteta baze**
- Nijedno ograničenje integriteta podataka (sigurnosti očuvanja podataka) ne sme se pri organizaciji logičkog modela relacione baze podataka naći u aplikativnom delu programa dostupnom širem broju korisnika, nego isključivo u delovima koji su pod kontrolom administratora baze. Drugim rečima, pravila integriteta se definišu u okviru baze podataka.
- *Ograničenja u relacionom modelu:* u relacionom modelu je potrebno ograničiti vrednosti nekih atributa u pojedinim relacijama. Sam domen atributa već predstavlja određeno ograničenje (starost, cena, itd...). Postoje tzv. opšta ograničenja koja važe za svaki relacioni model i koja se nazivaju pravilima integriteta relacionog modela.

-Integritet entiteta: Nijedan atribut koji je primarni ključ ili deo primarnog ključa neke bazne relacije ne može da uzme null vrednost.

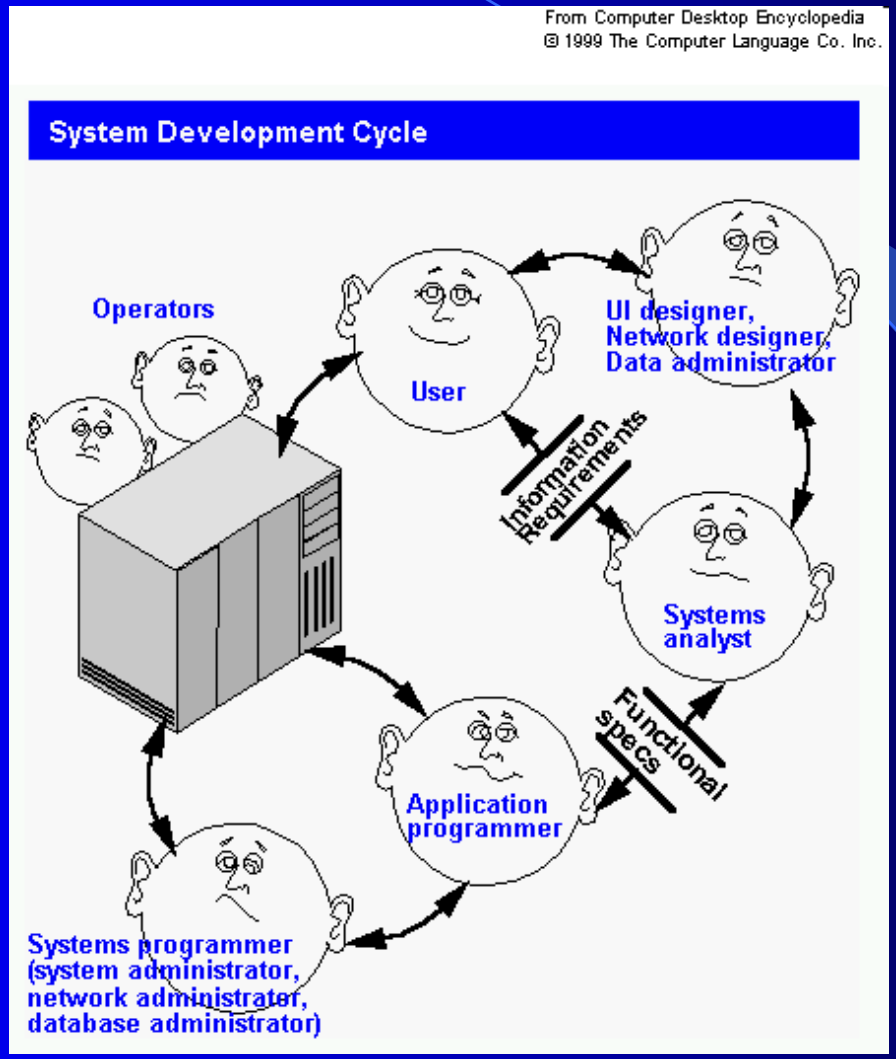
-Referencijalni intergritet: Skup vrednosti spoljnjeg ključa relacije R1 mora biti podskup skupa vrednosti primarnog ključa relacije R2 sa kojom se povezuje relacija R1. Relacije R1 i R2 ne moraju biti različite.

KODOVA PRAVILA

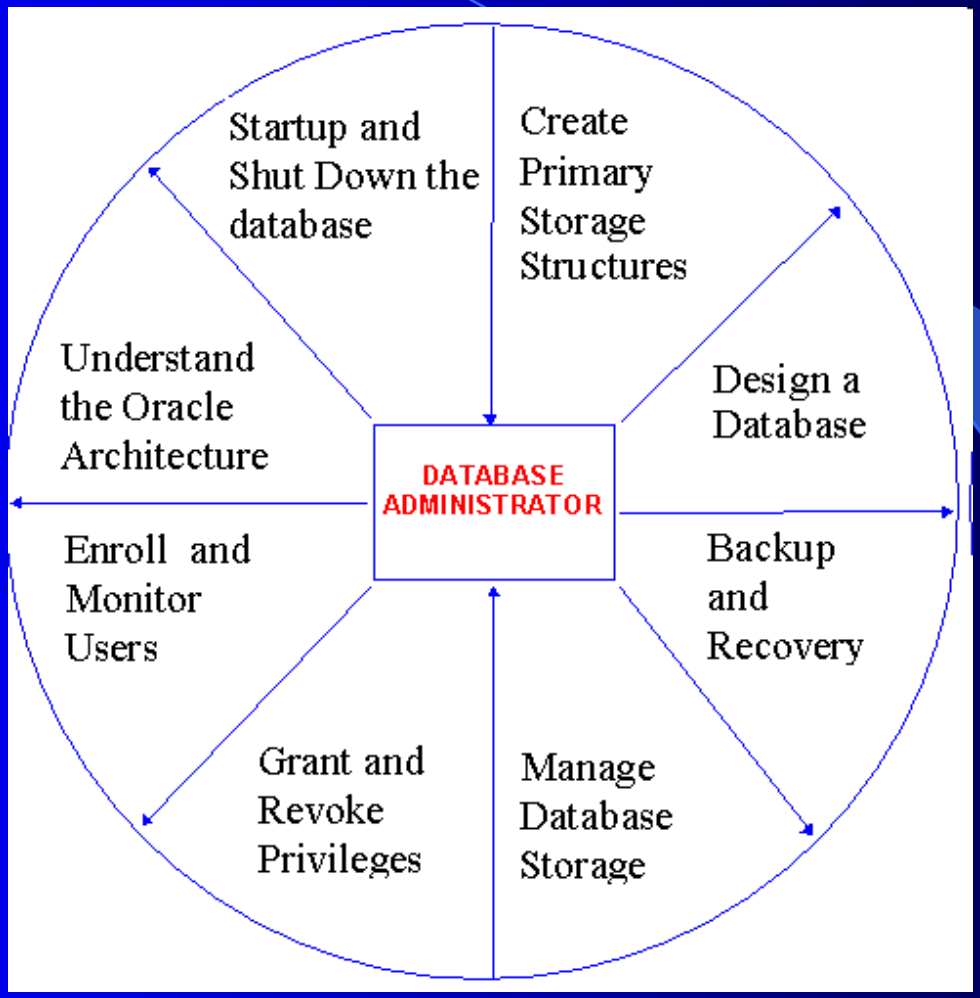
Ostala pravila koja definišu kvalitet i koja mora da ispuni neki sistem za upravljanje bazama podataka da bi se smatrao relacionim:

- **7. Jezikom treće generacije se ne mogu zaobići pravila integriteta definisana relacionim jezikom;**
- **8. Mogućnost kreiranja pogleda i definisanja operacija nad njima;**
- **9. Nad pogledima se mogu izvršavati i operacije održavanja baze podataka;**
- **10. Aplikacioni programi ostaju nepromenjeni ako se promene bazne tabele (sem u slučaju promene strukture tabela);**
- **11. Aplikacioni programi ostaju nepromenjeni ako se promeni fizička organizacija baze ili fizički metod pristupa;**
- **12. Sve navedene karakteristike su nezavisne od distribucije baze podataka.**

CIKLUS RAZVOJA DB SISTEMA



ADMINISTRATOR BAZE PODATAKA



Administrator baze podataka:

- koristi jezik za definiciju podataka za definiciju opšte logičke strukture baze podataka
- definiše i preslikavanja konceptualnog u interni i konceptualnog u eksterni nivo

Promenom fizičke strukture baze:

•nije neophodno menjati opštu logičku strukturu (šemu), već samo preslikavanje šeme u interni nivo

Promenom strukture šeme baze:

•podšeme ostaju nepromenjene, menja se samo preslikavanje šeme u podšeme

Administrator baze podataka:

Upravlja radom celokupnog sistema obavljajući sledeće funkcije:

- Razvoj konceptualnog nivoa, odnosno upravljanje podacima u sistemu.
- Izbor fizičke organizacije baze podataka i metoda pristupa.
- Definicija i realizacija preslikavanja konceptualni - interni i konceptualni - eksterni nivo, odnosno definisanje ili pomoć u definisanju podšema.
- Obezbeđenje sigurnosti i integriteta baze podataka.
- Definisanje strategija oporavka baze podataka posle mogućih oštećenja.
- Praćenje performansi sistema.

Programer baze podataka

- Programer razvija svoju aplikaciju nad njemu pogodnom logičkom strukturom koja predstavlja njegov "pogled" na celokupnu bazu podataka. On za to koristi neki konvencionalni programski jezik (COBOL, PL/I, C, C++, UML i drugi), koji se naziva "jezik domaćin" (Host Language), u koga se ugrađuju naredbe DDL-a (Data Definition Language) i naredbe jezika za rukovanje podacima u bazi podataka (Data Manipulation Language - DML). Za korisnike neprofesionalce obično postoji i neki neproceduralni upitni jezik (Query Language).

- Mnogi uslužni programi za obavljanje ovih funkcija stoje administratoru baze podataka na raspoloženju:

- Rutine za punjenje baze podataka (BP), "Dump/restore" rutine,
- Rutine za fizičku reorganizaciju BP,
- Rutine za statistike korišćenja podataka.

Međutim, njegov osnovni alat je rečnik (katalog) baze podataka (Data Dictionary). Rečnik baze padataka je baza podataka o bazi podataka (meta baza podataka) i u njemu su opisani svi objekti sistema (podaci, fizičke i logičke strukture, prava korišćenja i slično). Rečnik podataka se puni prilikom definisanja odgovarajućih objekata u sistemu.

Tehnologija sistema za upravljanje bazom podataka, rešava:

- Konkurentna obrada: Pošto se podatak pamti samo jednom, više programa istovremeno (konkurentno) zahteva pristup istim podacima. SUBP mora obezbediti da se konfliktni zahtevi i neželjene intreferencije programa do kojih u ovoj situaciji može da dođe, uspešno razreše.

- Zaštita podataka: Zaštita podataka se obično posmatra sa dva aspekta, *očuvanja integriteta baze podataka i sigurnost podataka*. Termin integritet baze podataka se koristi da označi tačnost, korektnost podataka u bazi. Integritet baze podataka može da naruši neka greška u programu ili neki sistemski otkaz. Sigurnost baze podataka se odnosi na zaštitu baze od neovlašćenog korišćenja i ažuriranja podataka. Problem je izražen kod dinamičkih baza podataka na Internetu. I jedan i drugi problem, u uslovima integrisane baze podataka, postaje očigledno mnogo složeniji, nego u uslovima konvencionalne obrade, gde svaka aplikacija ima "privatnu" datoteku u kojoj se podaci jednostavno štite, a narušavanje integriteta ima samo lokalne posledice.

Tehnologija sistema za upravljanje bazom podataka, rešava:

- Oporavak baze podataka: Ako se u bazi podataka svaki podatak čuva samo na jednom mestu, ako se poslovanje celog sistema zasniva na ovakvoj bazi podataka, tada neki otkazi sistema mogu imati katastrofalne posledice. Zbog toga je neophodno da SUBP, preko periodičnog kopiranja baze podataka na "arhivske memorije" i pamćenja transakcija koje su se između dva kopiranja dogodile, omogući efikasan oporavak baze podataka posle nekog otkaza.
- Zbog složenosti navedenih osnovnih i pomoćnih funkcija SUBP su složeni i veoma skupi softverski sistemi. Međutim prednosti koje pružaju su takve da se i pored visoke cene gotovo svuda uvode.

Ciljevi baze podataka

Primarni ciljevi baze podataka

- višestruko korišćenje podataka
- zaštićena intelektualna ulaganja
- niska cena
- manje izmene organizacije podataka
- poboljšanje performanse
- jasnoća
- lakoća i fleksibilnost korišćenja
- nepredviđeni zahtevi se mogu lako obraditi
- jednostavne izmene
- tačnost i konzistentnost
- sigurnost i tajnost
- dostupnost podataka

Sekundani ciljevi baze podataka

- fizička nezavisnost podataka
- logička nezavisnost podataka
- kontrolisana redundanca
- brz pristup i pretraživanje
- standardizacija podataka unutar organizacije
- rečnik podataka
- jezik za krajnje korisnike
- upravljanje integritetom
- brz oporavak od grešaka
- podesivost

-pomoć pri projektovanju i kontroli

- automatska reorganizacija ili migracija

- **Fizička nezavisnost podataka.** Razdvaja se logička definicija baze od njene stvarne fizičke građe. Znači, ako se fizička građa promeni (na primer, podaci se prepišu u druge datoteke na drugim diskovima), to neće zahtevati promene u postojećim aplikacijama.

- **Logička nezavisnost podataka.** Razdvaja se globalna logička definicija cele baze podataka od lokalne logičke definicije za jednu aplikaciju. Znači, ako se logička definicija promeni (na primer uvede se novi zapis ili veza), to neće zahtevati promene u postojećim aplikacijama. Lokalna logička definicija obično se svodi na izdvajanje samo nekih elemenata iz globalne definicije, uz neke jednostavne transformacije tih elemenata.

- **Fleksibilnost pristupa podacima.** U starijim mrežnim i hijerarhijskim bazama, staze pristupanja podacima bile su unapred definisane, dakle korisnik je mogao pretraživati podatke jedino onim redosledom koji je bio predviđen u vreme projektovanja i implementiranja baze. Danas se zahteva da korisnik može slobodno prebirati po podacima, i po svom nahodenju uspostavljati veze medu podacima. Ovaj zahtev zadovoljavaju jedino relacione baze.

- **Istovremeni pristup podacima.** Baza mora omogućiti da veći broj korisnika istovremeno koristi iste podatke. Pritom ti korisnici ne smeju ometati jedan drugog, i svaki od njih treba imati utisak da sam radi sa bazom.

- **Čuvanje integriteta.** Nastoji se automatski sačuvati korektnost i konzistencija podataka, i to u situaciji kad postoje greške u aplikacijama, i konfliktne istrovremene aktivnosti korisnika.

- **Mogućnost oporavka posle kvara.** Mora postojati pouzdana zaštita baze u slučaju kvara hardvera ili grešaka u radu sistemskog softvera.

- **Zaštita od neovlašćenog korišćenja.** Mora postojati mogućnost da se korisnicima ograniče prava korišćenja baze, dakle da se svakom korisniku regulišu ovlašćenja ˇsta on sme a ˇsta ne sme raditi sa podacima.

- **Zadovoljavajuća brzina pristupa.** Operacije sa podacima moraju se odvijati dovoljno brzo, u skladu sa potrebama odredišne aplikacije. Na brzinu pristupa može se uticati izborom pogodnih fizičkih struktura podataka, i izborom pogodnih algoritama za pretraživanje.

- **Mogućnost podešavanja i kontrole**. Velika baza zahteva stalnu brigu: praćenje performansi, menjanje parametara u fizičkoj građi, rutinsko snimanje rezervnih kopija podataka, regulisanje ovlašćenja korisnika. Takođe, svrha baze se vremenom menja, pa povremeno treba podesiti i logičku strukturu. Ovakvi poslovi moraju se obavljati centralizovano. Odgovorna osoba za to zove se administrator baze podataka.

Mogućnosti baze podataka

- A. Osnovne
- - upravljanje redundancom
- - zajedničko korišćenje podataka
- - ograničavanje neautorizovanog pristupa
- - obezbeđenje više interfejsa
- - predstavljanje kompleksnih veza među podacima
- - obezbeđenje integriteta ograničenja
- -BACKUP i RECOVERY

B. Dopunske

- mogućnosti da se primene standardi
- fleksibilnost
- smanjenje vremena razvoja aplikacije
- posedovanje ažurnih informacija
- ušteda (ekonomičnost)

NEZAVISNOST PODATAKA

- *NEZAVISNOST PODATAKA* je ***mogućnost korišćenja podataka bez poznavanja detalja o reprezentaciji podataka.***
- U konvencionalnoj obradi aplikacioni *programer mora znati* odgovor na sledeća tri pitanja pre bilo kakve manipulacije nad podacima:
  - **Kakav** je format podataka?
  - **Gde** su podaci locirani?
  - **Kako** se pristupa podacima?

Odgovori na ova tri pitanja se ugrađuju u aplikacioni program. Promena bilo koje od ovih stavki zahteva promenu aplikacionog programa.

U bazi podataka korisniku je za korišćenje podataka dovoljno da **zna njihov informacioni sadržaj. Nema potrebe da zna gde su smešteni, kako su predstavljeni i kako se pristupa podacima.**

Zašto nezavisnost podataka?

- Dopušta izmene sadržaja, lokacije, reprezentacije i organizacije baze podataka bez reprogramiranja aplikacionih programa koji koriste bazu podataka
- Dopušta izmene hardvera i softvera sistema bez reprogramiranja aplikacija korisnika
- Dopušta da se izste baze podataka podaci organizuju na drugačiji način za svakog korisnika. Svaki korisnik ima sopstveni pogled na podatke.
- Pojednostavljuje razvoj aplikacionih program
- Nudi centralizovano upravljanje podacima, čime se obezbeđuje sigurnost i integritet baze podataka.

TIPOVI I STRUKTURE BAZE PODATAKA

- **Centralizovana baza podataka** – terminalski pristup
  Podrazumeva smeštaj podataka na jednom mestu (središnjem računaru) i terminalski pristup od strane korisnika. Ovakav pristup znači da se svi zahevi i obrade podataka vrše na središnjem računaru, na kome su smešteni i podaci. Korisnik preko terminala, jedino unosi svoje zaheve, i dobija prikaz rezultata željenih operacija. Ovakav način organizacije baze postavlja velike zaheve na host računaru, koji osim smeštaja svih podataka, vrši i sve operacije obrade podataka, njihovog formatiranja i prikaza. Zato host računar mora imati vrlo veliku procesorsku snagu i visoke performanse.

TIPOVI I STRUKTURE BAZE PODATAKA

- **Client- server pristup**

- Client – server struktura podrazumeva, smeštaj podataka na središnjem - host računaru (server), koji izvršava većinu zahteva za manipulisanje i obradu podataka. Korisnici koji pristupaju bazi, taj pristup ostvaruju preko svojih PC računara, koji su mrežom povezani sa serverom. Za razliku od terminalskog pristupa, kod koga terminalske stanice nemaju nikakve mogućnosti sudelovanja u obradi podataka, PC računari na strani klijenta delomično sudeluju u obradi podataka. Korisnik preko programskog interfejsa formira zahtev za određenim podacima, koji se prosleđuje serveru. Serverski računalr prihvata zahev, pa ga obrađuje, a rezultate te obrade vraća klijentu. Klijentski računalr prihvata tako obrađene podatke, i formira ih u obliku kojeg definiše korisnički interfejs.

TIPOVI I STRUKTURE BAZE PODATAKA

- **Paralelna struktura baze podataka**

Paralelna struktura podrazumeva formiranje baze u okruženju računara međusobno povezanih u lokalnu mrežu. Između računara postoji brza veza, ostvarena preko mrežnih kartica 10/100MBs ili brže. Podaci su najčešće smešteni na samo jednom računaru, ali se pri obradi podataka može koristiti procesorska snaga svih računara u mreži. Ovakvim konceptom baze dobija se mogućnost istovremenog korišćenja resursa više računala, pa pojedini računari mogu imati i nešto slabije karakteristike.

TIPOVI I STRUKTURE BAZE PODATAKA

- **Distribuirana baza podataka**

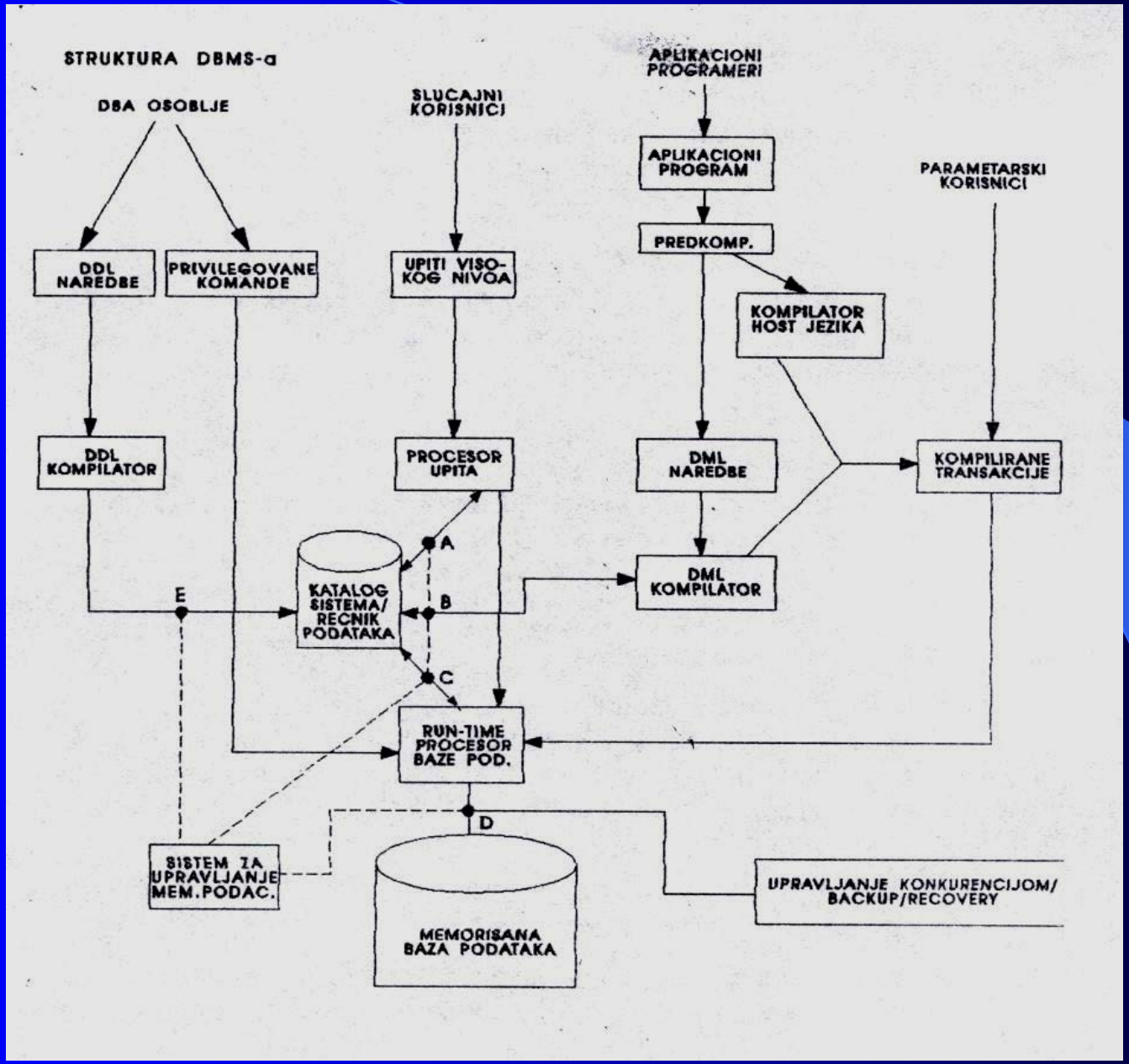
  Podrazumeva strukturu baze podataka u kojoj su podaci raširenи na više računara, koji su mrežno povezani. Jednostavno rečeno, distribuirana baza podrazumeva više lokalnih, međusobno povezanih baza. Pred samim korisnikom je ta "raspršenost" podataka skrivena, i on ima osećaj da pristupa jednoj centralnoj bazi.U današnjim uslovima postoji sve veće potreba za realizacijom distribuiranih baza podataka. Razlozi za to su brojni:

a) Mnogi korisnici za koje se rade baze podataka po svojoj prirodi su distribuirani na više lokacija. Uzmimo primer mnogih multinacionalnih kompanija, koje imaju podružnice diljem sveta. Svaka podružnica u mestu u kojem se nalazi formira svoju lokalnu bazu podataka, a sve te baze zatim se povezuju u distribuiranu bazu podataka, koja objedinjava sve lokalne baze.

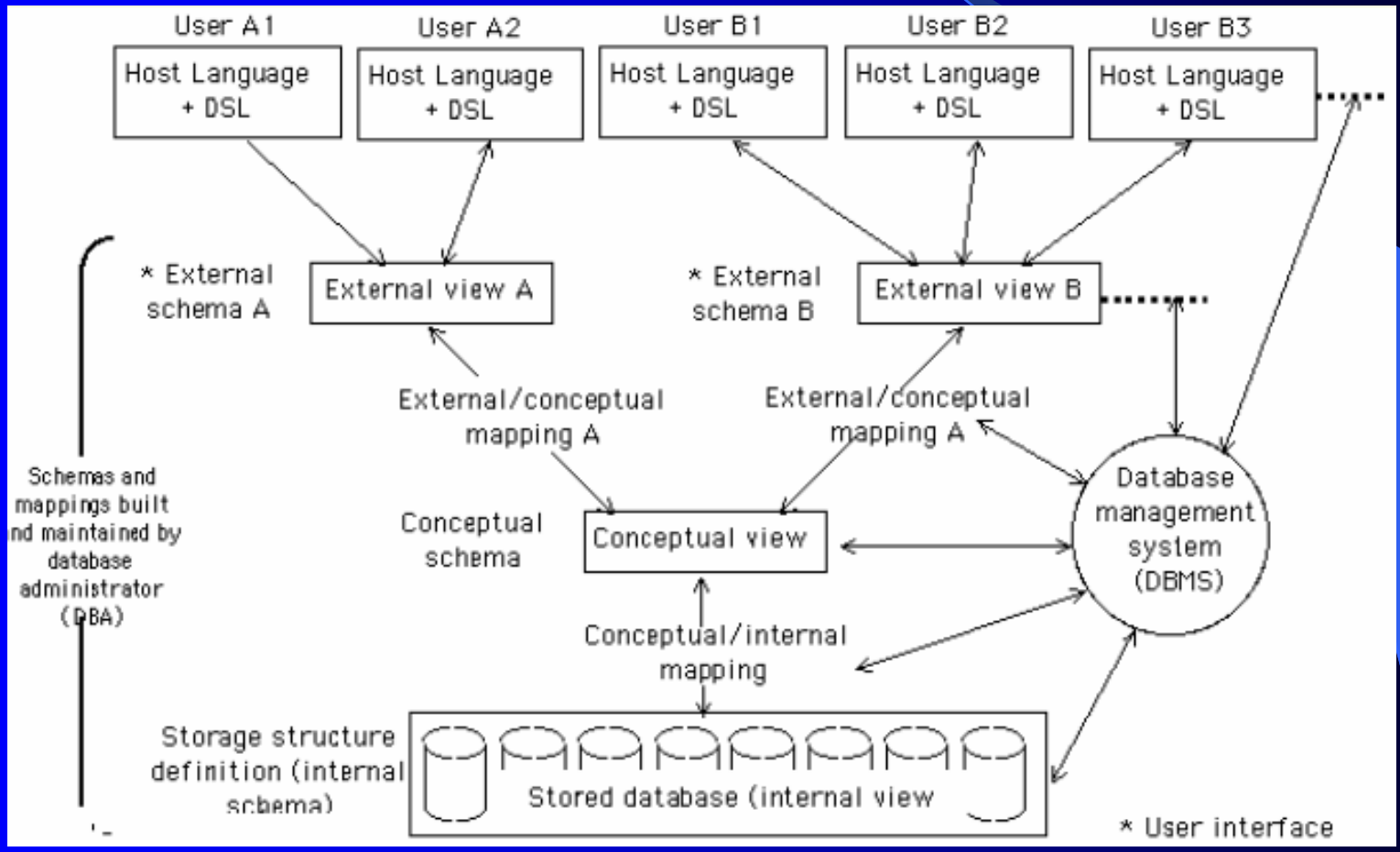
b) Distribucijom baze podataka povećava se raspoloživost i pouzdanost sistema. U slučaju ispada bilo kog računara u mreži, podaci na tom mestu postaju nedostupni korisnicima, ali su podaci na svim ostalim mjestima sačuvani i dostupni.

c) U distribuiranim sistemima se koristi tehnika repliciranja istih podataka na više lokacija u mreži. Zbog obrade manjih baza podataka, brzina obrade na pojedinim mjstima je veća u odnosu na brzinu koju bi imao sistem koji obrađuje veliku centralizovanu bazu.

Struktura DBMS



Opšta arhitektura SUBP tronivoska ANSI/SPARC



Struktura DBMS

- **DDL kompilator**

  Obrađuje definicije šema pripremljene na jeziku DDL i opise ovih šema smešta u katalog odakle ih mogu koristiti ostale komponente.

- **Run - Time procesor baze podataka**

  Rukuje pristupom bazi podataka u toku izvršenja programa. Izvršava operacije pretraživanja i ažuriranja nad bazom podataka. Za pristup disku koristi usluge sistema za upravljanje memorisanim podacima.

- **Procesor upita**

  Rukuje upitima visokog nivoa pri interaktivnom režimu rada. Obavlja leksičku i sintaksnu analizu upita i generiše poziv run-time procesora baze podataka koji izvršava traženu operaciju.

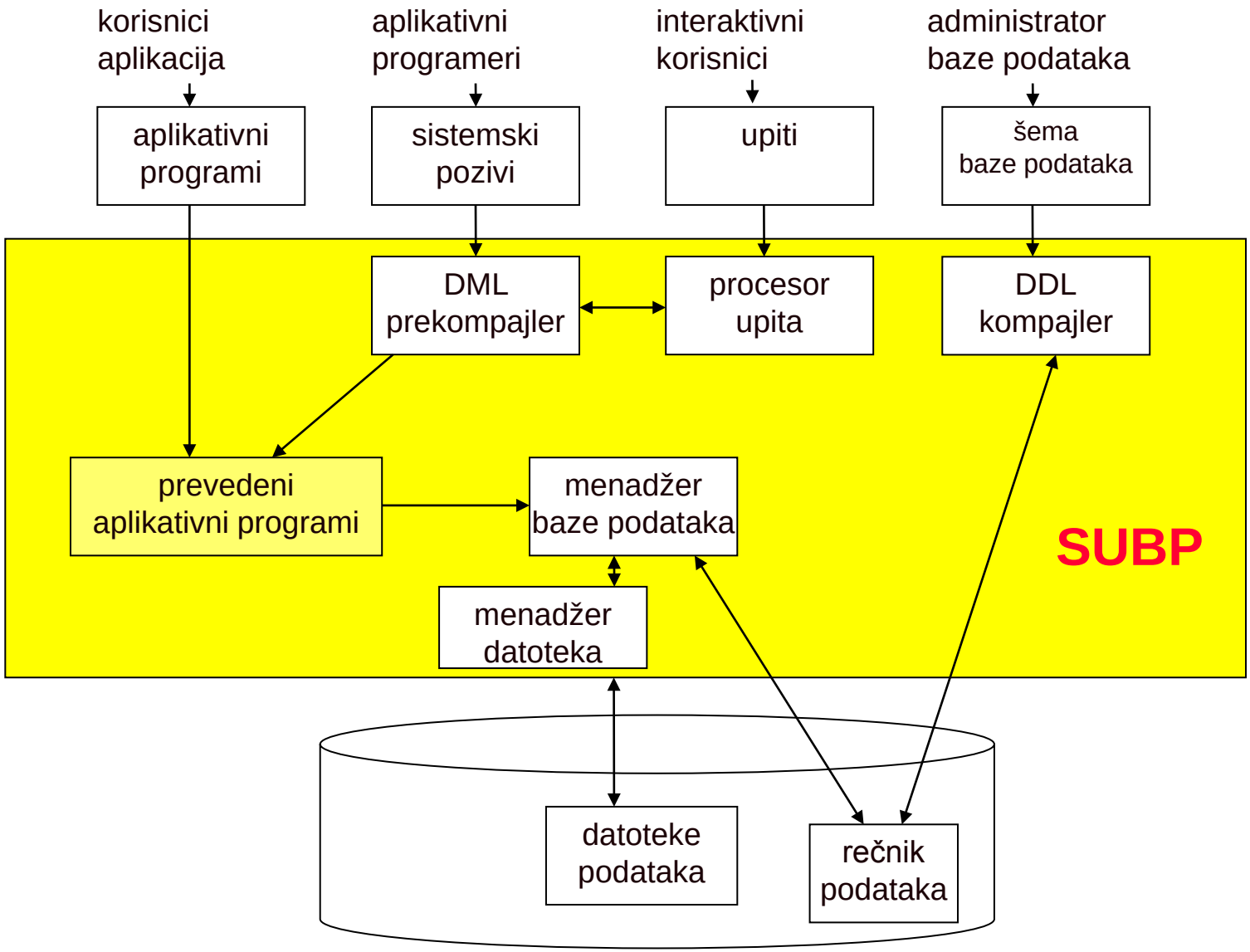
- **Predkompilator**

  Izdvaja DML naredbe iz aplikativnog programa koji je pisan na nekom host programskom jeziku i šalje u DML kompilator koji ih prevodi u objektni kod za pristup bazi podataka. Ostali deo programa se šalje host kompilatoru. Objektni kodovi iz ovih kompilatora se mogu linkovati i formirati transakcije za potrebe parametarskih korisnika. Ove transakcije se mogu pozivati iz formi i menija.

- **Katalog DBMS-a**

  Sadrži opis baze podataka koji obuhvata imena datoteka,elemenata podataka, detalje o načinu memorisanja svake datoteke, informacije o preslikavanju šema i ograničenja baze podataka.

korisnici


korisnici aplikacija
aplikativni programeri
interaktivni korisnici
administrator baze podataka
aplikativni programi
sistemski pozivi
upiti
šema baze podataka
DML prekompajler
procesor upita
DDL kompajler
prevedeni aplikativni programi
menadžer baze podataka
SUBP
menadžer datoteka
datoteke podataka
rečnik podataka

Jezici baze podataka

- DBMS obezbeđuje poseban jezik za svaku kategoriju korisnika baze podataka. Korisnicima su na raspolaganju sledeći jezici:

- Jezik za opis podataka (Data Description Language - **DDL**) Ako DBMS nema striktno odvojene nivoe, tada se DDL koristi za specificiranje konceptualne i interne šeme. Ako su konceptualni i interni nivo strogo odvojeni, tada se DDL koristi samo za specificiranje konceptualne šeme. DDL može biti ugrađen u neki programski jezik opšte namene ili može biti poseban jezik. U drugom slučaju DDL naredbe moraju imati takvu sintaksu da se mogu lako mašinski identifikovati. DBMS ima DDL kompilator koji prevodi DDL naredbe i kompiliran opis šema memoriše u katalogu baze podataka.

- Jezik za definisanje načina memorisanja podataka (Storage Definition Language -**SDL**) Ako su razdvojeni konceptualni i interni nivo, tada se SDL koristi za specificiranje interne šeme. Preslikavanje iz konceptualne u internu šemu može biti definisano u jednom od ova dva jezika.

- Jezik za definisanje pogleda (View Definition Language - **VDL**) Ako DBMS ima striktno razdvojena sva tri nivoa, tada se VDL koristi za specificiranje eksternih šema i njihovo preslikavanje u konceptualnu šemu.

- Jezik za rad sa podacima (Data Manipulation Language - **DML**). DML se koristi za pretraživanje, umetanje, brisanje i modifikaciju podataka u bazi podataka. Može biti neproceduralni ili proceduralni jezik. ***Neproceduralni*** DML je jezik visokog nivoa koji moze biti interaktivni, kada se DML naredbe unose direktno sa terminala, ili ugrađen u neki programski jezik (C,C++, PASCAL, COBOL, PL/I). Ako su DML naredbe ugrađene u neki programski jezik, tada moraju biti tako projektovane da ih DML kompilator može lako identifikovati. DML visokog nivoa koji se može interaktivno koristiti naziva se ***uptini jezik***. ***Proceduralni*** DML je jezik niskog nivoa koji mora biti ugrađen u neki programski jezik opšte namene. Takav jezik čita i upisuje u bazu podataka slog po slog. Programski jezik opšte namene u koji su ugrađene DML naredbe naziva se HOST jezik, a DML je tada njegov podjezik.

Intrfejsi DBMS-a

- DBMS obezbeđuje poseban interfejs za svaku kategoriju korisnika baze podataka. Na raspolaganju su sledeći interfesi:

- **- Interfejs baziran na meniju**
- DBMS za spregu sa korisnikom nudi **meni**. Pri formulaciji zahteva sistem vodi korisnika pomoću serije menija, tako da korisnik ne mora pamtiti komande i sintaksu upitnog jezika.
- **- Interfejs baziran na formama**
- Za spregu sa korisnikom DBMS nudi forme. Pri formulaciji zahteva korisnik treba samo da ispuni određena polja u formi. Neki DBMS-i imaju jezik za specifikaciju forme, koji pomaže programeru pri definisanju novih formi.
- **- Grafički interfejs**
- DBMS za spregu sa korisnikom nudi grafički prikaz. Šema baze podataka se prikazuje u obliku dijagrama i korisnik formira upit manipulišući elementima ovog dijagrama. Za izbor objekata sa prikazane šeme može se koristiti miš ili svetlosno pero.Grafički interfejs se često koristi u kombinaciji sa menijem.
- **- Interfejs na prirodnom jeziku**
- Ovaj interfejs prihvata zahtev formulisan na nekom prirodnom jeziku (na primer, engleskom) i pokušava da ga razume i obradi.
- **- Interfejs za parametarske korisnike**
- Za parametarske korisnike, koji imaju mali broj zahteva koji se ponavljaju, projektuje se poseban komandni jezik. Poželjno je da svaka komanda bude pridružena nekom funkcionalnom tasteru.
- **- Interfejs za administratora baze podataka**
- Administrator baze podataka ima privilegovan pristup bazi podataka, on kreira račune, postavlja parametre sistema, dodeljuje i oduzima pristupne privilegije, ažurira šeme. Za administratora baze podataka se projektuje poseban komandni jezik koji mu omogućava obavljanje ovih aktivnosti.

Elementi baza podataka

- ***OBJEKAT POSMATRANJA - ENTITET***
- **Proces po definiciji predstavlja stalnu promenu jedne ili više veličina u vremenu** u svetu koji nas okružuje (na primer temperatura vazduha, natalitet, bruto nacionalni dohodak, opterećenosti elektroenergetskog sistema, itd.). Veličine koje se ne menjaju u vremenu nisu interesantne sa stanovišta eksploatacije podataka. To su najčešće prirodne konstante kao što su; Ludolfov broj *pi*, ubrzanje zemljine teže *g*, dielektrična konstanta vakuuma EQ i tome slično. Njihovu vrednost je dovoljno poznavati samo jedanput, jer se ona ne menja u vremenu.
- Podaci su promjenjive ali diskretne a ne kontinualne veličine, pa postupak analize i praćenja procesa preko podataka (postupak sinteze baze podataka), zahteva provođenje neke vrste analogno-digitalne (A/D) konverzije. A kako se iz diskretne baze podataka nova informacija može dobiti samo obradom diskretnih podataka - i nova informacija je onda uvek diskretna veličina.
- Da bismo izveli pomenutu "A/D konverziju", i neki kontinualni proces opisali ograničenim brojem diskretnih vrednosti, prisiljeni smo da definišemo model-objekat koji predstavlja "digitalnu sliku" odnosno naše "viđenje" za nas interesantnog dela realnog sveta. Dobijeni rezultat naziva se onda model (delić realnog sveta, a u stručnoj literaturi naziva se još i entitet, objekat ili model-objekat.

Elementi baza podataka

- ***ENTITET ili objekat*****,** po svojoj prirodi može biti različit kao na primer:
- **do okruženja** (*član kolektiva, aparat, zgrada*)
- **apstraktni pojam** *(neka mera, nečije zvanje, boja,*)
- **događaj** (*udes, postupak upisa studenata*,)
- **asocijacija** *(kao što su polaznik- kurs, predmet- nastavnik, itd).*

- ***Obeležanje tipa entiteta***
- E(A1,A2,...,An), gde su E ime klase entiteta, a Ai (i=1,n) odabrani atributi entiteta E.
- Primer:
- RADNIK(IME, DATUM_ROĐENJA, ADRESA, TELEFON, SEKTOR, LD, ČLAN_POROĐ)
- RADNIK(IME,SEKTOR,LD)
- RADNIK(MATIČNI_BROJ_RADNIKA,MATIČNI_BROJ_GRAĐANA, IME_RADNIKA)

- Ovi tipovi entiteta modeliraju na različite načine entitet RADNIK.

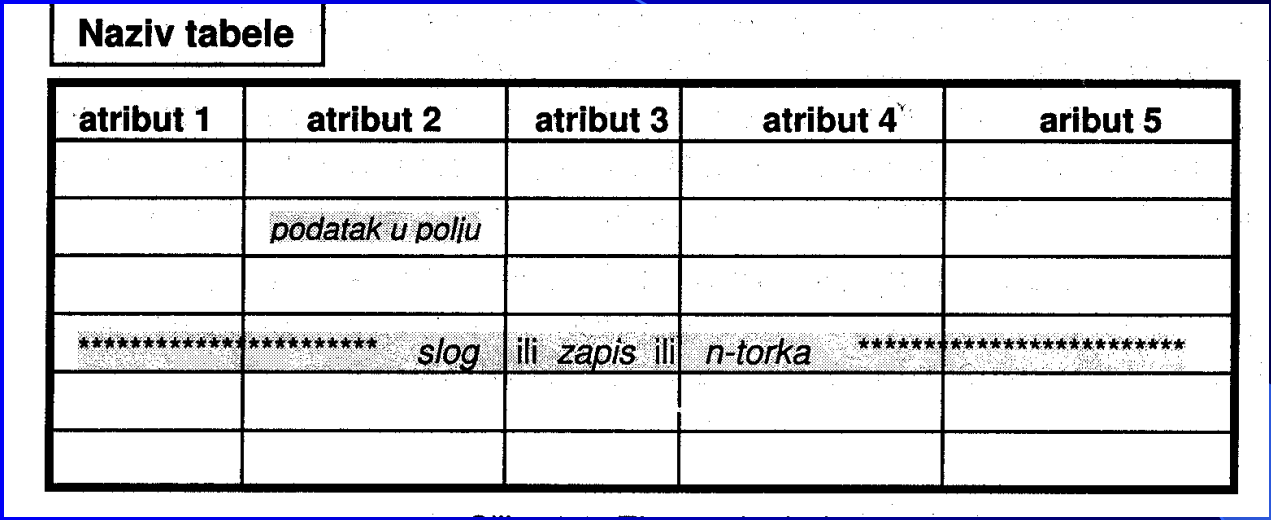
Elementi baza podataka

- ***Konkretizacija (pojava) tipa entiteta***
- To je skup podataka o određenom entitetu.

- Primer:
- Za tip entiteta RADNIK(IME,SEKTOR,LD), moguće pojave su:

- SAVA KRSTIĆ,INSTITUT,1500
- JOVAN RISTIĆ,TEST,850

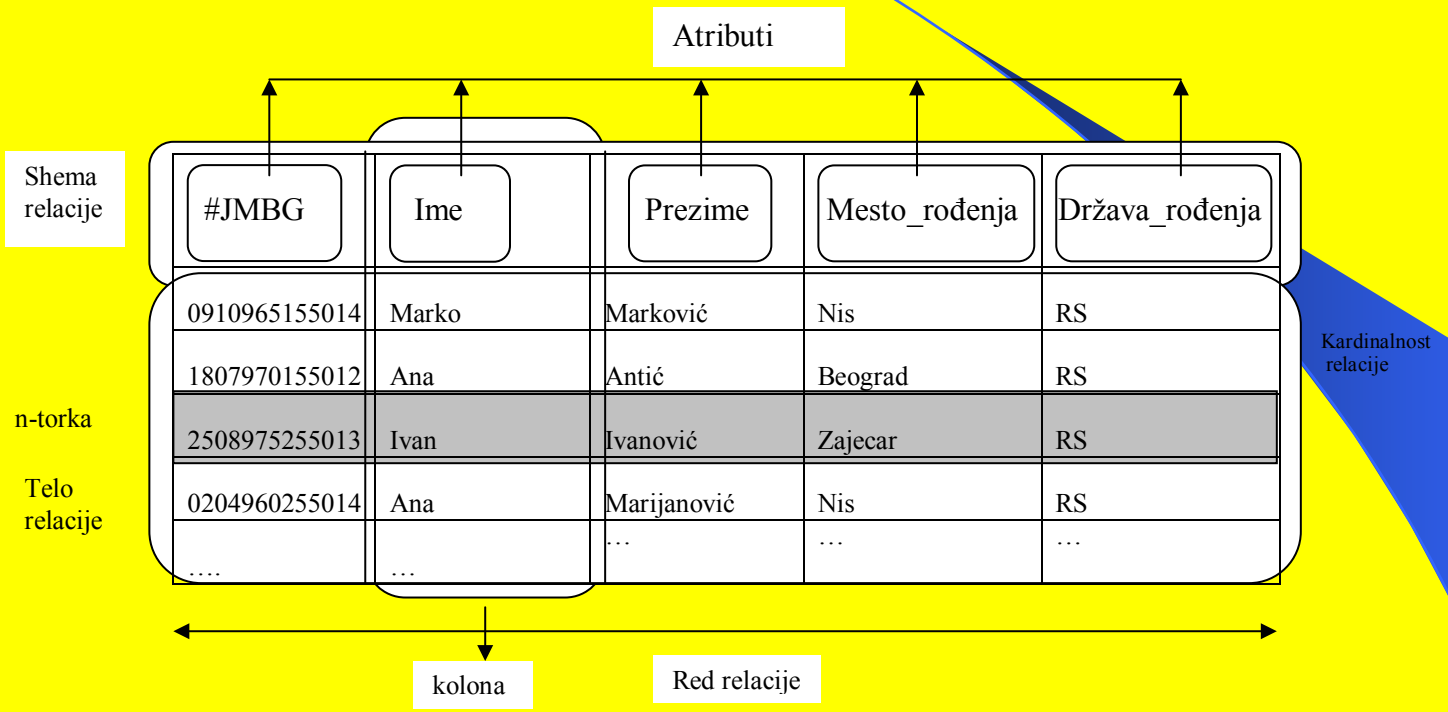
Elementi baza podataka



Svojstva objekata opisuju se preko **atributa.** Skup dozvoljenih vrijednosti koje može imati neki atribut naziva se **domen. Vodimo računa:**

- **Ako odaberemo premalo atributa,** model će biti jednostavan za predstavljanje i analizu, ali će mu verodostojnost biti mala.
- **A ako se model opiše mnoštvom atributa,** njegovoj verodostojnosti se neće moći prigovoriti, ali mu kompleksnost postaje takva da manipulacija podacima postaje teška ili neizvodljiva, a dobijene informacije konfuzne. Prema tome: ***Prepoznavanje prave mere pri modelovanju nekog procesa (izboru relevantnih atributa) jedan je od osnovnih zadataka projektanta, i to ne samo informacionog sistema.***

Elementi baza podataka



Elementi baza podataka

- ***Atribut (obeležje)***
- To je zajednička osobina, koju poseduju svi entiteti jedne klase.

- Primer:
- Atributi radnika: . . . . . Atributi automobila:
- - ime . . . . . - naziv
- - datum rođenja . . . . . - proizvođač
- - adresa . . . . . - kubatura
- - telefon . . . . . - vlasnik
- - sektor, gde radi . . . . . - registarski broj
- - lični dohodak . . . . . - datum registracije
- - članovi porodice
- Primer:
- Atributi studenata jednog fakulteta:
- - broj indeksa
- - ime
- - smer
- - godina
- - datum upisa
- - ocene

Elementi baza podataka

- ***Obeležavanje atributa***

- ***Stil A*** Obeležavaju se velikim slovom sa crticom ili znakom za podvlačenje kao poveznikom između reči.

- Primer:
- IME
- DATUM_UPISA
- BOJA_KOLA

- ***Stil B*** Obeležavaju se nizom reči, koje se pišu malim slovima, osim prvog slova svake reči. Nema delimitera između njih.

- Primer:
- Ime
- DatumUpisa
- BojaKola

Elementi baza podataka

- ***Vrednost atributa***
- Svaki entitet za svaki atribut ima određenu vrednost, odnosno POJAVU.
- Primer:
- Vrednosti atributa entiteta RADNIK su:
- - Ime = MIKA PERIĆ
- - Adresa = T.Rajića 33, 19350 Knjaževac
- - Datum Rođenja = 03/09/1962
- - Kućni Telefon = 019-41-123
- - Zanimanje = dipl.donosač pene za brijanje
- - Projekti = (P1,P2,P3)
- - Sektor = Majstor Đoka
- - LD = 450

Elementi baza podataka

- ***Jednovrednosni i viševrednosni atributi***
- Za određeni entitet atribut može imati jednu ill više vrednosti.
- Primer:
- - Atribut DATUM_ROĐENJA entiteta RADNIK može imati samo jednu vrednost
- - Atribut PROJEKTI entiteta RADNIK može imati više vrednosti (P1,P7,P16)

- ***NULL vrednost atributa***
- Neki atributi mogu biti bez vrednosti za neki entitet. Tada se kreira NULL vrednost.
- Primer:
- Ako radnik IVANA GOCIĆ nema telefon u stanu, atribut TELEFON za ovog radnika će imati NULL vrednost.

Elementi baza podataka

- *Domen atributa*
- To je skup vrednosti iz kojeg semantički definisani objekat uzima vrednosti.
- Obeležavanje domena:
- dom(Ime_atributa) = (domen1,domen2,...)

Primer:

- dom(BOJA_KOLA) = (bela,crvena,zelena)
- dom(OCENA STUDENTA) = (5,6,7,8,9,10)

- ***Konkretizacija atributa***
- To je konkretna vrednost koju atribut uzima iz svog domena da predstavi podatak o određenom entitetu. Samo konkretizacija atributa može predstavljati podatak.

Primer:

- Student PERA ILIĆ ima broj indeksa RE 5034/88. Tada je podatak Pera Ilić konkretizacija atributa IME_STUDENTA, a RE 5034/88 konkretizacija atributa BROJ_INDEKSA.

Elementi baza podataka

- ***Prosti i složeni atributi***
- Atribut je ***prost*** ***(etementaran - atomaran)*** ako se dalje ne može dekomponovati, ili ako se u konkretnoj situaciji ne dekomponuje na komponente koje čine atribute.
- Atribut je ***složen*** ***(kompozitan)*** ako je sastavljen iz niza elementarnih atributa.
- Konkretizacija etementarnog atributa je elementarni (prost) podatak, a konkretizacija složenog atributa je složeni (strukturni) podatak.
- Vrednost složenog atributa jednaka je konkatenaciji vrednosti prostih atributa, koji ga čine.
- Primer:
- Elementarni - prosti atributi
- OCENA__STUDENTA
- BOJA_AUTOMOBILA
- NAZIV_PROIZVODA
- Složeni atributi
- ADRESA_STUDENTA
- IME_STUDENTA
- DATUM_UPISA

Elementi baza podataka

- - Dekomponovanje složenih atributa

- ADRESA_STUDENTA (ULICA, BROJ_ZGRADE, BROJ_STANA, GRAD, POŠTANSKI_BROJ, ZEMLJA, KARAK_BROJ_ZEMLJE)

- IME_STUDENTA(IME,SREDNJE_SLOVO,PREZIME)
- DATUM_UPISA(DAN,MESEC,GODINA)

Elementi baza podataka

- ***Izvedeni atributi***
- To je atribut čije se vrednosti dobijaju primenom nekog algoritma na vrednosti drugih atributa (elementarnih, složenih, izvedenih). Vrednost izvedenog atributa je izvedeni podatak.
- Primer:

- Atribut SREDNJA_OCENA studenta se može dobiti primenom srednje vrednosti na atribute OCENA u entitetu STUDENT.

- Atribut BROJ_ZAPOSLENIH se izvodi iz broja radnika koji rade u sektoru.

Elementi baza podataka

- ***Ključ***
- To je atribut ili skup atributa koji je jedinstven za sve pojave određenog tipa entiteta. Svaki tip entiteta poseduje bar jedan ključ. **Ključ** K relacije R je podskup atributa od R, koji ima sledeća svojstva:
  - 1. *Vrednosti atributa iz K jednoznačno određuju n-torku u R.*
  - 2. *Ako izbacimo iz K bilo koji atribut, tada se narušava 1. svojstvo*
- Budući da su sve *n*-torke u R međusobno različite, K uvek postoji, jer skup svih atributa zadovoljava svojstvo 1. Izbacivanjem suvišnih atributa dolazi se do podskupa koji zadovoljava svojstvo 2.
- Jedan tip entiteta može imati više ključeva. To su ključevi kandidati. Jedan odključeva kandidata je ***primarni*** a ostali su ***sekundarni***.
- Primer:
- U tipu entiteta RADNIK ključ može biti IME, MATIČNI_BROJ_RADNIKA, MATIČNI_BROJ_STANOVNIKA.

Elementi baza podataka

- ***Obeležavanje primarnog ključa***

  Obeležavanje primarnog ključa može biti na dva načina:

- - Primarni ključ u tipu entiteta se podvlači kontinualnom linijom
- - Iza imena primarnog ključa se stavlja povisilica (#).
- Primer:

- RADNIK(MATIČNI_BROJ_RADNIKA,MATIČNI_BROJ_STANOVNIKA,IME)

- RADNIK(MATIČNI_BROJ_RADNIKA#, MATIČNI_BROJ_STANOVNIKA, IME)

- *Ključni i sporedni atributi*

- Svi atributi koji pripadaju ključu tipa entiteta nazivaju se ključnim atributima, dok su ostali sporedni atributi. Atributi, koji pripadaju primarnom ključu nazivaju se primarnim, a ostali atributi su tada neprimarni.

Elementi baza podataka

- ***RELACIJA:***
- **BINARNA RELACIJA** R, koja se uvodi u skup S={Si[i=1,n]} ili između skupova S1={Si[i=1,n1]} i S2={Si[i=1,n2]} , predstavlja pravilo povezivanja (stavlja u odnos) elemenata skupa S ili skupova S1 i S2. Binarna relacija R je podskup Dekartovog proizvoda SxS ili S1xS2.
- *Binarna relacija* R između skupova S1 i S2 definiše dva preslikavanja, R:S1->S2 i R:S2->S1. Svojstvo ovih preslikavanja koje je posebno značajno za modeliranje podataka je ***kardinalitet***. Kardinalitet je broj objekata skupa S1 koji je u vezi sa objektima iz skupa S2 i obrnuto.
- ***N-ARNA RELACIJA*** Rn, između skupova S1,S2,...,Sn predstavlja pravilo povezivanja redom elemenata skupova S1,S2,...,Sn. Rn je podskup Dekartovog proizvoda S1xS2x...xSn.
- U isti skup podataka se može uvesti proizvoljan broj relacija.

Elementi baza podataka

- ***Graf relacija***

- Ako je binarna relacija R definisana u skupu S, tada se uređeni par G=(S,R) naziva ***graf relacije***. Elementi skupa S={s_1 ,s_2,...,s_n} su čvorovi grafa, a elementi skupa R= {(s_i,s_j) } grane grafa. Ako je par (s_i,s_j) iz skupa R, tada čvor si povezujemo sa čvorom sj sa orijentacijom grane od s_i ka s_j.
- Primer:
- S = (Jugoslavija, Austrija, Mađarska, Rumunija)
- Relacija (R) je "je sused". Graf relacije (R) je:
- r = { (Jugoslavija, Austrija), (Jugoslavija, Mađarska), (Jugoslavija, Rumunija), (Austrija, Mađarska), (Mađarska, Rumunija) }
- Primer:
- S1 =(Niš, Beograd, Skoplje, Bar, Sarajevo)
- S2 = (Srbija, Crna Gora, Makedonija)
- Relacija (R) je "nalazi se u". Graf relacije (R) je:
- r {(Niš, Srbija), (Beograd, Srbija), (Skoplje, Makedonija), (Bar, Crna Gora) }

Elementi baza podataka

- ***Primer:***

- *Pretpostavimo, da u sektoru za urbanizam, žele da imaju informacije o svim ulicama. Entitet, je prema tome ULICA, a relevantni atributi kojima se opisuje ulica mogli bi biti: naziv, dužina, širina, vrsta kolovoza, godina izgradnje, broj kuća itd., dok podatak o promeni temperature gornjeg sloja asfalta, u ovome slučaju, neće biti relevantan. Entitet ULICA šematski možemo predstaviti kao:*
- *ULlCA < naziv, dužina, širina, vrsta_kol., god_izg., broj_kuća... >*

Elementi baza podataka

Naziv	Dužina (km)	Širina (m)	Vrsta kolovoza	Godina izgradnje	Broj kuća
SimeMilutinovića	0.56	12.3	asfalt	1908	231
Jove Jovanovića	0.20	7.5	granitne kocke	1898	56
Vuka Karadžića	1.75	6.00	asfalt, makadam	1864	442
Zeke Buljubaše	0.08	5.50	kaldrma	1888	12

Slika 1.2 Primjer tabele "ULICA"

- Veličina, odnosno dužina, tabele u kojoj se nalaze svi podaci o svim ulicama u opštini, zavisi od broja ulica i mora imati onoliko redova, slogova, ili n-torki koliko ima ulica u toj opštini. Dakle, broj redova jednak je broju pojedinačnih primeraka - objekata nekog entiteta.
- Izgradnjom nove ulice, novi podaci za tu ulicu se onda jednostavno dopisuju u već postojeći spisak, u postojeću tabelu. Prema tome, svaka tabela mora da ima definisano:

- ime ili naziv tabele,
- spisak atributa i
- vrednosti atributa, to jest podatke upisane u polja.

Elementi baza podataka

- ***Primer:***
- Neka druga tabela, u istom sektoru opštine, može da sadrži podatke o nekom drugom entitetu, na primer, stambenim zgradama. Nazovimo tu tabelu **ZGRADA,** a skup za nju relevantnih atributa mogao bi biti:

ZGRADA *< ulica, kućni_br., god_izg., br_sprat., br_stanova... .>*

- ***Ulica*** je, kao što vidimo, u datoteci **ZGRADA** atribut, dok je u prethodnom primeru tabela imala taj naziv. *Prema tome atribut u nekoj tabeli može biti naziv neke druge tabele*. O tome treba voditi računa kada se neki proces opisuje podacima, da ne bi došlo do zabune. Izbor imena atributa i tabela diktiran je prije svega našim željama, interesima i "pogledima" (view) koje na realni svijet hoćemo da "bacimo" preko podataka koje posedujemo, odnosno koje informacije očekujemo da iz njih možemo dobiti.
- Izbor imena tabele nije kritičan i u praksi se proporučuje da izabrano ime tabele asocira na prirodu procesa u entitetu koji se prati podacima. Međutim, prilikom izbora imena atributa treba biti obazriv, jer osnovno pravilo kaže da: ***atribut mora biti tako odabran da se može i mora opisati samo jednim elementarnim (atomarnim) podatkom.***

Pristup podacima

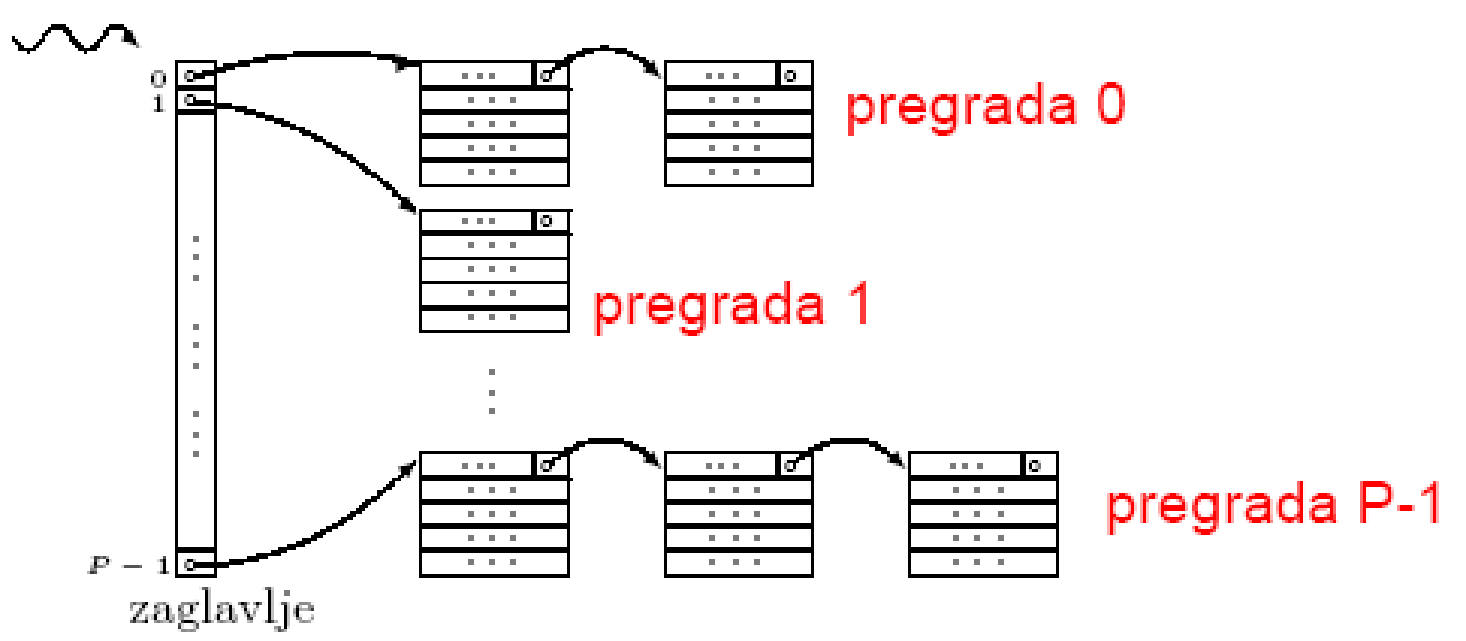
- Osim osnovnih datoteka koje odgovaraju pojedinim relacijama, fizička građa relacione baze može sadržati i dodatne pomoćne datoteke koje ubrzavaju pretraživanje i uspostavljanje veza među podacima.
- Tipični primeri takvih datoteka su:
  - Hash datoteka
  - Datoteka s indeksom

Hash datoteka

- Zapise datoteke smeštamo u P pregrada, označenih rednim brojevima 0, 1, 2, . . . , P − 1. Svaka pregrada se sastoji od jednog ili više blokova. Zadaje se tzv. hash funkcija (h), koja daje redni broj h(k) pregrade u kojoj se treba nalaziti zapis sa vrednošću ključa k.

- Skup mogućih vrednosti ključa obično je znatno veći od broja pregrada. Važno je da (h) uniformno distribuira vrednosti ključa na odgovarajuće pregrade. Tada se neće dešavati da se pregrade neravnomerno pune.

Hash datoteka



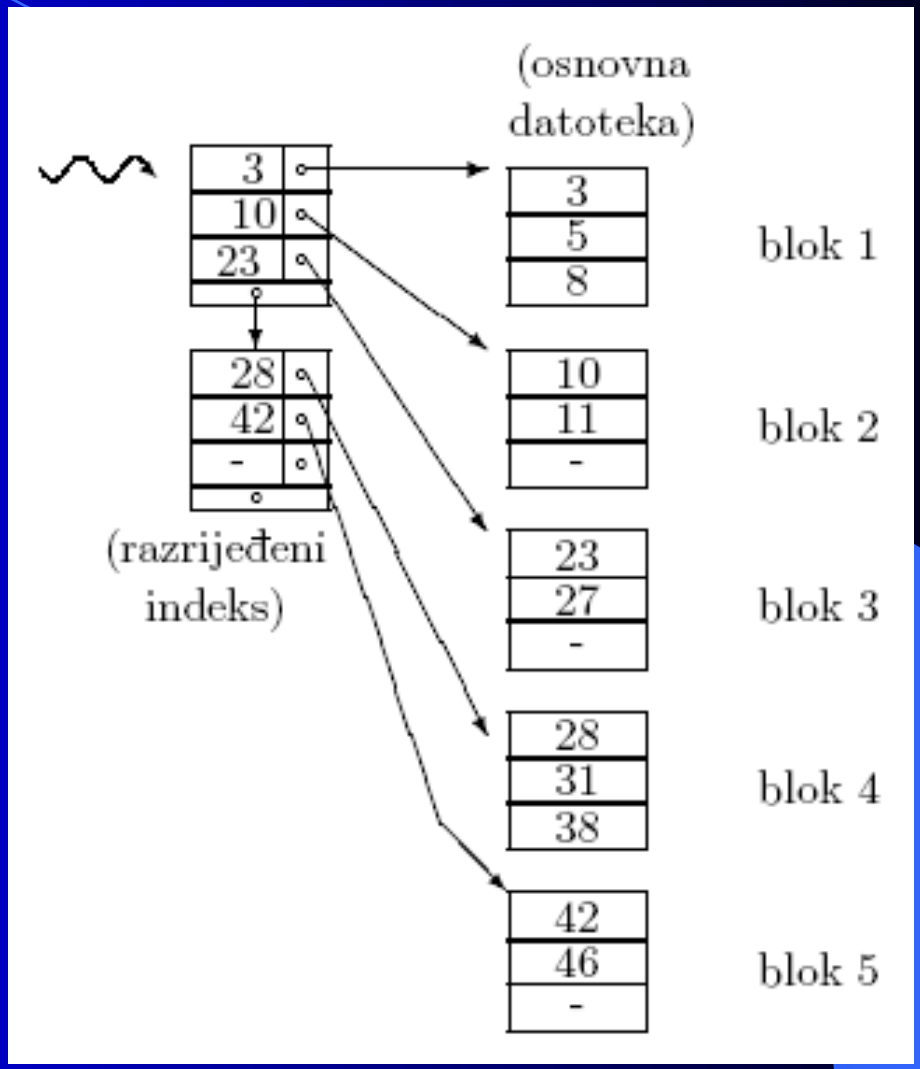
- Zapis sa zadanom vrednošću ključa k tražimo tako da izračunamo h(k), i sekvencijalno pretražimo h(k)-ti pregradu. Ako je hash funkcija zaista uniformna, i ako je broj pregrada dobro odabran, tada ni jedna od pregrada nije suviše velika. Pristup na osnovu ključa zahteva svega nekoliko čitanja bloka (obično oko 2 čitanja).

Hash datoteka

- Kao primer, promatramo zapise o studentima fakulteta koje želimo spremati u hash datoteku sa 1024 (= 2^{10}) pregrada. Definicija tipa zapisa u jeziku C izgleda ovako:
  - typedef struct { int SNO; // Matični borj
  - char SNAME[20]; // Ime
  - enum {1,2,3,4} LEVEL; // Godina
  - enum {M,F} GENDER; // Pol
  - } STUDENT;
- Podelimo 10 bitova u rednom broju pregrade na sledeći način: 4 bita za SNO, 3 bita za SNAME, 2 bita za LEVEL, 1 bit za GENDER.
- Zadajemo sledeće hash funkcije, gde su v1, v2, v3, v4 vrednosti za SNO, SNAME, LEVEL, GENDER respektivno:
  - h1(v1) = v1 % 16 (ostatak kod deljenja sa 16),
  - h2(v2) = (broj znakova u v2 koji su različiti od blanko) % 8,
  - h3(v3) = v3 − 1,
  - h4(v4) = (0 za v4 jednak M, 1 inače).
  - Tada je redni broj pregrade za zapis (58651, Smith, 3, M) zadan sa $(1101|101|10|0)_2 = (748)_{10}$.

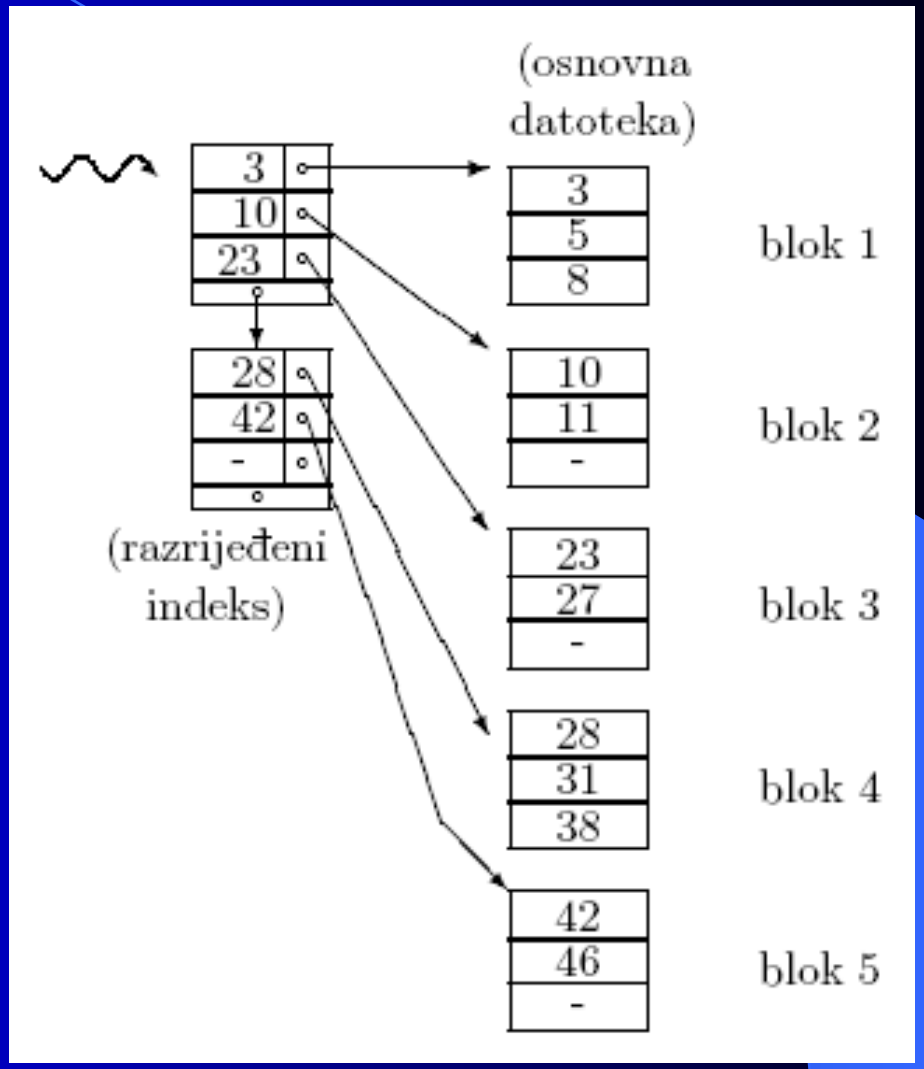
Datoteka sa indeksom

- Indeks je mala pomoćna datoteka koja olakšava traženje u velikoj (osnovnoj) datoteci. Izložićemo dve varijante datoteke sa indeksom.
  - **Indeks sekvencijalna organizacija** zahteva da zapisi u osnovnoj datoteci budu sortirani po vrednostima ključa (na primer rastući). Blokovi ne moraju biti sasvim popunjeni. Dodajemo tzv. razređeni indeks. Svaki zapis u indeksu odgovara jednom bloku osnovne datoteke i oblika je (k, a), gde je k najmanja vrednost ključa u dotičnom bloku, dok je a adresa bloka.



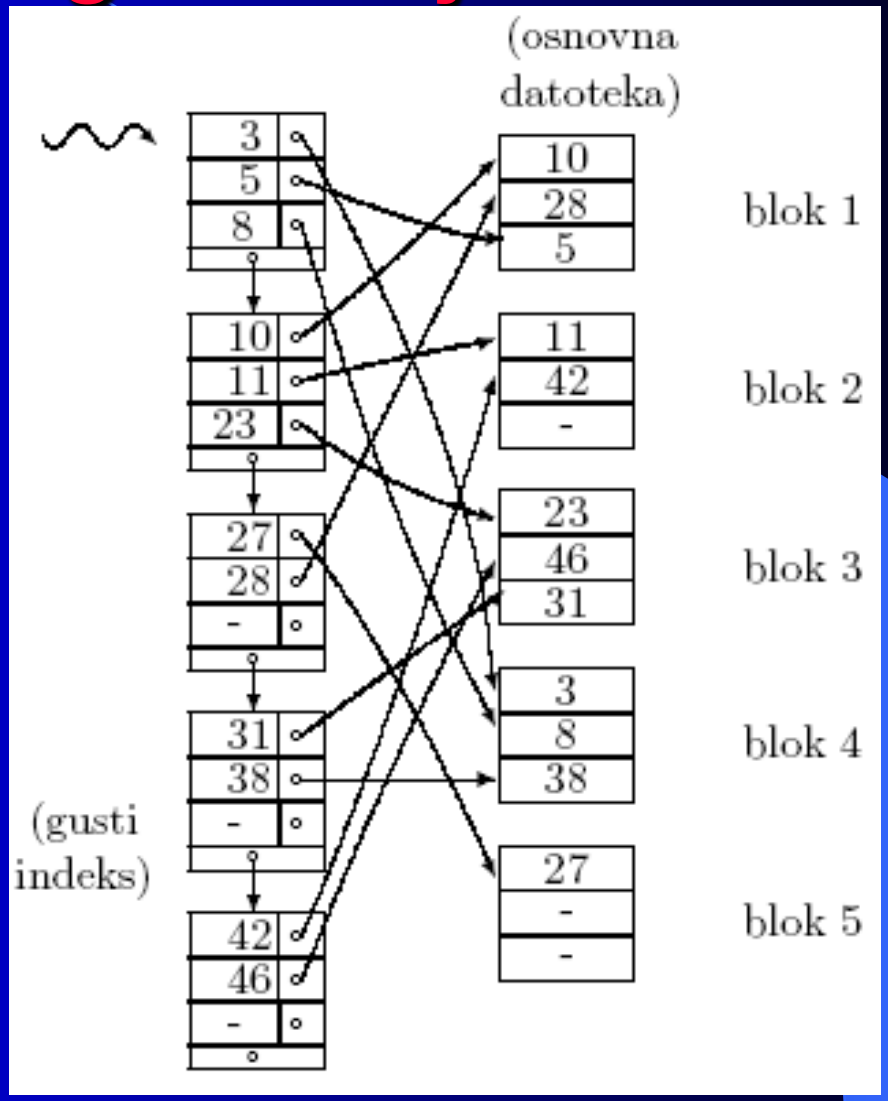
Datoteka sa indeksom - indeks sekvencijalna organizacija

- Da bi u osnovnoj datoteci našli zapis sa zadanom vrednošću ključa k_0, čitamo indeks i tražimo najveći k_1 takav da je $k_1 \leq k_0$ i pritom par (k_1, a_1) postoji u indeksu. Zatim učitamo i pretražimo blok sa adresom a_1. Pristup po primarnom ključu zahteva (u najgorem slučaju) onoliko čitanja bloka koliko ima blokova u indeksu +1.

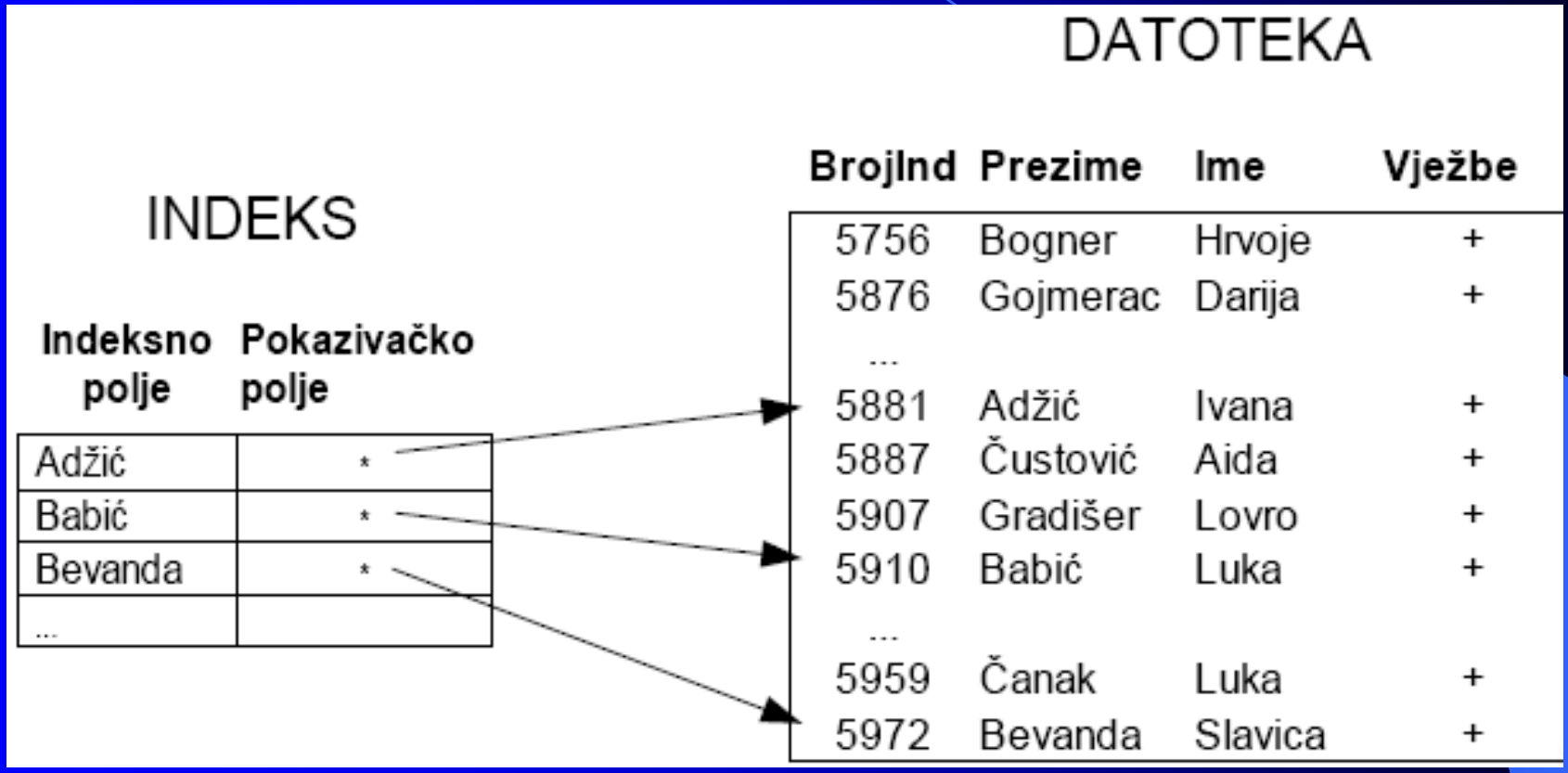


Datoteka sa indeksom - indeks direktna organizacija

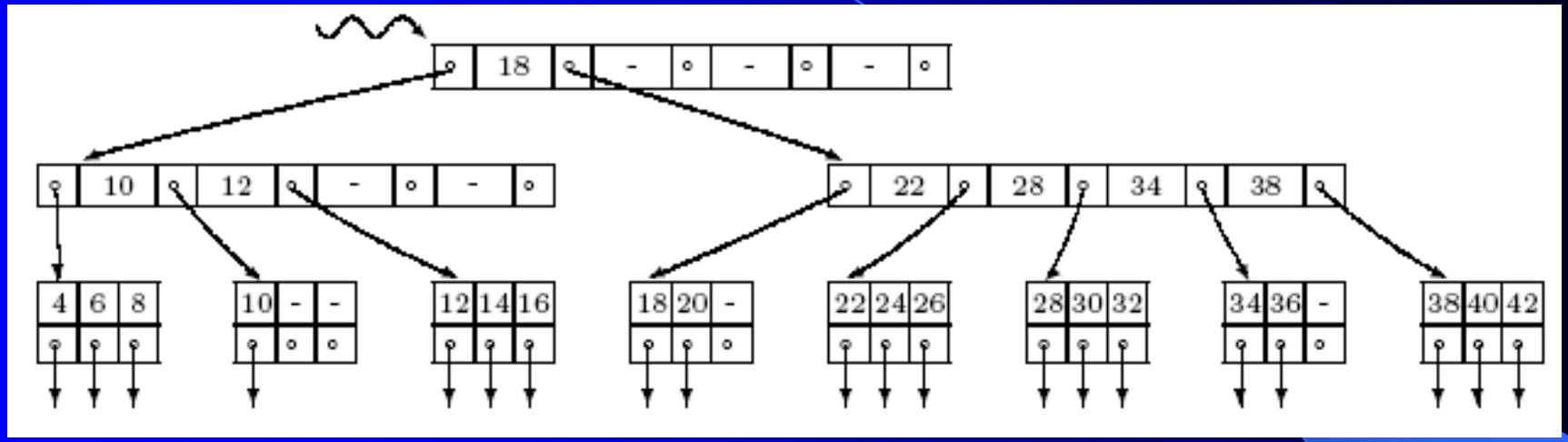
- Dopušta da zapisi u osnovnoj datoteci budu upisani u proizvoljnom redosledu. Dodajemo tzv. gusti indeks. Svaki zapis u indeksu odgovara jednom zapisu osnovne datoteke i oblika je (k, a), gdje je k vrednost ključa u dotičnom zapisu osnovne datoteke, a je pointer adresa zapisa iz osnovne datoteke. Za razliku od nesortirane osnovne datoteke, indeks je sortiran po ključu.



Datoteka sa indeksom - **indeks direktna organizacija**



Datoteka sa indeksom B-stablo



- **Traženje.** Želimo u B-stablu pronaći par (k, a), gde je k zadana vrednost ključa, a je pointer adresa traženog zapisa u osnovnoj datoteci. U tu svrhu sledimo put od korena do lista koji bi morao sadržavati k. To se radi tako da redom čitamo unutrašnje čvorove oblika (a_0, k_1, a_1, k_2, a2, . . . ,k_r, a_r), i uporedimo k sa k_1, k_2, . .., k_r. Ako je ki ≤ k < ki+1, dalje čitamo čvor na koga ukazuje ai. Ako je k < k1, dalje čitamo čvor sa adresom a_0. Ako je k ≥ k_r, koristimo adresu a_r. Kad nas taj postupak konačno dovede u list, tražimo u njemu zadani k.

Invertovana datoteka

- Uvodimo **sekundarni indeks** - to je mala (pomoćna) datoteka koja olakšava pristup zapisima velike (osnovne) datoteke na osnovu podatka koji nije ključ. Sekundarni indeks za podatak A sastoji se od zapisa oblika

  (v, {p1, p2, p3, . . .}), gde je

- v vrednost za A,
- a {p1, p2, p3, . . .} je skup pointera na zapise u osnovnoj datoteci u kojima A ima vrednost v.

Ukoliko postoji ovakav indeks, kaže se da je osnovna datoteka invertovana po podatku A. Datoteka koja je invertovana po svakom od svojih podataka zove se potpuno invertovana datoteka.

Invertovana datoteka

- Kao primer, posmatramo tabelarni prikaz datoteke nastavnika na fakultetu. Ako invertujemo tu datoteku po svakom podatku osim IME, dobijamo gusti primarni indeks za ključ MAT_BR i tri sekundarna indeksa za ODELJ, STAROST i ZVANJE. Indeks za STAROST sadrži intervale umesto pojedinačnih vrednosti.

pointer	*MAT_BR*	*IME*	*ODELJ*	*STAROST*	*ZVANJE*
a1	12453	Matić, R.	Matematika	35	Dr.sc.
a2	16752	Pavić, D.	Matematika	26	Dipl.inž.
a3	27453	Milošević, B.	Računarstvo	37	Dr.sc.
a4	34276	Dimić, J.	Fizika	55	Dr.sc.
a5	37564	Katić, K.	Računarstvo	45	Dipl.inž.
a6	43257	Radić, M.	Matematika	32	Mr.sc.
a7	45643	Janić, Z.	Fizika	24	Dipl.inž.
a8	56321	Popović, G.	Računarstvo	34	Dr.sc.
a9	57432	Simić, J.	Matematika	52	Dr.sc.

Invertovana datoteka

MAT_BR	*pointer*
12453	a1
16752	a2
27453	a3
34276	a4
37564	a5
43257	a6
45643	a7
56321	a8
57432	a9

intervali

Pointer za zapis	*STAROST*
a2,a7	21-30
a1,a3, a6 ,a8	31-40
a5	41-50
a4,a9	51-65

? STAROST=23

ODELJ	*Pointer za zapis*		*ZVANJE*	*Pointer za zapis*
Fizika	a4,a7		Dipl.inž.	a2,a5,a7
Matematika	a1,a2, a6,a9		Mr.sc.	a6
Računarstvo	a3,a5, a8		Dr.sc.	a1,a3,a4, a8, a9

Pristup na osnovu bilo kojeg podatka ostvaruje se tako da u odgovarajućem indeksu pronađemo zadanu vrednost podatka, pročitamo popis pointera, i za svaki od pointera pročitamo zapis u osnovnoj datoteci. Invertovana organizacija omogućuje i brz odgovor na pitanja o postojanju ili broju zapisa sa zadatim svojstvima.