Cvetana Krstev

Baze podataka i bibliotečki informacioni sistemi

1 Uvod u baze podataka

1.1 Uvod u baze podataka

1.1.1 Šta su baze podataka?

Teško je dati egzaktnu definiciju ovog pojma. Jedna veoma opšta definicija bi mogla da glasi:

Definicija 1.1: Baze podataka predstavljaju skup informacija organizovanih s nekim zajedničkim ciljem.

Nije od velikog značaja kakva podrška postoji za prikupljanje i skladištenje podatka (papir, datoteke,...) – sve dok se podaci prikupljaju i skladište na neki organizovan način i sa specifičnim ciljem može se govoriti o bazama podataka.

Još preciznije, pod bazama podataka ćemo podrazumevati strukturirani skup podataka koji omogućava takvo skladištenje velike količine informacija koje olakšava njihovu eksploataciju (dodavanje, ažuriranje, pretraživanje). Naravno, danas, kada se govori o bazama podataka, misli se prvenstveno automatizovane baze podataka.

1.1.2 Primer neautomatizovane baze podataka

Jedan primer neautomatizovane baze podataka je telefonski imenik na okretanje (engl. *roll-a-dex*) u kome je svako ime (osobe ili preduzeća) na zasebnoj kartici. Lako se dodaje kartica na pravo mesto, lako se menja podatak (zamenjuje kartica), lako se pronalazi željeni broj jer su svi podaci uvek uređeni (Videti Slika 1-1).

Definicija 1.2 Automatizovane baze podataka predstavljaju organizovani skup podataka koji je zabeležen na nekom računarskom nosicu podataka; ovaj skup podataka prestavlja informacije o realnom svetu koje neka zajednica korisnika može po potrebi da menja i pretražuje.

Slika 1-1 Neautomatizovana baza podataka - (a) telefonski imenik na okretanje; (b) primer kartice za ovakav imenik - firme u vidu vizit karata proizvode ovakve standardizovane kartice, spremne za umetanja.

1.1.3 Primer

Razmotrimo potrebu za organizovanjem podataka na primeru jednog preduzeća koje se bavi autobuskim prevozom. Pretpostavimo da preduzeće raspolaže velikim brojem autobusa različitih proizvođača i da održava veliki broj međugradskih linija. Među potencijalnim korisnicima podataka ovakvog preduzeća su različite službe unutar samog preduzeća, zatim, međugradske autobuske stanice i sami putnici. Svaki od ovih korisnika ima potrebu za određenim podacima pa su pitanja koja oni mogu postaviti raznovrsna.

Putnik, kao korisnik, želi da izvrši rezervaciju sedišta u autobusu na određenoj liniji za određeni datum. Služba koja planira raspoređivanje autobusa po linijama želi da zna kako su autobusi popunjeni na određenim linijama i koji su vozači na raspolaganju. Vozači žele da dobiju raspored svojih obaveza. Služba održavanja zahteva da zna kada je odreženi autobus prošao tehnički pregled.

Odgovori na sve ove raznorodne zahteve su sadržani u istom skupu podataka koji na podesan način opisuje autobuske linije, raspoložive autobuse, vozački kadar, itd. Ali za pojedinačne korisnike nisu od značaja svi ovi podaci: putniku je svejedno da li će vozač biti Petrović ili Pavlović. Vozaču je svejedno koje će putnike voziti od Beograda do Subotice. Finansijskoj službi je od značaja koliki su prihodi i rashodi preduzeća tokom

određenog perioda. Direktor preduzeća, pak, želi da analizira rentabilnost linija.

Neki primeri mogućih zahteva korisnika ovih podataka su:

- rezervacija: „Ima li slobodnih mesta na liniji Beograd - Kraljevo danas u 10 h?“, „Rezervisati tri sedišta u autobusu Beograd - Kraljevo za narednu sredu u 10 h“ i „Kada sve ima autobusa iz Beograda za Kraljevo autoputem?“
- posada autobusa: „Koji vozači su vozili na liniji Beograd - Kraljevo - Beograd u oktobru prošle godine“ ili „Na kojim je sve linijama vozio vozač Petrović tokom oktobra prošle godine?“
- autobus: „Kada je poslednji put autobus broj 333 prošao tehnički pregled?“ ili „Prikaži autobuse u voznom parku uređene po broju mesta za sedenje i godini proizvodnje“, itd.

Služba koja bi pružala odgovore na sva ova pitanja može biti ustrojena i bez računarske podrške. To podrazumeva da je izvestan broj radnika neprekidno zaposlen na proizvodnji papirnih dokumenata koji predstavljaju odgovore na gore postavljena pitanja. Ovakav prilaz je, bez sumnje, neefikasan, a posebno kada je u istom trenutku potrebno proizvesti veći broj dokumenata nego što ima raspoloživih radnika. Otuda proističe potreba za informatizacijom rada ovakvog preduzeća.

1.1.4 Problem koherentnosti podataka

Baze podataka se stvaraju jer postoji potreba da se prikupe podaci između kojih postoje neke veze s ciljem da se pronađu informacije koristeći kriterijume za pretraživanje koji se oslanjaju na sadržaj tih informacija. Uzmimo za primer jednu bazu podataka o muzičkim albumima u kojoj se pamte za svaki album sledeći podaci: žanr kojem album pripada, umetnik i naziv albuma. Izvod iz jedne takve baze je predstavljen u sledećoj tabeli.

Žanr	**umetnik**	**naslov**
alternativni rok	les Wampas	Les Wampas vous aiment
alt. rok	Les Wampas	Tutti frutti
alternativni rok	Wampas	Les bottes rouges
alt. rok	Les Sheriff	Les 2 Doigts Dans La Prise
folk-rok	Joan Osborn	Righteous Love
folk/rok	Leonard Cohen	Songs from a room

Uslov *sine qua non* koji garantuje mogućnost i kvalitet pronalaženja podataka prema sadržaju je koherentnost tih podataka. Podaci predstavljeni

ovom tabelom su u tom smislu neadekvatni i stvaraju više problema prilikom konsultovanja. Na primer, grupa *Les Wampas* je u bazi podataka predstavljena na tri različita načina: *Les Wampas*, *les Wampas*, *Wampas*. Ako bi se tražili podaci o albumima grupe *Les Wampas* koristeći tu nisku kao ključ pretrage dobio bi se kao odgovor samo album *Tutti frutti*, iako u bazi postoje tri albuma ove grupe. Na sličan način, žanrovi *alternativni rok* i *folk-rok* su predstavljeni na dva načina u bazi.

Koherentnost podataka je ključni problem baza podataka. Prvi i najvažniji odgovor za rešavanje ovog problema je da se što je moguće više ograniči redundantnost informacija, tj. njihovo nepotrebno ponavljanje. Da bi se to postiglo moguće je primeniti različite metode s kojima ćemo se mi upoznati. Ali da bi se te metode mogle primeniti važna je *konceptualna šema* baze podataka. Etapa modeliranja baze podataka u koji je utvrđe ova šema je, prema tome, veoma bitna.

1.1.5 Sistem za upravljanje bazama podataka

Koherentnost podataka može da bude ugrožena u fazi ubacivanja novih podataka ili modifikacije postojećih podataka, a takođe i zbog kvara računara ili nestanka struje ili zbog istovremenog pristupa, tj. menjanja, podataka. Rešavanje svih ovih problema nije, srećom, prepušteno onima koji osmišljavaju, tj. projektuju, bazu podataka već se ta zaduženje prenose na *sistem za upravljanje bazama podataka*. Ovaj sistem je, u stvari, kolekcija programa koji se brinu o upravljanju i pristupu bazi podataka. Sistem za upravljanje karakteriše *model za opis podataka* koji on podržava. Taj model može da bude hijerarhijski, mrežni, relacioni ili objektni. Model koji se prirodno nametnuo u poslednjih tridesetak godina je relacioni model, koji se poslednjih godina nadograđuje objektnim pristupom. Sistemi za upravljanje bazama podataka koji podržavaju relacioni model koriste jezik SQL – *Standard Query Language* – koji predstavlja standardizovani pristupni (upitni) jezik za relacione baze podataka.

1.2 Modeli baza podataka

1.2.1 Malo o istorijatu

Različiti modeli podataka razvijeni radi strukturiranja informacija u bazama podataka zasnivaju se na teorijskim matematičkim principima primenjenim na informatička istraživanja.

Prvi model koji je pokušavao da reši problem redundantnosti podataka je razvila firma IBM, u okviru projekta Apollo (svemirski program NASA-e) tokom 60-tih godina prošlog veka. Radi se o *hijerarhijskom modelu* koji ćemo ukratko predstaviti kasnije.

Krajem 60-tih godina predložen je *mrežni model* kao proširenje hijerarhijskog modela. Prve specifikacije ovog modela je objavila krajem 60-tih godina radna grupa DBTG (*Data Base Task Group*) američke organizacije CODASYL (*Conference On Data Systems Languages*). Autor ovog modela, Charles William Bachman, je za svoj teorijski doprinos razvoju modela baza podataka dobio prestižnu Tjuringovu nagradu za 1973. godinu.

U toku ovog perioda računari su se vrlo dinamično razvijali što se tiče računarske snage, rasprostranjenosti i cene. To je omogućilo da informatički modeli i programski jezici dostignu takav nivo apstraktnosti koji ih čini nezavisnim od arhitekture specifičnog računara. U ovakvom povoljnom okruženju je Edgar Frank Kod (*Edgar Frank Codd*), direktor IBM-ovog razvojnoj centra iz San Hozea, objavio 1970. godine rad u kome predlaže da se raznovrsni podaci skladište u obliku tablica. Ovaj novi model koji je nazvan *relacioni model podataka* smatrao se u tom trenutku čisto kao zanimacija naučnika jer nije bilo jasno kako bi računar mogao uspešno i efikasno da obrađuje ove tablice. Ipak, Kod je nastavio da sledi svoju ideju koja je dovela do razvoja najuspešnijeg modela baza podataka.

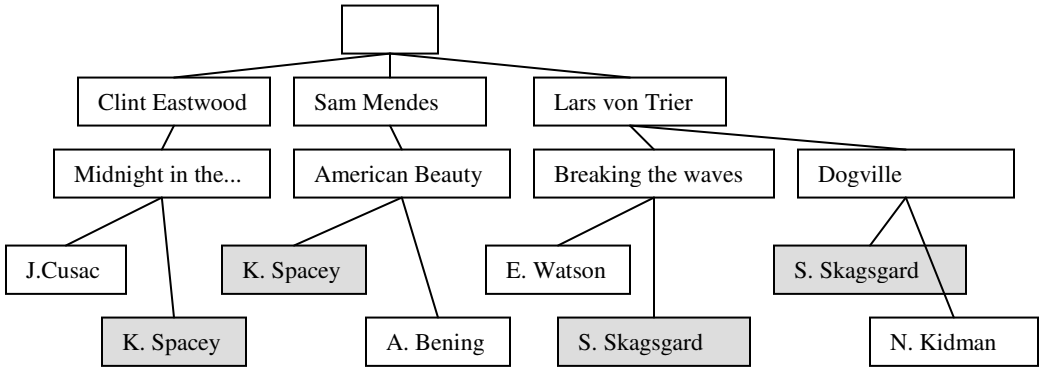
Paradigma objektno-orijentisanog programiranja je nastala sa programskim jezikom Simula (*Simple universal language*) 1967. godine, što znači da je nastala pre relacionog modela. Međutim, tek je u 80-tim godinama XX veka objektno-orijentisano programiranje dobilo pravi polet, a tek 90-tih godina je došlo i do realnih primena sa jezicima kao što su C++, Java i Python. Pojam *objektno-orijentisanih baza podataka* se takođe pojavio u tom periodu, a posebno sa manifestom *The Object-Oriented Database System Manifesto* koji je 1989. predstavljen na konferenciji DOOD (*Conference on Deductive and Object-Oriented Databases*).

Istovremeno se razvoj objektnih sistema za upravljanje bazama podataka vrlo brzo suočio s neophodnošću ostvarivanja kompatibilnosti s postojećim modelima, a pre svega s relacionim modelom. Odatle je nastao *relaciono-objektni model* s ciljem da se relacioni model proširi bazičnim objektnim konceptima. Ovo objektno proširenje relacionog modela je bilo predmet novog standarda koji je donet 1999. godine – SQL-3.

1.2.2 Hijerahijski model

Konceptualni modeli iz ovog perioda su još uvek tesno povezani sa organizacijom datoteka na računaru. Tako, slično organizaciji datoteka na disku i hijerahijski sistemi za upravljanje bazama podataka organizuje zapise u drvoliku strukturu na takav način da svaki zapis ima samo jedan nadređeni zapis. Hijerarhijska baza podataka se može, prema tome posmatrati kao jedno drvo, slično onom na Slika 1-2. Hijerarhijski model je prvo zamišljen i upotrebljen za NASA-in program Apolo. Dugo se koristio IBM-ov sistem IMS (*Information Management System*) koji se zasnivao na ovom modelu.

Hijerarhijske veze između različitih tipova podataka mogu jednostavno da daju odgovore na neka pitanja, dok se odgovori na druga pitanja dobijaju mnogo teže. Ako relacija između dva zapisa više nije 1:*N* (npr. glumci Kevin Spacey i Stellan Skarsgard su obojica glumili u dva filma) hijerarhijski model više nije adekvatan i mora se preći na mrežni model.

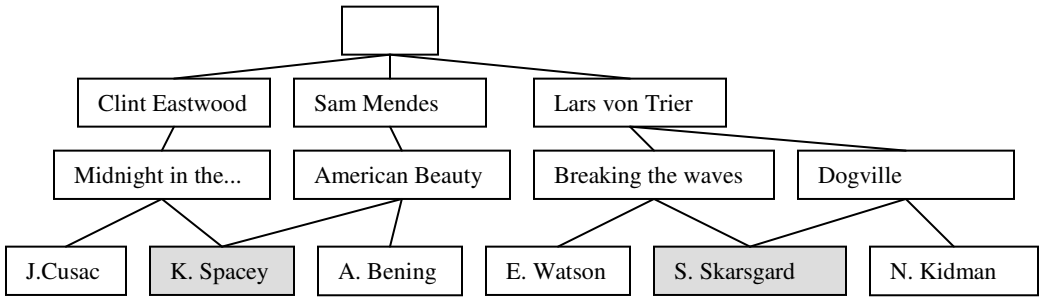


Slika 1-2 Primer hijerarhijskog modela

1.2.3 Mrežni model

Mrežni model je proširenje hijerarhijskog modela koji omogućava da se uklone mnoge poteškoće koje stvara hijerarhijski model pošto on dozvoljava da se uspostavljaju veze koje su urođene drvetima, ali i mrežama. Primer na Slika 1-3 pokazuje da sada to što su Kevin Spacey i Stellan Skarsgard glumili u dva filma ne znači da njihovi zapisi moraju da se ponove.

Kao i kod hijerarhijskog modela, i kod mrežnog modela da bi se pronašao neki podatak treba poznavati put (tj. veze) koje vode do tog podatka, što znači da su programi za obradu zavisni od podataka. Zbog ovog važnog nedostatka i hijerarhijski i mrežni model su odbačeni u praksi i koriste se samo na konceptualnom nivou.



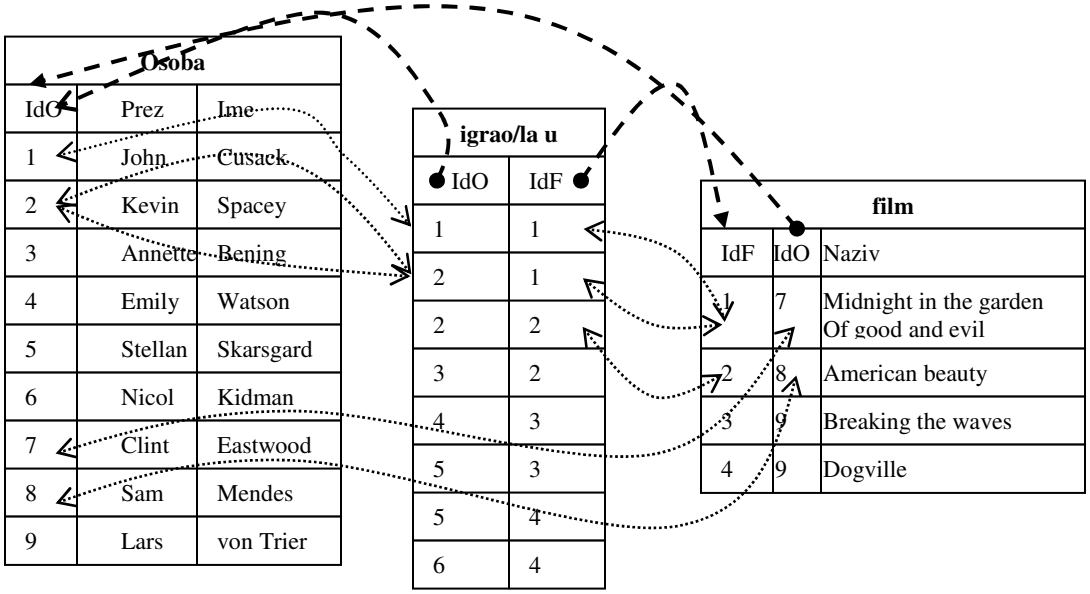
Slika 1-3 Primer mrežnog modela

1.2.4 Relacioni model

Ovaj model je nastao 60-tih godina XX veka u IBM-u. Njegov autor Edgar Kod je želeo da izradi model pomoću koga će moći da se obradi velika količina podataka, koji će obezbediti visok nivo nezavisnosti (od samih podataka) i čija će interna reprezentacija podataka obezbediti solidnu osnovu za reševanje problema koherentnosti i redundantnosti podataka. Pošto je bio matematičar po obrazovanju on je ovaj model izgradio koristeći matematičku teoriju skupova i predikatski račun (ili logiku prvog reda). On je 70-tih godina objavio rad u kome predlaže da se heterogeni podaci skladište u tabele nalik onima na Slika 1-4. Prvi prototip ovakavog sistema za upravljanje bazama podataka izgrađen je u IBM-u. Do osamdesetih godina ova metodologija je napredovala i postala je opšteprihvaćena u komercijalnom svetu.

Edgar Kod nije razvio samo model podataka već i neproceduralni jezik za manipulaciju podataka u bazi koja se zasniva na relacionom modelu. Koristeći ovaj jezik korisnik definiše kakav rezultat želi da dobije, a ne proceduru koja će do tog rezultata dovesti. Tako je nastao jezik SEQUEL (*Structured English Query Language*) koje je kasnije preimenovan u SQL (*Structured Query Language*). Ovaj jezik je standardizovan u ANSI-ju 1986, a međunarodna organizacija ISO ga je standardizovala 1987.

Najveći deo ovog kursa će se zasnivati na ovom modelu.



Slika 1-4 Relacioni model

1.2.5 Objektno orijentisani modeli

1.2.5.1 Objektni model

Baza podataka je u principu jedna komponenta neke aplikacije. Zbog toga je neophodno da baza podataka, programski jezik i metodologija za razvoj aplikacije čine jednu koherentnu celinu. Objektno orijentisane baze podataka se s toga najprirodnije integrišu sa objektno orijetnisanim pokretom koga karakterišu programski jezici C++, Java i Python. Glavna karakteristika objektno orijentisanih sistema za upravljanje bazama podataka je združivanje dve tehnologije: tehnologija baza podataka i objektno orijentisanog programiranja. U ovom modelu informacije se predstavljaju u obliku *objekata*, što znači da podatak više nije pojedinačna skalarna vrednost već strukturirana kolekcija.

Glavni problem ovog pristupa, koji je uveden 90-tih godina XX veka, je nepostojanje standarda. Takođe, jedna radna grupa koju je 1991. osnovao proizvođač *Sun Microsystems* i koja je okupljala 5 proizvođača objektno orijentisanih sistema za upravljanje bazama podataka predložila je objektno proširenje upitnog jezika SQL – OQL. Međutim da bi se takvo proširenje moglo primeniti potrebna je kompatibilnost sa postojećim, relacionim, modelom. Tako je nastao relaciono-objektni model.

1.2.5.2 Relaciono-objektni model

Cilj relaciono-objektnog modela je da proširi relacioni model izvesnim suštinskim karakteristikama objektnog modela. Tako su sve krakteristike klasičnih sistema za upravljanje bazama podataka zadržane pošto je ovaj novi sistem u potpunosti kompatibilan sa klasičnim relacionim modelom. Ovo objektno proširenje relacionog modela je standradizovano 1999. godine – to je SQL-3. Ovim standardom se upitni jezik SQL proširuje objektnim karakteristikama: pozivanje operacija, slaganje (kompozicija) operatora, podrška nasleđivanju, itd. SQL-3 su usvojile mnogi veliki proizvođači, među njima IBM, Oracle, Sybase.

1.2.5.3 Mapiranje objektnog u relaciono

Tekući trend je stvaranje kohabitacije jednog objektno orijentisanog jezika i jedne relacione baze podataka. To se ostvaruje pomoću *mapiranja objektnog u relaciono* koje uvodi, takozvanu, istrajnost u objektni svet. Pod istrajnošću (*persistent*) se podrazumeva postojanje objekta i posle završetka rada programa. Pomoću mapiranja se zapravo stvara virtualna objektna baza podataka. Da bi se ovo ostvarilo nastali su okviri (*framework*) kao kolekcije programskih rutina, alata i metodologija koji pomažu, ubrzavaju i olakšavaju razvoj i održavanje složenih aplikacija. Jedan takav okvir je JDO (*Java Data Objects*) za programski jezik Java koji je razvio Sun Microsystems i okvir otvorenog koda Hibernate koji predstavlja sumeđu ka relacionoj bazi koja zamenjuje pristup bazi podataka pozivima objektnih metoda visokog nivoa.

1.3 Sistem za upravljanje bazama podatka

Skup programa koji obezbeđuju upravljanje bazom podataka i pristup ka njoj se naziva *sistem za upravljanje bazama podatka*. Jedan sistem za

upravljanje bazama podataka održava u principu više različitih baza podataka koje su tematski različite. Sistem za upravljanje bazama podataka omogućava dodavanje podataka, njihovu modifikaciju i pretraživanje. Koji god model podataka da je primenjen – mrežni, relacioni ili objektni – glavni zadatak sistema za upravljanje bazama podataka je da sakrije od korisnika fizičku reprezentaciju podataka i da obezbedi zaštitu i koherentnost podataka u višekorisničkom okruženju u kome mnogi korisnici istovremenu pristupaju podacima.

Sistem za upravljanje bazama podataka ostvaruje više značajnih ciljeva u stvaranju i eksploataciji baze podataka:

- ***fizička nezavisnost***: interna reprezentacija podataka i metoda ze pristup sistemu datoteka mora za korisnika da bude transparentna. To znači da promene fizičke organizacije podataka ne zahtevaju modifikaciju programa obrade nad bazom podataka;
- ***logička nezavisnost***: jedan isti skup podataka različiti korisnici mogu da vide na različite načine. Sve ove različite poglede definišu *eksterne šeme* koje moraju u potpunosti da budu integrisane sa globalnom *konceptualnom šemom*. Ma kako bila duboka promena konceptualne šeme ona ne bi trebalo da ima uticaja na eksternu šemu. To znači, na primer, uvođenje nove rubrike - kolone u tabeli - ne zahteva izmene programa obrade koji nisu neposredno pogođeni ovom izmenom.
- ***jednostavno rukovanje podacima*:** omogućava korisniku koji ne mora biti informatičar da na jednostavan način njima rukuje (postavlja upite i ažurira podatke). Pristup podacima se obavlja preko jezika za manipulaciju podacima – DML (*Data Manipulation Language*). Od izuzetne je važnosti da ovaj jezik omogući da se odgovori na upite dobiju u nekom razumnom vermenu.
- ***centralizovano administriranje podataka***: sistem za upravljanje bazom podataka obezbeđuje alate za opis strukture podataka i omogućava nadgledanje te strukture, kao i opis njene evolucije, tj. njenih promena u vremenu (Ovo su zadaci ***administratora baze podataka***). Svi podaci treba da budu centralizovani u skladištu koje je zajedničko za sve aplikacije. U stvari, različiti pogledi na podatke se najlakše razrešavaju ako se podacima upravlja iz jednog mesta.

- ***efikasnost pristupa podacima***: obezbeđuje dobar *protok* (broj transakcija[1] u sekundi) i *odziv sistema* (srednje vreme čekanja za obavljanje jedne transakcije)
- ***kontrolisana redundantnost podataka***: da bi se izbegli problemi sa ažuriranjem podataka, svaka informacija u bazi podataka treba da bude predstavljena samo jednom. Time se smanjenje potreban prostor za skladištenje podataka uz istovremeno izbegavanje višestrukog ažuriranja i stvaranja nesaglasnosti među podacima (*narušavanje koherentnosti*).
- ***koherentnost podataka***: podaci moraju da zadovolje *uslove integriteta* pomoću kojih se definiše koherentno stanje baze podataka. Ovi uslovi moraju da se izraze jednostavno i da se automatski proveravaju kod svakog umetanja, promene ili brisanja podatka. Koherentnost podataka se obezbeđuje preko jezika za opis podataka DDL (*Data Description Language*).
- ***deljenje podataka***: sistem za upravljanje bazama podataka treba da omogući istovremeno korišćenje podataka u različitim aplikacijama i da otkriva i obrađuje slučajeve u kojima postoji konflikt u pristupu bazi između više korisnika. Ovaj problem se lako rešava kad više korisnika istovremeno konsultuje podatke, a mnogo je ozbiljniji kada više korisnika istovremeno podatke modifikuje.
- ***bezbednost podataka***: podaci u bazi podataka moraju biti zaštićeni od neovlašćenog pristupa. Da bi se to ostvarilo svakom korisniku treba da se dodeli određeni nivo prava za pristup podacima.
- ***otpornost na kvarove***: Šta se dešava ako dođe do kvara (ili nestanka struje) usred neke transakcije promene podataka, ili ako neke datoteke koje sadrže podatke postanu nečitljive zbog kvara na disku? Treba da postoji mogućnost da se baza vrati u stabilno stanje u kome je bila pre početka transakcije promene podataka.

1.3.1 SUBP zasnovana na datotekama prema klijent/server modelu

Treba razlikovati dva tipa funkcionisanja sistema za upravljanje bazama podataka: SUBP koji se zasnivaju na datotekama i one koji se zasnivaju na modelu klijent/server. Klijent/server arhitektura smanjuje saobraćaj na mreži i, sve u svemu, bolje je prilagođena za rad sa velikom

[1] Transakcija je vremenski uređeni niz operacija nad bazom podataka: BP prelazi iz jednog stanja u sledeće izvršenjem transakcije.

količinom podataka, velikim brojem korisnika, jakim i mnogobrojnim uslovima integriteta i otpornija je na kvarove.

1.3.1.1 SUBP zasnovan na datotekama

Kod ovog načina rada bazu podatak čini skup datoteka (recimo jedna datoteka za jednu tabelu) koje su pohranjene na nekom deljenom memorijskom prostoru, na primer na mrežnom serveru. Aplikacija koja je instalirana na svakoj radnoj stanici je zadužena za obradu svih SQL komandi. Upiti se dakle obrađuju lokalno i da bi to bilo moguće sve tabele koje su za taj upit potrebne se moraju preneti na tu radnu stanicu. Ovaj način rada stoga dovodi do veoma intenzivnog saobraćaja na mreži. Prednost ovog načina rada je jednostavnost, prividno nepostojanje administriranja i mogućnost korišćenja otvorenih formata dokumenata. Najvažniji sistemi za upravljanje bazama podataka koji ovako rade su FoxPro, Paradox, Access i MySQL.

1.3.1.2 Klijent/server arhitektura SUBP

Kod ovog načina rada SUBP radi na jednom računaru koji se naziva server i na kome su uskladišteni i podaci. Izvršavanje upita se obavlja na serveru, a mrežom između servera i klijenta putuju samo upit i rezultati. Na taj način se ostvaruje nezavisnost između podataka i klijenata koji postavljaju upite. Na primer, prava korisnika mogu da budu nezavisna od sistema za eksploataciju.

Glavna prednost ovog pristupa je mogućnost rada sa velikim kolekcijama podataka upravo zato što ti podaci ne putuju mrežom za svaki upit. Takođe, upravljanje istovremenim radom velikog broja korisnika kao i upravljanje transakcijama se lakše ostvaruje. Nedostatak je što je potrebno dosta rada da se sve ovo uspostavi, a takođe je potrebna jaka mašina koja bi služila kao server baze podataka i koja bi bila posvećena samo tom zadatku. Najvažniji sistemi za upravljanje bazama podataka koji koriste klijent-server arhitekturu su Oracle, DB2, SQL-Server, Sybase i PostgreSQL.

1.3.2 Glavni sistemi za upravljanje bazama podataka an tržištu

Podaci iz 2006. godine govore da je komercijalni sistem za upravljanje bazama podataka na tržištu Oracle firme Oracle Corporation (47,1%), sledi DB2 firme IBM (21,1%), SQL-Server firme Microsoft (17,4%), a za njima su sa po 3.4% udela u tržištu Sybase i Teradata.

Nasuprot ovim veoma skupim proizvodima su besplatni ili slobodni proizvodi. Najpoznatiji proizvodi ove vrste su: MySQL koji se posebno koristi za kreiranje dinamičkih internet lokacija (npr, kombinacija PHP/MySQL), Firebird kao najuspešniji sistem za obradu transakcija, PostgreSQL koji omogućava rad sa relaciono-objektnim bazama, Ingres koji dobro zadovoljava potrebe preduzeća i Clodspace koji može dobro da kooperira sa programskim jezikom Java.

Iako ovi proizvodi ne mogu po performansama da se mere sa skupim vlasničkim softverom, neki od njih im se približavaju. Najveće razlike su trenutno u tome što vlasnički sistemi nude izvanredna okruženja za izradu aplikacija kao i korisničke sumeđe koja slobodan softver u principu nema ili su rudimentarna.

1.3.3 Proces projektovanja baze podataka

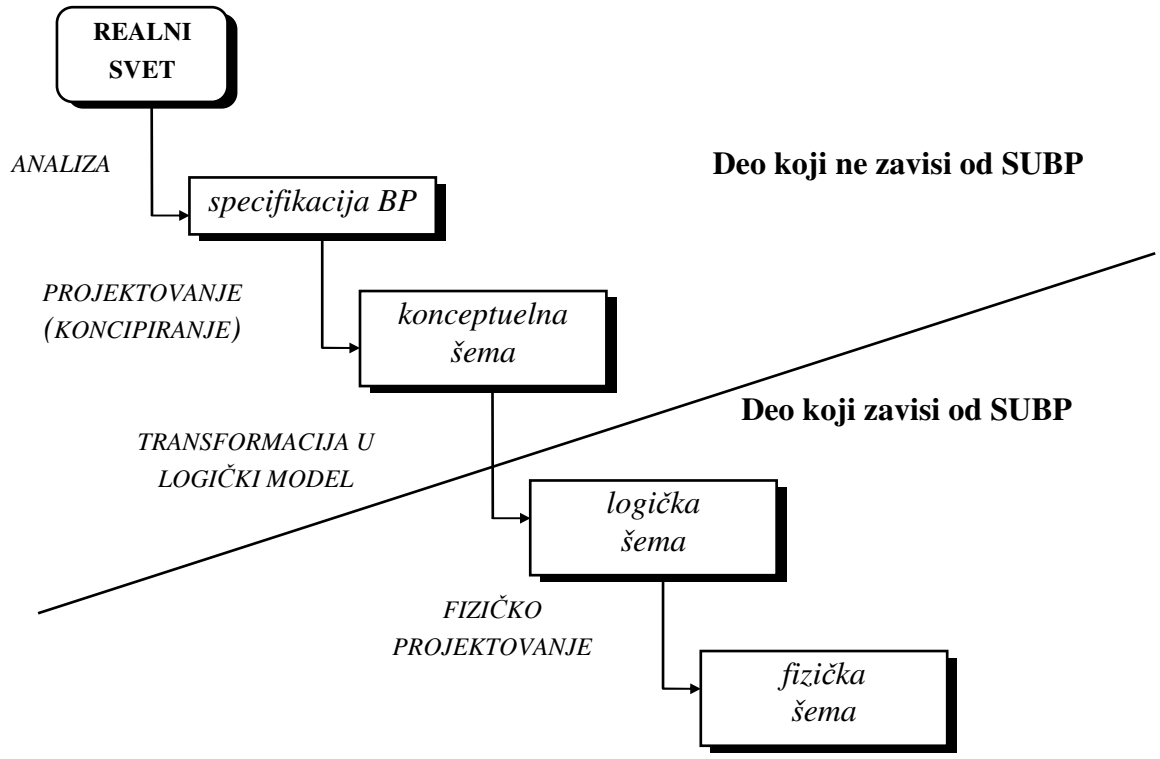
Izgradnja baze podataka prolazi kroz više etapa kojima se postupno segment realnog sveta, koji treba da bude predstavljen bazom podataka, prevodi u odgovarajuće informatičke apstrakcije. U izvesnom smislu, „*sistem baze podataka možemo posmatrati kao hijerarhiju apstrakcija*". Naime, baza podataka predstavlja reprezentaciju informacija iz nekog segmenta realnog sveta. Informatičko predstavljanje elemenata realnog sveta, tada, nužno sadrži komponentu apstrahovanja, „lišavanja svojstava", objekata realnog sveta.

U primeru autobuskog preduzeća, putnik je sveden na ulogu korisnika u međugradskom prevozu („objekat koji se premešta sa jednog mesta na drugo") i sva njegova druga svojstva nisu od značaja. Putnik predstavljen u bazi podataka je svedena predstava svojstava putnika koja ne govori ništa o putniku-čoveku, koji sa vlastitim univerzumom - brigama ili radostima, sedi na autobuskoj stanici i čeka dolazak autobusa.

Ovaj proces apstrahovanja se izražava kroz proces projektovanja baze podataka u kome se, u svakom koraku, vrši supstitucija svojstava realnog objekta njegovim sve podrobnijim informatičkim aproksimacijama. Proces projektovanja prolazi kroz više faza. U prvoj fazi, koju ćemo nazvati

analiza, na osnovu istraživanja svojstava relevantnih objekata realnog sveta gradi se ***specifikacija*** (opis) baze podataka. Specifikacija BP izražava korisničke poglede na podatke, kao i zahteve u pogledu njihove eksploatacije. Na osnovu specifikacije se u fazi ***projektovanja*** (ili *koncipiranja*) gradi ***konceptuelna šema*** baze podataka. U ovom koraku se zahtevi iskazani specifikacijom prevode na opšti opis baze podataka kroz opis opštih struktura podataka i procedura. Bitno je istaći da specifikacija i projektovanje **ne zavise** ni od konkretnog sistema za upravljanje bazama podataka, niti od konkretnog računarskog sistema.

Konceptualna šema se, dalje, transformiše u *logički model* koji nazivamo ***logička šema***, a u kojoj se na jeziku za definiciju i manipulaciju podataka izražavaju veze podataka opisane konceptuelnom šemom. Logička šema se u fazi ***fizičkog projektovanja*** transformiše u ***internu*** (ili *fizičku*) ***šemu***. Na ovom nivou se detaljno i potpuno predstavljaju rezultati prethodnog projektovanja konkretnim strukturama podataka i odgovarajućim algoritmima. Ovde ćemo se upoznati sa jednim modelom projektovanja (tzv. model entitet-veze) i njegovom transformacijom u logički model (tzv. relaciona algebra). Šematski prikaz procesa projektovanja baze podataka prikazan je na Slika 1-5.



Slika 1-5 Proces projektovanja baza podataka

Napomena:
Ovaj tekst se zasniva na knjizi:
Laurent Audibert, *Base de données de la modélisation au SQL*, Ellipses Édition, 2009, ISBN 978-2-7298-5120-0
Introduction aux bases de données, 9-24.