GEOPROSTORNE BAZE PODATAKA

skripta v1 (05.2018.)

Autor: Adis Hamzić MA geod.

Sadržaj:

PREDGOVOR

Ova skripta je namjenjena studentima koji pohađaju predmet „Geoprostorne baze podataka“. Trenutna verzija skripte većim dijelom pokriva plan i program predmeta, međutim pojedine oblasti još nisu obrađene. Za svaku od obrađenih oblasti data je teoretska osnova i praktični primjeri koji se bave odgovrajućom materijom. Iako je skripta nastala iz različitih izvora kao osnova za izradu skripte su korištene sljedeće tri knjige:

1. „Spatial databases – A tour“ (2003) čiji su autori: Shekhar S. i Chawla S.,

2. „Geoprostorne baze podataka“ (2006) autora Galić Z., i

3. „Spatial databases with application to GIS“ (2002) čiji su autori: Rigaux, P., Scholl, M., i Voisard, A.

Obzirom da je za studente sa odsjeka za geodeziju ovaj predmet prvo sretanje sa prostornim bazama podataka, ne ulazi se duboko u materiju prostornih baza podataka već se skripta bavi samo osnovnim pojmovima i zadacima baza podataka.

Prvo poglavlje se bavi osnovnim pojmovima baza podataka i sistema za upravljanje bazama podataka. Drugo poglavlje obrađuje topološke koncepte i u okviru ovoga poglavlja je opisan model 9 presjeka i prošireni model 9 presjeka. Treće poglavlje se bavi standardnim jezikom za upite (SQL) i u okviru ovoga poglavlja su obrađene sljedeće oblasti: relaciona algebra, relacioni SQL i prostorni SQL. U četvrtom poglavlju se obrađuju grafovi uz dodatak primjene SQL- a nad grafovima. Peto poglavlje nudi osnovne pojmove iz oblasti optimizacije SQL upita. Šesto poglavlje je napravljeno u obliku tutoriala za izradu jednostavne prostorne baze podataka korištenjem *SpatiaLite*biblioteke.

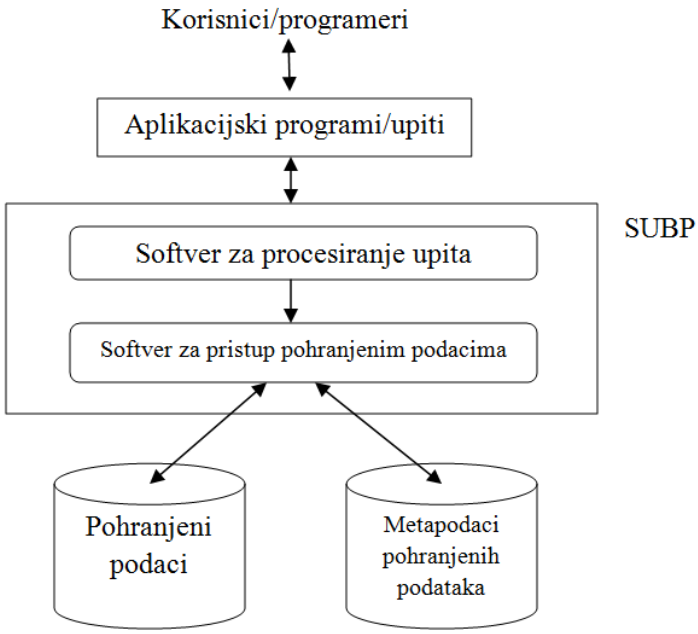
Ako imate sugestije, primjedbe ili ako ste pronašli neke greške u materijalima molim Vas da kontaktirate autora na sljedeći e-mail: hamzicadis87@gmail.com.

Adis Hamzić MA geod.

1. BAZA PODATAKA I SISTEM ZA UPRAVLJANJE BAZOM PODATAKA

Baza podataka je velika kolekcija međusobno povezanih podataka koji su pohranjeni u računarskom okruženju. U takvom okruženju, podaci su *perzistentni*[1], što znači da mogu preživjeti neočekivane softverske i hardverske probleme (izuzev teških slučajeva uništenja diska). I veliki opseg podataka i perzistentnost, dvije glavne osobine baze podataka, su u suprotnosti sa manipulacijom informacijama pomoću programskih jezika, koji imaju opseg podataka dovoljno mali da bi mogao opstati u glavnoj memoriji i koji nestaju nakon završenja rada programa. (Rigaux, Scholl, & Voisard, 2002)

Baze podataka mogu pohranjivati informacije o bilo čemu (ljudima, proizvodima, objektima i sl.). Kako količina podataka raste sve je teže manipulisati njima (pretraživanje, ažuriranje, itd.). Da bi se izbjegli ovakvi problemi potrebno je premjestiti podatke u bazu podataka napravljenu korištenjem sistema za upravljanje bazama podataka (SUBP). Šema modernog SUBP-a prikazana je na slici 1.



Slika 1: Pojednostavljeno okruženje sistema za upravljanje bazom podataka (Rigaux, Scholl, & Voisard, 2002)

SUBP je kolekcija softvera koji upravljaju strukturom baze podataka i kontrolira pristup podacima pohranjenim u bazi. Uopšteno, SUBP olakšava procese. (Rigaux, Scholl, & Voisard, 2002):

[1] postoje ili ostaju u istom stanju na neodređeno vrijeme

- *Definisanja* baze podataka; tu spada specificiranje tipova podataka, strukture, i ograničenja koja se moraju uzeti u obzir;
- *Izgradnje* baze podataka - tu podrazumijevamo pohranjivanje samih podataka u perzistentno skladište;
- *Manipulacije* bazom podataka;
- *Postavljanje upita* nad bazom radi vađenja specificiranih podataka.
- *Ažuriranje* baze podatka.

Konvencionalni sistemi za upravljanje bazama podataka razvijeni su i uspješno se primjenjuju uglavnom za upravljanje velikim skupovima podataka u poslovnim i administrativnim, tzv. *standardnim* aplikacijama. U posljednje vrijeme, mnoga istraživanja u domeni baza podataka usmjerena su na podršku baza podataka za tehničko-naučne, tj. *nestandardne* aplikacije. Uočeno je da konvencionalni modeli podataka i sistemi baza podataka ne zadovoljavaju zahtjeve u nestandardnim aplikacijskim domenama kao što su računarski podržano projektovanje (CAD - eng. *Computer Aided Design*), geoinformacijski sistemi (GIS), multimedijalni sistemi, itd. Ovi novi zahtjevi su stimulativno uticali na istraživanja u proširenju tehnologije baza podataka prema potrebama u tim aplikacijskim domenama i doveli do projektovanja *nestandardnih sistema baza podataka*. Kod ovih sistema neophodna je podrška baza podataka za upravljanje geometrijskim i prostornim, tj. *geoprostornim* podacima. Za ovakve nestandardne sisteme baza podataka, koji podržavaju ovakve vrste podataka, koriste se različiti pojmovi: slikovni, rasterski, prostorni, geoegrafski, geoprostorni ili geometrijski sistemi baza podataka. Geoprostorni sistem baza podataka je sistem baze podataka sa svim osobinama standardnog sistema, s dodatnim mogućnostima za reprezentaciju, manipulaciju i analizu (upite) objekata u prostoru. To znači da korisnik može promatrati geoprostorne i standardne (alfanumeričke) podatke na homogen način. (Galić, 2006)

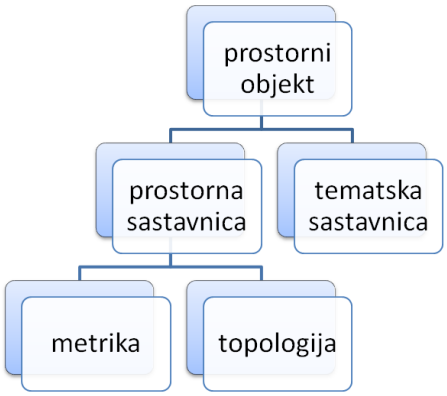
Moderni sistemi za upravljanje podacima katastra nekretnina zasnovani su na sistemima za upravljanje bazama podataka. Sa aspekta modeliranja, baza podataka se može posmatrati kao hijerarhija apstrakcija. Svaki nivo u toj hijerarhiji je jedna vrsta modela, što znači da prikazuje skup objekata i operacija nad tim objektima. Zadaća sistema za upravljanje bazama podataka je da omogući računalnu realizaciju svakog od tih modela, osiguravajući pri tome transformisanje modela na višem nivou apstrakcije u modele na nižem nivou. Na jednom nivou te hijerarhije apstrakcija nalazi se *model podataka*, odnosno logička, korisnicima prilagođena slika podataka u bazi, uz skup operacija koje korisnici izvršavaju nad

tim podacima. Glavna osobina tog nivoa je potpuna odsutnost bilo kakvih uticaja računalnog sistema na model podataka. Na nivou neposredno ispod modela podataka nalazi se reprezentacija modela podataka pomoću pogodnih *struktura podataka* i operacija modela, kao algoritama koji operiraju nad tim strukturama podataka. I na ovom nivou su odsutni uticaji računalnog sistema, ali ne u potpunosti, nego su prisutni u apstrahiranom obliku. Stoga se ta razina naziva razinom *fizičke reprezentacije*. Da je nivo modela podataka viši u odnosu na nivo fizičke reprezentacije, pokazuje i činjenica da korisnici postavljaju *upite* na nivou modela, prema unaprijed definisanim operacijama modela, a ti se upiti realizuju na fizičkom nivou, kao kompleksne procedure pretraživanja strukture baze. (Galić & Govedarica, 2007)

1.2 Prostorni podaci

Svi prostorni podaci sastavljeni su od tri komponente (slika 2):

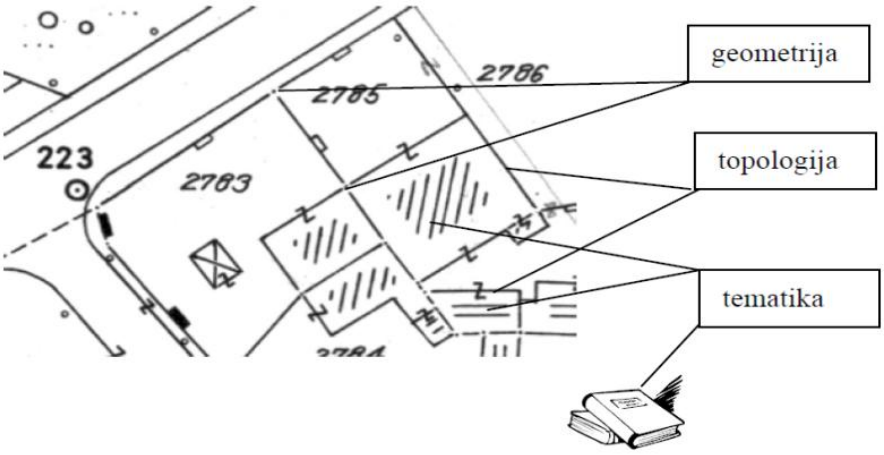
1. metričke,
2. topološke, i
3. tematske.



Slika 2: Struktura prostornog objekta

Prve dvije čine prostornu sastavnicu prostornih podataka, a posljednja opisnu sastavnicu podataka. **Metrika** opisuje oblik (geometriju, strukturu), položaj i veličinu prostornog objekta u referentnom sistemu (obično Kartezijev koordinatni sistem). Oblik geoprostornog objekta opisuje apstrakciju njegove geometrisjke strukture (tačka, linija, poligon ili pravougaonik). Tako se npr. grad može modelirati kao poligon, a oblik rijeke kao linija. Položaj objekta označava položaj objekta u odnosu na referentni koordinatni sistem. Prema prostornom prostiranju razlikujemo 0-dimenzionalne (0D), 1-dimenzionalne (1D), 2-dimenzionalne (2D) i 3-dimenzionalne (3D) objekte. Uvođenje metrike omogućava izvođenje daljnjih metričkih informacija i računanje veličina, kao što su udaljenost između dva objekta, obim i površina poligona, ili dužina linije. **Topologija** opisuje relacije između geoprostornih objekata koje se

odnose na susjedstvo, povezanost, uključenost i slične relacije među objektima. Bitna osobina tih relacija je njihova nezavisnost o referentnom sistemu i nepromjenljivost pri topološkim transformacijama, kao što su translacija i rotacija. **Tematske** osobine opisuju atributno utemeljene podatke koji se obično izražavaju standardnim, alfanumeričkim podacima (npr. naziv grada, oznaka auto-ceste, naziv ulice, itd.). (Galić, 2006)

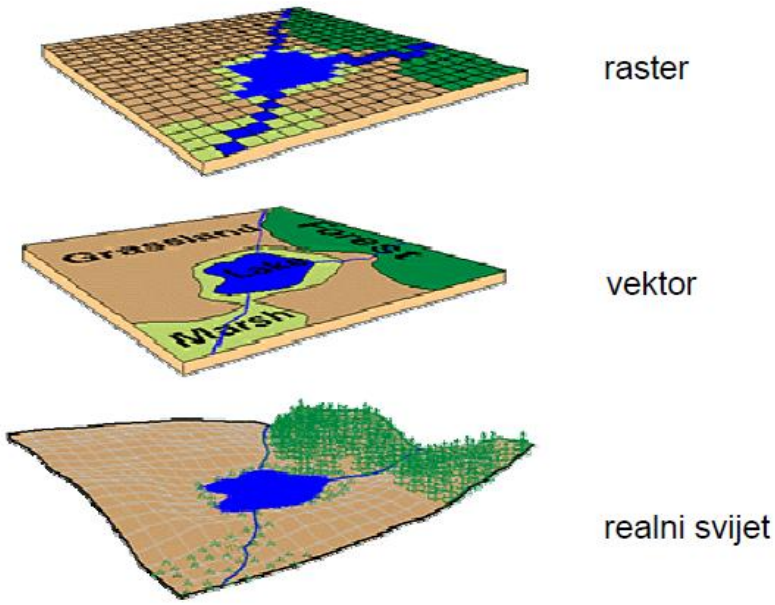


Slika 3: Vrste podataka na katastarskom planu (Roić, Matijević, & Cetl, 2002)

Katastar je tipičan primjer aplikacije u kojoj je praćenje historije promjena od posebne važnosti pa je tako osim metričke, topološke i tematske komponente prostornih podataka u katastru jako bitna i vremenska komponenta. U cilju podrške razvoju aplikacija, tj. funkcionalnosti pretraživanja i pristupu katastarskim podacima kroz njihovu historiju, model podataka omogućava razvoj tzv. ***bitemporalnih baza podataka***, u kojima su transakcijsko vrijemei vrijeme validnostiobjekta ortogonalni, te su oba specificirana kao atributi katastarskog objekta baze podataka.

1.2.1 Vrste prostornih podataka

Dva su pogleda na prostorne podatke zavisno od toga da li su primarne njihove osobine ili njihov položaj. Obzirom na unutrašnju predstavu i strukturu podataka potrebnu za implementaciju ta dva pogleda, može se reći da postoje dvije vrste modela prostornih podataka. To su rasterski i vektorski model (slika 4). Nekada se koriste sinonimi površinski (arealni) za rasterske podatke i linijski (linearni) za vektorske podatke. Razlog tome je bolja prilagođenost pojedinog modela za prikaz površinskih odnosno linijskih obilježja prostornih objekata.

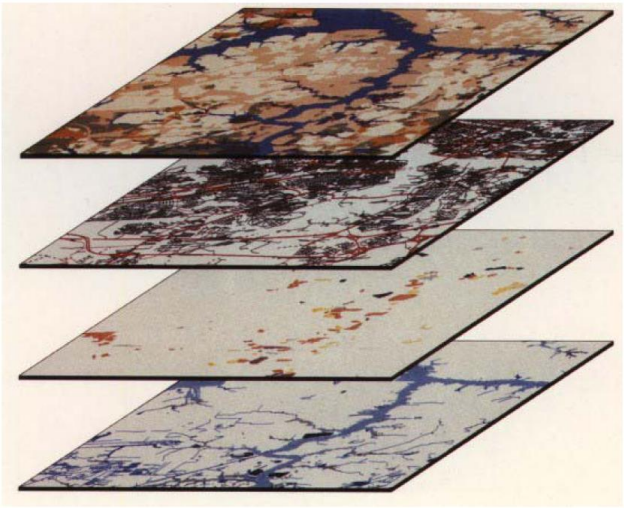


Slika 4: Rasterski i vektorski prikaz realnog svijeta (URL)

Rasterski podaci

Rasterski podaci temelje se na površinama, a osnovni geometrijski element je piksel (eng. *pixel*, skraćeno od *picture element*). Položaj piksela određen je redom i kolonom u slikovnoj matrici. Rasterska slika je slika prikazana pomoću vrijednosti amplituda svjetline (ili boje) tačaka. (Roić & Cetl, 2003)

Rasterski sistemi dijele prostor na pravougaone ćelije istog oblika i veličine. Ovaj način predstave prostora se obično koristi kod slojevitog načina modeliranja kao što se može vidjeti na slici 5. Svakoj ćeliji se pridružuje određena atributna vrijednost. Ćelije iz različitih slojeva se preklapaju (ako je postupak geokodiranja urađen pravilno) tako da je moguće vršiti različite analize kombinacijom vrijednosti pomoću različitih matematičkih ili nekih drugih operacija. Rasterski podaci jako dobro opisuju objekte ili pojave koje nemaju jasno definirane granice (rijeke, planine, i sl.). Dakle, kod rasterskih podataka u prvom planu je obilježje, a u drugom položajna tačnost.



Slika 5: Slojeviti način modeliranja kod rasterskih podataka (Matijević, 2004)

Vektorski podaci

Kod vektorske predstave prostora u prvom planu je položaj, a u drugom obilježje, osim toga prostor je podijeljen na nepravilne dijelove. Sistemi koji predstavljaju prostor na ovaj način zovu se vektorski (slika 6). Važna osobina vektorskog modela je uključivanje topoloških odnosa. Na ovaj način je ostvaren puni topološki model (kod rasterskih modela topologija postoji ali u implicitnom obliku). Tako se postiže „svijest" pojedine ćelije o njezinim susjedima, te nadalje otvara mogućnost ispitivanja korektnosti podataka kontrolom topoloških uslova i zakonitosti. Razlika između rasterskog i vektorskog modela je i u prostornoj razlučivosti[2], koja je kod rasterskih podataka manja nego kod vektorskih. Povećanjem razlučivosti povećava se i količina podataka pa je nekada potrebno povećati stepen uopštenja prostornog objekta.

[2]Razlučivost ili rezolucija je veličina kojom se definira mogućnost razdvajanja/razaznavanja sitnih detalja i koristi se za opisivanje kvaliteta slike

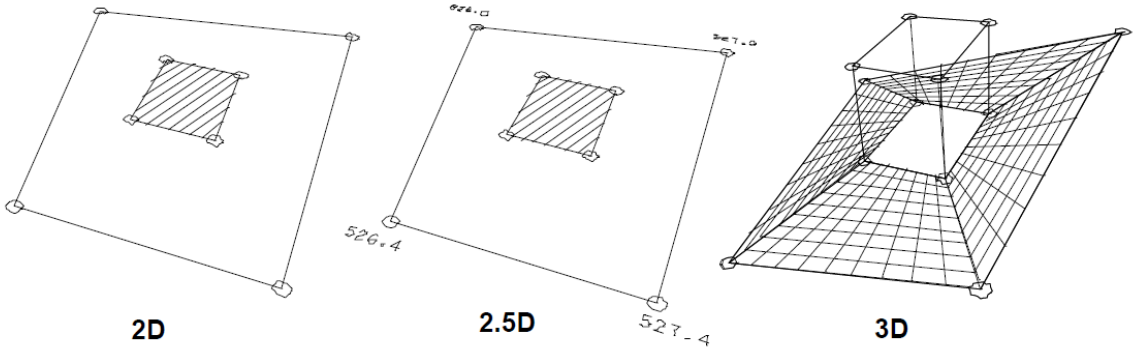
Slika 6: Vektorski način predstave realnog svijeta(Sutton & Dassau, 2009)

Kao što je već rečeno, prostorni podaci se sastoje od prostorne (metrika i topologija) i opisne (tematika) sastavnice. Osnova prostorne sastavnice je geometrija koja sadrži metričke podatke najčešće date koordinatama u nekom referentnom sistemu. Međutim, samo poznavanje položaja karakterističnih tačaka objekta ne određuje njegov izgled u potpunosti nego je potrebno uvesti odnose između pojedinih objekata i njihovih osobina. Odnosi između prostornih objekata su definisani topologijom.

Dakle, geometrija i topologija jednoznačno određuju oblik, veličinu i položaj modela objekta u prostoru, odnosno oni čine njegovu prostornu sastavnicu. Spajanjem prostorne s opisnom odnosno atributnom sastavnicom dobivamo potpuno određen objekat iz stvarnog svijeta. U prethodnoj rečenici treba voditi računa na to da je objekat potpuno određen razmjerno na model podataka, odnosno stepen uopštenja koji je u konkretnom slučaju prihvaćen. Opisane sastavnice su jednako važne i izdvajanjem bilo koje iz cjeline prostornog objekta narušava se njegova cjelovitost, a nepotpun podatak može navesti na pogrešne zaključke o njegovoj prirodi. Osim toga svaka sastavnica prostornog podatka ima različite posebnosti i ponaša se prema drugim pravilima. (Matijević, 2004)

Geometrija prostornih objekata određena je oblikom i relativnim položajem njegovih karakterističnih tačaka. Najčešći oblik određivanja položaja tih tačaka su koordinate u izabranom referentnom sistemu (obično pravougli Kartezijev koordinatni sistem ali može biti i bilo koji drugi koordinatni sistem). Kao što prosotrni objekti imaju svoju dimenzionalnost tako i prostorni podaci imaju svoju dimenzionalnost. Ako imamo geometrijske podatke u ravnini govorimo o 2D podacima. Ako ti podaci imaju i visinu kao atribut tada se radi o 2.5D podacima, a ako 3D podacima (geometrijski podaci u trodimenzionalnom prostoru) dodamo i

vremensku komponentu dobit ćemo 4D podatke. Dimenzionalnost podataka prikazana je na slici 7.



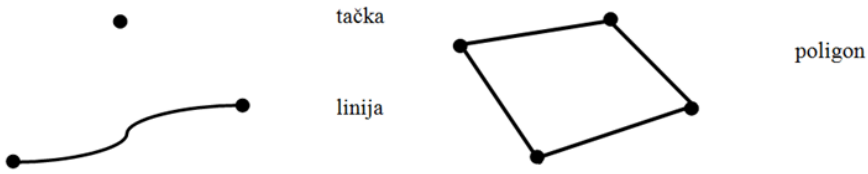
Slika 7: Vrste geometrijskih podataka prema dimenzionalnosti (Matijević, 2004)

U vektorskom modelu, objekti se prave od tačaka i bridova kao primitiva. Tačka je predstavljena parom koordinata, dok su složeniji linearni i površinski objekti predstavljeni pomoću struktura (listama, skupovima, nizovima) tačaka. Za razliku od rastereske predstave, vektorska predstava ne zauzima mnogo memorije. Npr., poligon je predstavljen konačnim brojem njegovih čvorova. (Rigaux, Scholl, & Voisard, 2002)

Kao što je već rečeno, tačka je nosilac geometrijske informacije, a sve ostale više strukture (linije i poligoni) se grade pomoću tačaka. Iz koordinata tačke je moguće dalje izvoditi metričke podatke o tim strukturama kao što su dužina linije ili površina poligona.

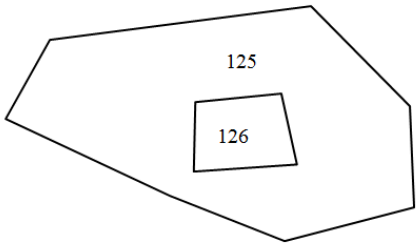
Sljedeći stepen u strukturi vektorskog modela nastaje povezivanjem tačaka u linijske strukture koje se zovu rubovi ili bridovi (eng. *edges*). Povezivanjem čvorova u bridove među njima se uspostavlja jednoznačan topološki odnos koji u geometrijskom smislu može imati različite oblike. Bez obzira na to da li je veza između čvorova prava ili zakrivljena u topološkom smislu čvorovi su na isti način povezani. I dalje ostaje uslov jednog čvora na jednom mjestu u prostoru, a analogija se proširuje i na bridove. Osim ovoga uslova kod bridova nije dozvoljeno križanje. Ukoliko dođe do križanja bridova na tome mjestu je obavezan čvor. Poštivanjem ovih pravila olakšano je obavljanje operacija nad podacima.

Daljim povezivanjem linija dobivaju se poligoni (površi) koji su određeni svojim bridovima. Dakle povezivanjem tačaka dobivamo linije, a povezivanjem linija dobivamo poligone (slika 8).



Slika 8: Tačka, linija i poligon

Stvari su donekle komplikovanije ako se pojave poligoni koji nisu u potpunosti ispunjeni već sadrže tzv. otoke (eng. *islands*). Takvi slučajevi su česti, a jedan od primjera je veća katastarska čestica u okviru koje je u cjelosti sadržana neka manja. Prirodno je da kod takvih slučajeva želimo odvojeno promatrati i analizirati svaku od površina, te biti u stanju izdvojiti manju iz one veće. Površina je sada sastavljena od vanjskog i mogućih unutrašnjih rubnih poligona koje ovdje nazivamo prstenovi. Kako za vanjski, tako i za eventualne unutrašnje prstenove važi da moraju biti zatvoreni, odnosno moraju biti topološki ekvivalnetni krugu. Kako bi se razlikovala od u uobičajenom govoru korištenog pojma za geometrijsku strukturu ona se u topološkom smislu naziva petlja ili strana (eng. *face*). (Matijević, 2004)



Slika 9: Petlja - poligon sa otokom

Ova struktura je ograničena sljedećim uslovima:

- Svaki brid je ograničen sa tačno dva čvora;
- U svakom čvoru može započinjati ili završavati više bridova;
- Svaki brid ima dvije petlje (lijevu i desnu) u odnosu na smijer početak-kraj;
- Svaka petlja je ograničena jednim vanjskim i jednim ili više unutrašnjih prstenova.

Da sumiramo, ova prezentacija dozovljava predstavu različitih polilinijskih i površinskih objekata. U poređenju sa rasterskim modelom podataka, veoma je koncizna prezentacija, jer je potrebna memorija za prikaz proporcionalna broju čvorova linearne aproksimacije objekata. Što je preciznija aproksimacija, veći je broj tačaka, i naravno više je prostora potrebno za pohranu objekta. (Rigaux, Scholl, & Voisard, 2002)

Rasterski i vektorski podaci – prednosti i nedostaci

Rasterski ili vektorski tip podataka – koji odabrati? Ovo pitanje često postavljaju oni kojima su potrebni podaci za geoprostornu analizu. Najbolji odgovor bi bio i jedan i drugi. Dakle, ako ste u mogućnosti da za područje na kome želite vršiti prostornu analizu nabavite i rasterske i vektorske podatke, to bi bilo optimalno rješenje. Na ovaj način biste mogli iskoristiti prednosti i jednog i drugog tipa podataka i istovremeno otkloniti nedostatke i ograničenja. Idealno bi bilo da postoji raster kao podloga i da se zatim na njega vektorski iscrtaju objekti koji su od značaja za odabranu analizu. Raster, osim što informacijama obogaćuje kartu obično je i vizuelno prihvatljiviji. S druge strane, vektorski podaci daju preciznost za one objekte koji su od posebnog značaja. Međutim, često postoje ograničenja (najčešće finansijska) zbog kojih nije moguće obezbijediti i jedne i druge podatke. U tom slučaju potrebno je odabrati one koji najbolje odgovaraju konkretnom zadatku. U tabelama su dati prednosti i nedostaci za rasterske i vektorske podatke.

Tabela 1: Vektorski podaci – prednosti i nedostaci (Eriksson, 2013)

Vektorski podaci	
Prednosti	**Nedostaci**
Mala potrošnja memorije	Loši za prikaz pojava koje se kontinualno mijenjaju
Lako ažuriranje	Čvrste granice
Jednostavni za kombinovanje sa atributnim podacima	Potrebno mnogo fizičkog rada ako je potrebno prikupiti kvalitetne podatke
Visoka preciznost	
Upravljanje topološkim relacijama	

Tabela 2: Rasterski podaci – prednosti i nedostaci (Eriksson, 2013)

Rasterski podaci	
Prednosti	**Nedostaci**
Jednostavna struktura podataka	Zauzima mnogo memorije
Jednostavno generisanje (npr. sa satelitskog ili aerofotogrametrijskog snimka)	Raspored ćelija ne prati prirodne granice
Jednostavna analiza	Niska preciznost (u odnosu na vektorske podatke)

1.3 Apstraktni geoprostorni tipovi podataka

Apstraktni tipovi podataka (ATP) su uvedeni kao način prevlađivanja male snage modeliranja – što se tiče korištenja nestandardnih baza podataka – koja je naslijeđena u osnovnom relacionom modelu podataka. ATP je apstraktni funkcionalni pogled na objekte: skup operacija je definiran na objektu date vrste. Ideja iza ovoga koncepta je da se sakrije struktura tipa podataka korisniku (programeru), koji mu može prići samo kroz operacije koje su nad njim definirane. Gledano iz vanjskog svijeta, interfejs/sučelje[3] ADT-a je lista operacija. (Rigaux, Scholl, & Voisard, 2002)

Dva koncepta su jako važna kod apstraktnih tipova podataka, to su **skrivanje informacija** (eng. *information hiding*) i **učahurenje** (eng. *encapsulation*). Skrivanje informacija omogućava pristup korisnika deklaraciji tipa podataka, ali ne i njegovoj implementaciji što onemogućava korištenje neprimjerenih funkcija. Učahurenje znači da se implementacija može zamijeniti nekom drugom, ekvivalentnom implementacijom, bez vidljivih uticaja na programe ili upitni jezik. (Galić, 2006)

1.3.1 Definisanje prostornih operatora ATP

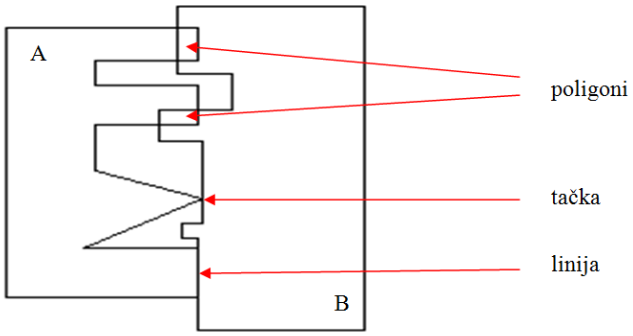
Odabir skupa operacija za pridruživanje apstraktnim tipovima podataka nije jednostavan jer rezultat operacije mora biti ili neki atomični tip (npr. *real* ili *integer*) kojeg obezbjeđuje baza podataka ili neki od apstraktnih tipova podataka. Slika 10 prikazuje presjek poligona A i B. Svaki poligon predstavlja beskonačan skup tačaka koje leže u njemu i na njegovim

[3] U kompjuterskoj nauci interfejs/sučelje predstavlja tačku komunikacije između kompjutera i bilo koga drugoga entiteta (raspored kontrola na grafici aplikacije koje reaguju na aktivnosti korisnika)

granicama. Ako prihvatimo da je presjek dva poligona „skup svih zajedničkih tačaka poligona“ rezultat može biti:

1. Dva poligona (u ovome slučaju rezultat može biti predstavljen objektom tipa region);
2. Dva poligona i linija;
3. Dva poligona, linija i tačka.

U navedenim slučajevima nemamo tipove podataka za predstavljanje ovih kombinacija.



Slika 10: Rezultat presjeka poligona A i B

Radi jednostavnosti, usvajamo prvu semantiku. Dakle, rezultat je skup poligona, a linije i tačke ćemo ignorirati. Zapazite da ovo baš i ne odgovara semantici presjeka skupova tačaka jer definisali smo presjek kao skup svih zajedničkih tačaka, a ipak u rezultatu izostavljamo zajedničku tačku. Međutim, rezultat mora biti jedan od odabranih tipova podataka. Ostale operacije su također odabrane u skladu sa sistemom tipa.

Svaka operacija koja slijedi ima svoje ime i signaturu. Signatura operacije iskazuje tip svakog od njenih argumenata i njenog rezultata. Kratak opis operacije je dat nakon svake signature operacije. Podrazumijeva se postojanje četvrtog apstraktnog tipa podataka, pravougaonik, koji nije opisan ovdje.

ATP region uključuje sljedeće operacije:

- **TačkaURegionu** (*PointInRegion*): *region×tačka→bool* (Testira da li tačka pripada regionu.)
- **Preklapa** (*Overlaps*): *region×region→bool* (Testira da li se dva regiona presjecaju.)
- **PreklapaPavougaonik** (*OverlapsRect*): *region×pravougaonik→bool* (Testira da li region presjeca pravougaonik.)
- **Isjeca** (*Clipping*): *region×pravougaonik→region* (Računa presjek regiona i pravougaonika: region, moguće prazan.)

- **Presjek** (*Intersection*): *region×region→region* (Vraća presjek dva regiona: region, moguće prazan.)
- **Susreće** (*Meet*): *region×region→bool* (Testira susjedstvo dva regiona.)
- **Površina** (*Area*): *region→real* (Vraća vrijednost površine regiona.)
- **UnijaRegiona** (*RegionUnion*): {*region*} →*bool* (uzima skup regiona i vraća region, uniju uzetih regiona. Ova funkcija se može gledati kao agregatna funkcija u SQL terminologiji, slično *min, max* i *sum*, među drugim funkcijama.)

ATP linija uključuje sljedeće operacije:

- **TačkaULiniji** (*PointInLine*): *linija×tačka→bool* (Testira presjek između tačke i linje.)
- **Dužina** (*Lenght*): *linija→real* (Računa dužinu linije.)
- **PreklapaLR** (*OverlapsLR*): *linija×region→bool* (Testira presjek linije i regiona.)

ATP tačka uključuje sljedeću operaciju:

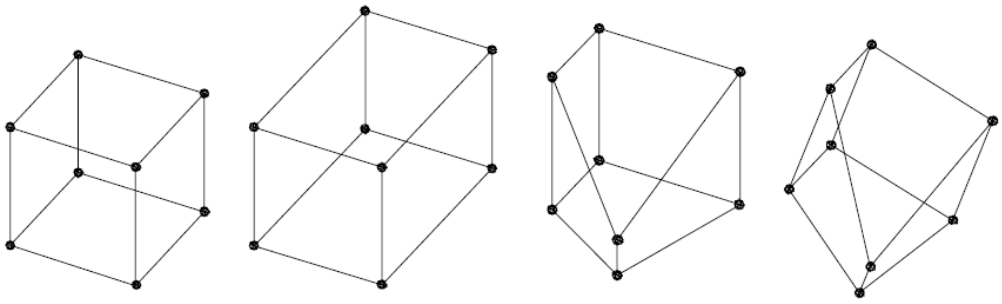
- **Dužina** (*Distance*): *region×tačka→real* (Računa udaljenost između tačke i granice regiona.)

Ova lista nije potpuna. Njena namjena je da obezbjedi čitaocu jednostavan skup operacija dovoljnih za postavljanje upita. Treba napomenuti da nekoliko intuitivnih i korisnih operacija nije moguće definirati, zbog restrikcije u tipovima podataka. Npr., unija regiona je ponovo region, ali ovo ne bi vrijedilo za uniju linija. Slično, ne definišemo presjek linije i pravougaonika ili regiona, jer rezultat nije uvijek lista parova povezanih segmenata. U stvarnoj implementaciji trebamo uzeti u obzir ova ograničenja ali ovo čini sistem tipova zapetljanijim. (Rigaux, Scholl, & Voisard, 2002)

2. TOPOLOŠKI KONCEPTI

2.1 Model 9 presjeka

Topologija je grana matematike koja proučava prostorne odnose između geometrijskih objekata. Najvažnija osobina tih odnosa je da se ne mjenjaju prilikom primjene topoloških transformacija (translacija, rotacija, promjena mjerila, rastezanje, gnječenje, i sl.). Dakle, topologija se bavi samo nemetričkim prostorom što znači da metričke vrijednosti (koordinate, dužine, uglovi,...) nisu razmatrane.



Slika 11: Rezultati topoloških transformacija (Matijević, 2004)

Jedan od temeljnih koncepata potrebnih za analizu prostornih podataka u GIS-u je razumjevanje geometrijskih veza između proizvoljnih prostornih objekata. Kategorizacija takvih binarnih topoloških relacija između regija, linija i tačaka se bazira na upoređivanju 9 presjeka između unutrašnjosti, granica i vanjštine dva objekta. Osnovni kriterij za razlikovanje različitih topoloških veza je postojanje ili nepostojanje presjeka, tj. da li je presjek prazan ili ne. Na osnovu toga može se pokazati da teoretski postoji 2^9=512 međusobno isključivih topoloških veza. Izvedeno je koje od ovih 512 binarnih veza ustvari postoji u $\mathbf{R^2}$ (2D prostoru) između regija, linija i tačaka. (Egenhofer & Herring, 1994)

Ispitivanje prostornih veza je težak zadatak koji zahtijeva multidisciplinarni napor. Predikati kao „daleko od“ ili „dodiruje“ nisu jasni kao veze koje postoje npr. između realnih brojeva. Potrebna je formalna definicija prostornih veza da bi se razjasnila različita razumijevanja prostornih veza među korisnicima, te da bi se zaključile veze između prostornih objekata. (Rigaux, Scholl, & Voisard, 2002)

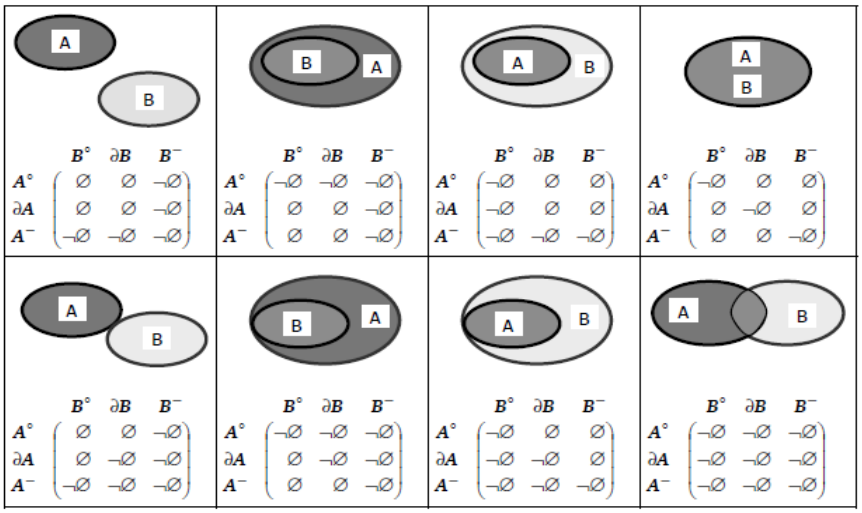
Da bi se rješili navedeni problemi uveden je model 9 presjeka kojim se ispituju prostorne veze između objekata u prostornoj bazi podataka. Binarna topološka relacija R između dva geometrijska objekta Ai B određuje se uporedbom unutrašnjosti (A^o), granice (∂A) i vanjštine

(A^-) objekta A, i unutrašnjosti (B^o), granice (∂B) i vanjštine (B^-) objekta B. Ovih šest objektnih dijelova moguće je kombinirati tako da stvaraju devet temeljnih opisa topoloških relacija između dva geometrijska objekta. To su presjeci(Galić, 2006):

1. unutrašnjosti A s unutrašnjosti B, označen s $(A^o \cap B^o)$,
2. unutrašnjosti A s granicom B, označen s $(A^0 \cap \partial B)$,
3. unutrašnjosti A s vanjštinom B, označen s $(A^o \cap B^-)$,
4. granice A s granicom B, označen s $(\partial A \cap \partial B)$,
5. granice A s unutrašnjosti B, označen s $(\partial A \cap B^o)$,
6. granice A s vanjštinom B, označen s $(\partial A \cap B^-)$,
7. vanjštine A s vanjštinom B, označen s $(A^- \cap B^-)$,
8. vanjštine A s granicom B, označen s $(A^- \cap \partial B)$, i
9. vanjštine A s unutrašnjosti B, označen s $(A^- \cap B^o)$.

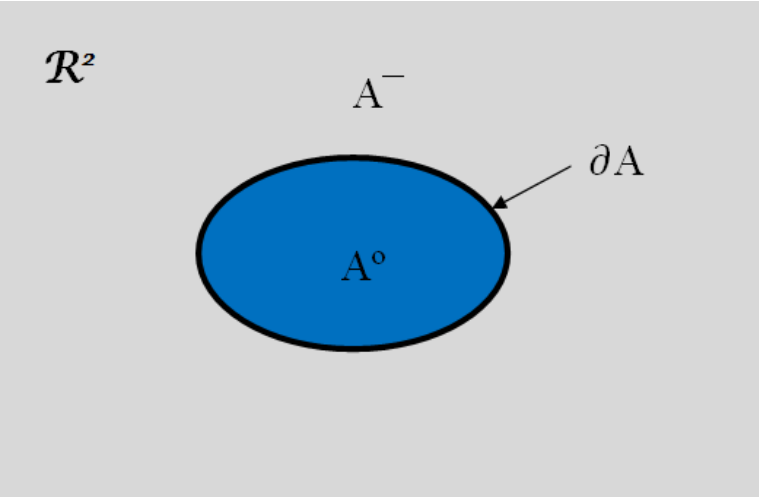
Okvirzaopisivanjetopološkihrelacija između dvije ćelije A i B, je uređeniskupovihdevetpresjeka kojisemožetačnoiskazatimatricomformata 3 x 3:

$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

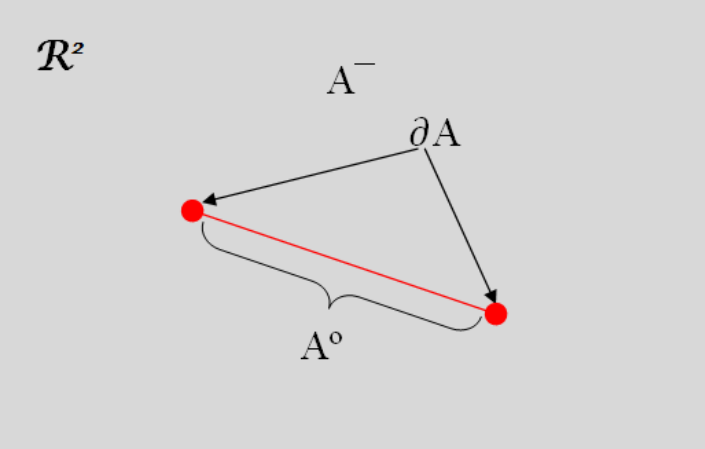


Slika 12: Geometrijska interpretacija 8 relacija među jednostavnim regijama (Galić, 2006)

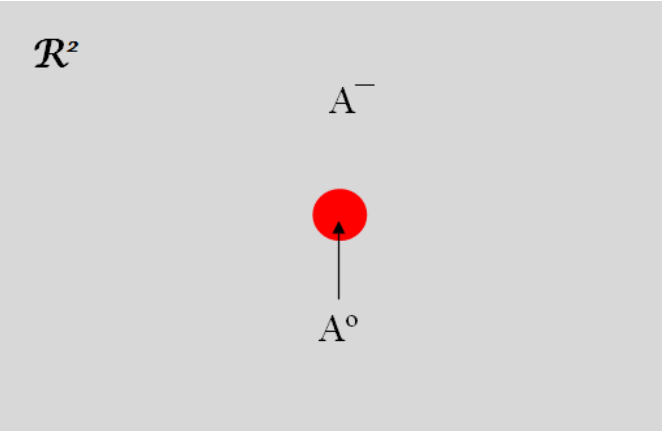
Unutrašnjost, granica i vanjština za poligone, linije i tačke



Slika 13: Unutrašnjost, granica i vanjština poligona



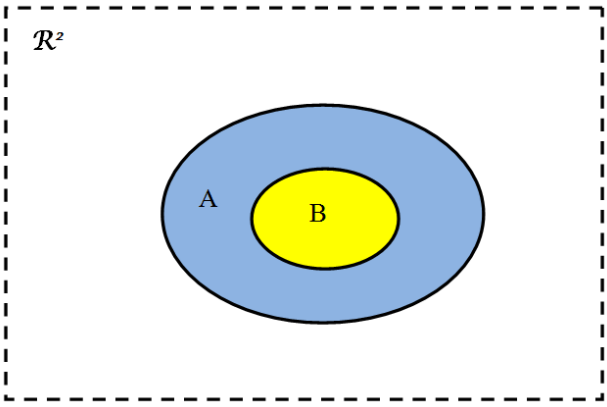
Slika 14: Unutrašnjost, granica i vanjština linije



Slika 15: Unutrašnjost i vanjština tačke

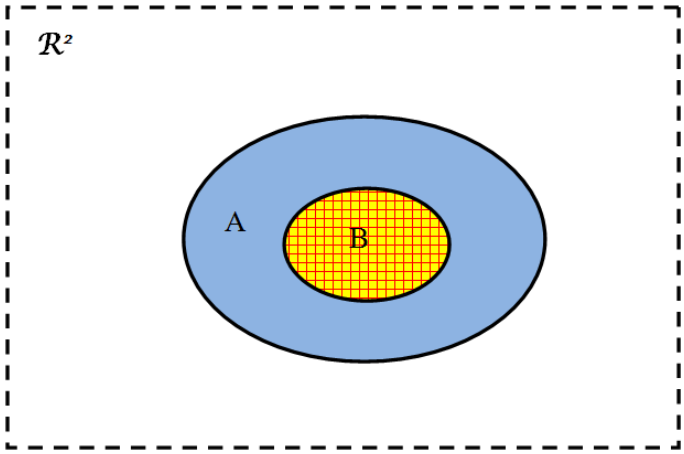
Primjer 1:

Napraviti matricu 9 presjeka za poligone A i B na slici:



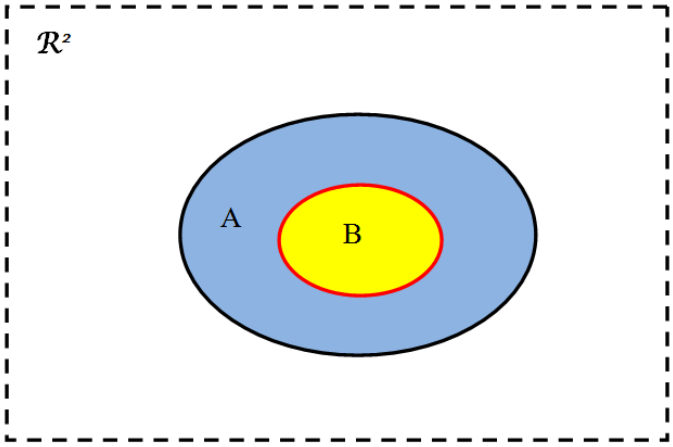
$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

1. $A^o \cap B^o$



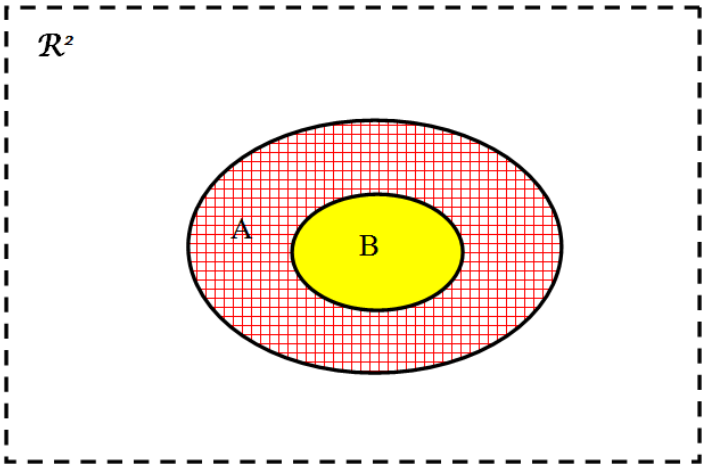
$$R(A,B) = \begin{pmatrix} \boldsymbol{A^o \cap B^o} & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

2. $A^0 \cap \partial B$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & \boldsymbol{A^0 \cap \partial B} & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

3. $A^o \cap B^-$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & \boldsymbol{A^o \cap B^-} \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

4. $\partial A \cap B^o$

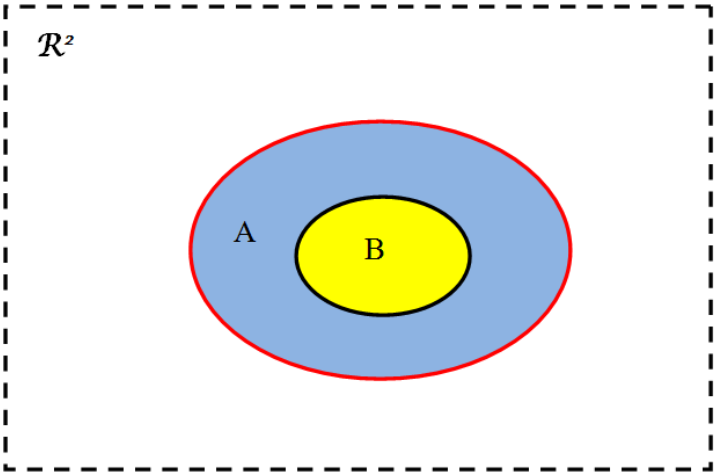
$$\boldsymbol{\partial A \cap B^o = \emptyset}$$

5. $\partial A \cap \partial B$

$$\boldsymbol{\partial A \cap \partial B = \emptyset}$$

$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \boldsymbol{\partial A \cap B^o} & \boldsymbol{\partial A \cap \partial B} & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & ? \\ ? & ? & ? \end{pmatrix}$$

6. $\partial A \cap B^-$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \boldsymbol{\partial A \cap B^-} \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ ? & ? & ? \end{pmatrix}$$

7. $A^- \cap B^o$

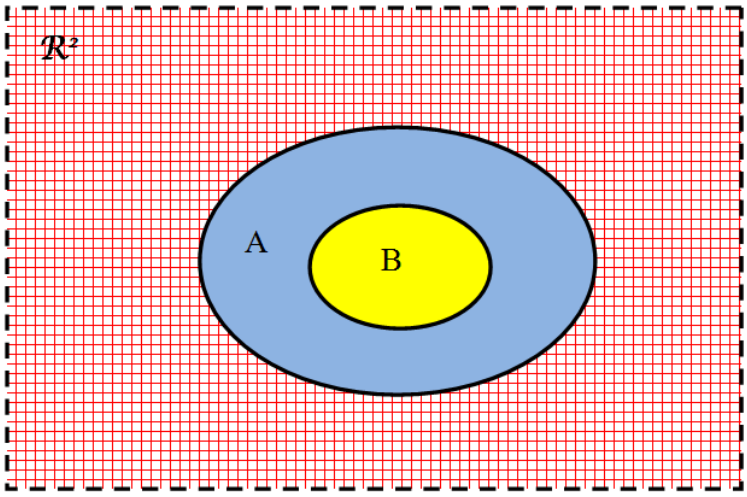
$$\boldsymbol{A^- \cap B^o = \emptyset}$$

8. $A^- \cap \partial B$

$$\boldsymbol{A^- \cap \partial B = \emptyset}$$

$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ \boldsymbol{A^- \cap B^o} & \boldsymbol{A^- \cap \partial B} & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & ? \end{pmatrix}$$

9. $A^- \cap B^-$

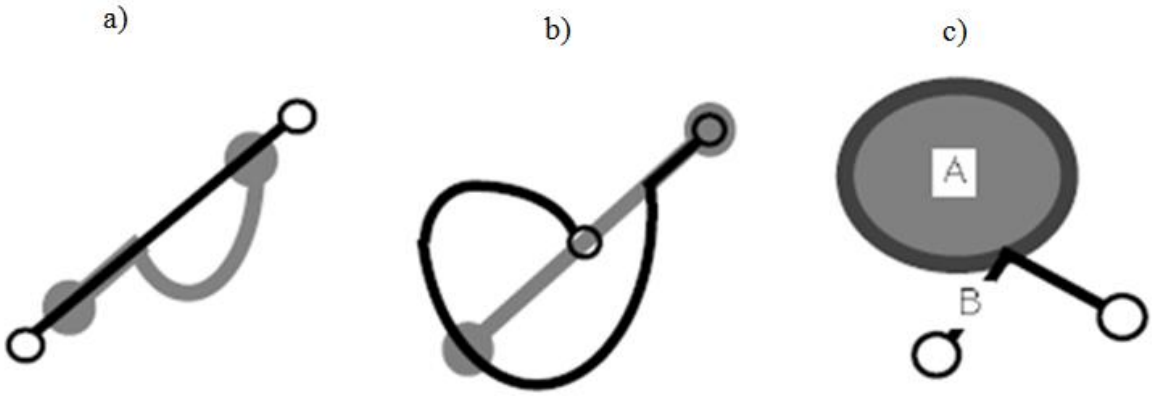


$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & \boldsymbol{A^- \cap B^-} \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

Zadaci za vježbu:

1. Ispiši matrice 9 presjeka za sljedeće slučajeve:

   (Napomena: Crna linija je A, a siva B!)



2. Nacrtaj neku od mogućih kombinacija dvije geometrije kao primjer sljedećih matrica 9 presjeka:

a) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

b) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

c) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

3. Nacrtaj sve topološke relacije između tačke i linije, a zatim ispiši odgovarajuće matrice za svaki od slučajeva!

2.2Dimenzijski prošireni model 9 presjeka (DE-9IM)

Nedostatak modela 9 presjeka (9-IM) je što razlikuje samo prazne i neprazne presjeke između granice i unutrašnjosti geometrijskih objekata, i rezultira u prevelikom broju relacija koje bi se morale definisati i implementirati kao operacije nad apstraktnim tipovima podataka. Zbog ovih nedostataka model 9-presjeka proširujemo dimenzijom rezultata presjeka, uz definisanje minimalnog broja relacija, od kojih svaka ima semantički jasan naziv - predikat. Ovaj model nazivamo dimenzijskim proširenim modelom 9-presjeka (DE-9IM). (Galić, 2006)

Tačke, linije i poligone označit ćemo kao *T*, *L* i *P*, a ako je neophodno razlikovat ćemo dva objekta istoga tipa i označiti ih indeksima L_1i L_2. Za definisanje topoloških relacija koristit ćemo sljedeći skup operatora: granica (∂), unutrašnjost (°), vanjština ($\bar{}$) i dimenzija (*dim*). Taj zadnji operator je zapravo funkcija koja vraća dimenziju skupa tačaka ili *null* (−) za prazan skup. Dimenzijski prošireni model 9-presjeka (DE-9IM) dobije se jednostavnim proširenjem svakog presjeka u 9-IM modelu njegovom dimenzijom:

$$DE9I = \begin{pmatrix} \dim(A^o \cap B^o) & \dim(A^0 \cap \partial B) & \dim(A^o \cap B^-) \\ \dim(\partial A \cap B^o) & \dim(\partial A \cap \partial B) & \dim(\partial A \cap B^-) \\ \dim(A^- \cap B^o) & \dim(A^- \cap \partial B) & \dim(A^- \cap B^-) \end{pmatrix}$$

Matrica presjeka sadrži skup od devet vrijednosti presjeka *p* (po jedna za svaki element matrice). Te vrijednosti presjeka mogu se izraziti kao skup čiji su elementi T, F, *, 0, 1, 2. Za svaki *x* iz skupa presjeka vrijedi:

p = T=> dim(x) □{0,1,2}, tj. x □ □

p = F=> dim(x) = -1, tj. x = □

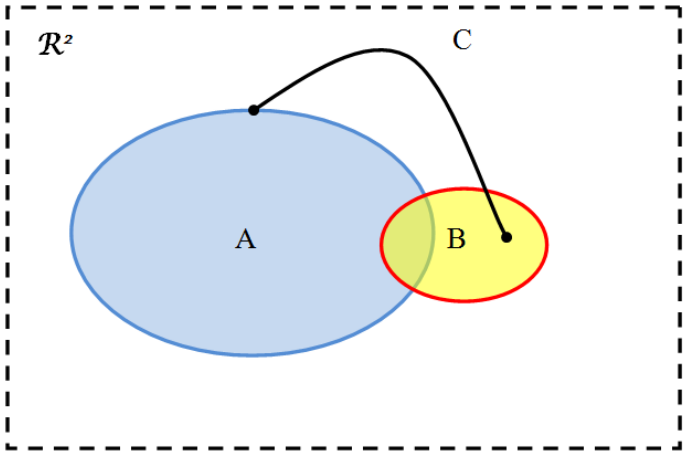
*p = *=> dim(x) □ {-1,0,1,2},* tj. nije bitno

p = 0 => dim(x) = 0

$p = 1 \Rightarrow dim(x) = 1$

$p = 2 \Rightarrow dim(x) = 2$

Primjer: Napisati dimenzijske matrice 9 presjeka za presjeke poligona i presjeke poligona sa linijom na slici ispod!



Rješenje:

$$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \end{pmatrix}$$

$$R(A,C)_{dim} = \begin{pmatrix} A^o \cap C^o & A^0 \cap \partial C & A^o \cap C^- \\ \partial A \cap C^o & \partial A \cap \partial C & \partial A \cap C^- \\ A^- \cap C^o & A^- \cap \partial C & A^- \cap C^- \end{pmatrix} = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 0 & 1 \\ 1 & 0 & 2 \end{pmatrix}$$

$$R(B,C)_{dim} = \begin{pmatrix} B^o \cap C^o & B^0 \cap \partial C & B^o \cap C^- \\ \partial B \cap C^o & \partial B \cap \partial C & \partial B \cap C^- \\ B^- \cap C^o & B^- \cap \partial C & B^- \cap C^- \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & -1 & 1 \\ 1 & 0 & 2 \end{pmatrix}$$

Korisnik nije direktno zainteresiran za presjeke granica, unutrašnjosti i vanjštine, nego za uobičajene relacije (susjedstvo, preklapanje, itd.) među objektima. Iako pojam granice korisniku može biti poznat, pojmovi unutrašnjosti i vanjštine mogu biti slabije razumljivi, jer su zasnovani na formalnoj, matematičkoj teoriji skupova tačaka.

Definicije topoloških relacija:

Disjoint (razdvaja)

Zadvijedate (topološkizatvorene) geometrije*A*i*B*:

A.Disjoint(B)□ A □ B = □

Touches (dodiruje)

Relacija između dvije geometrije A i B (za P/P, L/L, L/P, T/P i T/L):

A.Touches(B) □ (A° □ B° = □)□ (A □ B) □ □

Crosses (prelazi preko)

Relacija za T/L, T/P, L/L i L/P:

A.Crosses(B) □ (dim(A° □ B°) < max (dim(A°), dim (B°)))□ (A □ B□ A)□ (A □ B□ B)

Within (unutar)

Relacija Within definira se kao:

A.Within(B)□ (A□ B = A)□ (A° □ B°□ □)

Overlaps (preklapa)

Relacija Overlaps je definirana za slučajeve P/P, L/L i T/T, kao:

A.Overlaps(B)□ (dim(A°)=dim(B°)=dim(A°□ B°))□(A□ B□ A)□(A□ B□ B)

Contains (sadrži)

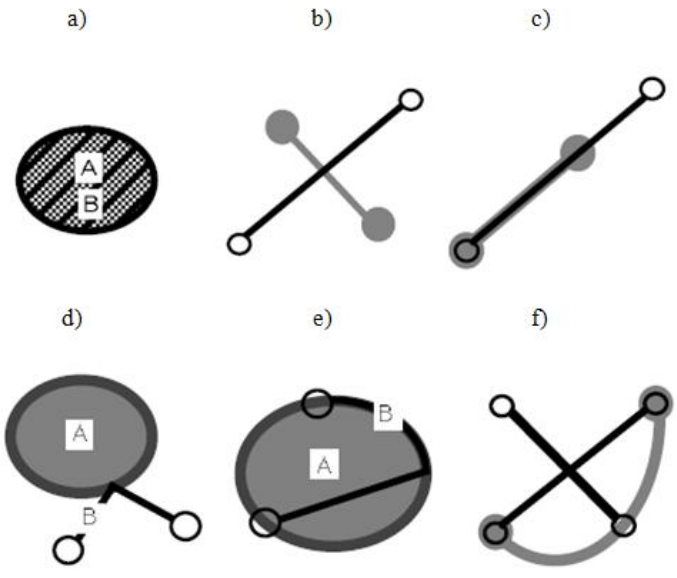
A.Contains(B)□ B.Within(A)

Intersects (siječe)

A.Intersects(B)□ ! A.Disjoint(B)

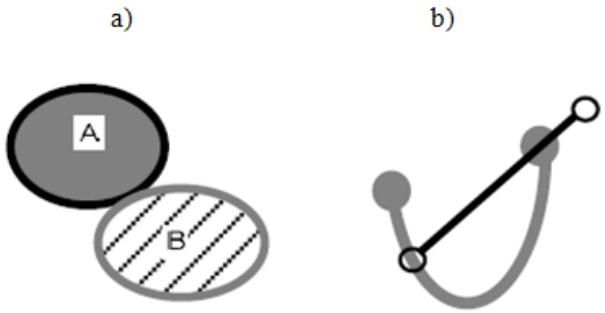
Zadaci za vježbu:

1. Nacrtaj dimenzijske matrice 9 presjeka za sljedeće slučajeve:
   (Napomena: Crna linija je A, a siva B!)



2. Pronađite greške u dimenzijskim matricama 9 presjeka!

   (Napomena: Crna linija je A, a siva B!)



a) $$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 1 & 1 \\ 2 & 0 & 2 \end{pmatrix}$$

b) $$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

3. SQL

Strukturirani jezik za upite ili SQL (eng. *Structured Query Language*) razvio je IBM (eng. *International Business Machines)* u kasnim 70-tim godinama prošlog stoljeća. Jezik se postepeno usavršavao, a njegova dotjerana varijanta pojavljuje se u današnjem IBM relacijskom SUBP-u pod imenom DB2 (od eng. *Database Managment System - DBMS*). Manje softverske tvrtke ugradile su SQL u svoje SUBP-ove pa je postao dostupan širem spektru korisnika što je doprinjelo njegovoj popularnosti. Ostale softverske tvrtke (npr. Ingres Corp, DEC i dr.) su već bile razvile svoje vlastite jezike, ali su bile prisiljene prilagoditi se trendu. Zbog pojave više verzija SQL-a, donesen je zajednički standard, ISO-ANSI za SQL jezik. Taj standard se stalno mijenja u skladu sa razvojem računara i potreba korisnika.

SQL je standard prema ANSI (eng. *American National Standards Institut*), ISO (eng. *Organization for International Standardization*), Unix (X/Open), IBM Standard i FISP (eng. *Federal Information Processing Standard*). Prva verzija standardnog SQL-a objavio je ANSI 1961. i ta je verzija sadržavala velik dio upitnog dijela jezika, ali sa znatno ograničenim mogućnostima za definiciju podataka. Standard je proširen 1989. godine i poznat je pod nazivom SQL-89. Novi standard objavljen je već 1992. godine i poznat pod nazivom SQL-92, ali također i pod nazivom SQL-2. Standard je donio veliki broj novih elemenata jezika, ali je uglavnom bio kompatibilan s prethodnim. Najnoviji standard poznat pod nazivom SQL-99 ili SQL-3 sadržava nove mogućnosti poput ugradnje aktivnih pravila i okidača, rekurzivnih operacija, novih tipova podataka itd. U komercijalnim sistemima za upravljanje bazama podataka SQL-2 standard je ovog trenutka najšire prihvaćen. Ipak, zbog njegove složenosti do sada još niti jedan sistem ne podržava taj standard u potpunosti. Prema stepenu udovoljavanja standardu, sistemi se klasificiraju u tri kategorije: *Entry SQL*, *Intermediate SQL* i *Full SQL*. Entry SQL kategorija je slična SQL 99 standardu, Intermediate SQL pokriva mogućnosti standarda koje su najznačajnije za primjenu u danas raspoloživim komercijalnim proizvodima. Proizvođači komercijalnih sistema također ugrađuju i svoje, uglavnom nestandardne, DDL (eng. *Data Definition Language*) i DML (eng. *Data Manipulation Language*) naredbe. Ti su nestandardni dijelovi problematični jer programski kod postaje nepremostiv između različitih SQL sistema, a također se bitno otežava usuglašavanje oko budućih standarda.(Mlinarić, 2006)

Osnovu za sve ove jezike predstavljaju dva jezika koje je *Codd* nazvao podjezicima baze podataka. Pod pojmom podjezika baze podataka *Codd* podrazumijeva jezik za pretraživanje i

jezik za ažuriranje baze podataka iz kojih su odstranjene komponente za njihovo izvođenje na računalu, a to su relacijska algebra i relacijski račun. **Relacijska algebra** je model proceduralnog jezika koji sadrži niz operacija nad relacijama. Operacije relacijske algebre su tako izabrane da se njihovim kombinovanjem relativno jednostavno mogu izvesti ažuriranje i pretraživanje baze podataka. **Relacijski račun** zasniva se na računu predikata prvog reda i spada u neproceduralne jezike. Ova dva podjezika su ekvivalentna po mogućnostima i bilo koji izraz u relacijskom računu može biti transformiran u semantički ekvivalentan niz operacija relacijske algebre. Kao rezultat djelovanja pojedinih operacija nad relacijama dobivamo ponovo neku novu relaciju.(Mlinarić, 2006)

Važna osobina jezika je neproceduralnost – opisuje se što se želi dobiti kao rezultat, ali ne i kako se do tog rezultata dolazi. SQL objedinjuje funkcije jezika za definisanje podataka (DDL), jezik za upravljanje podacima (DML) i jezik za postavljanje upita (*Query Language - QL*).

3.1 Definisanje i upravljanje podacima u bazi podataka

Kao što je već navedeno SQL objedninjuje funkcije DDL-a i DML-a (jezika za definisanje podataka i jezika za manipulisanje podacima), osim toga također se koristi za pretraživanje baze podataka korištenjem upita. Pojednostavljeno, DDL služi za definiciju objekata (tabela, pogleda, indeksa, itd.) u bazi podataka, a DML omogućava manipulaciju objektima (unos, izmjena, brisanje, i sl.).

Tipovi podataka (URL 2)

Za definisanje i manipulaciju podacima u SQL-u su dostupni sljedeći tipovi podataka:

- BOOLEAN
  Vrijednost može biti samo tačno (*eng. true*) ili netačno (*eng. false*). Ove vrijednosti se mogu porediti međusobno ili sa logičkim izrazom. Za pohranu podataka se koristi 1 bajt.
- INTEGER ili INT
  Cijeli broj pohranjen u 4 bajta (*eng. byte*), tj. 4*8=32 bita (*eng. bit*). Dopušteni raspon brojeva je od $-(2^{n-1})$ do $2^{n-1}-1$ (n=32), tj. od -2147483648 do 2147483647. Međutim, vrijednost -2147483648 se koristi za pohranu NULL vrijednosti tako da je raspon brojeva ustvari od -2147483647 do 2147483647.
- SMALLINT

Cijeli broj pohranjen u 2 bajta. Raspon brojeva koji se mogu prikazati je od -32767 do 32767.

- CHARACTER ili CHAR

  Znakovni niz (eng. String), npr. CHAR(24) znači da je maksimalna dužina niza znakova 24.

- NCHAR (proširenje standarda)

  Kao i CHAR tip podataka, razlika je u tome što ima dodatak kojim se omogućava ispravno leksikografsko uređenje nizova znakova koji sadrže nacionalne kodne stranice (*eng. character set*).

- DATE

  Vrijednosti ovoga tipa se uvijek prikazuju u obliku datuma. Format ispisa određen je varijablom DBDATE (varijabla okruženja koja specificira format datzma krajnjeg korisnika, npr. DD.MM.YYYY. ili MM.DD.YYYY.). Oblik datumske konstante uvijek se piše pod navodnicima, npr. „25.10.2012.“. Ovaj tip podatka zauzima 4 bita memorije. Kod ovoga tipa podataka moguće su operacije sabiranja i oduzimanja.

datum1-datum2	rezultat je cijeli broj koji jednak broju dana proteklih između datuma 1 i 2;
datum+broj	rezultat je datum koji će biti kada od odabranog datuma prođe odabrani broj dana;
datum-broj	rezultat je datum koji je bio prije odabranog datuma za odabrani broj dana.

- FLOAT

  Koristi se za prikaz decimalnih vrijednosti i odgovara DOUBLE (DOUBLE PRECISION) tipu podataka u C jeziku. Najčešće se radi o prostoru za pohranu veličine 8 bajtova.

- SMALLFLOAT (proširenje standarda)

  Kod ovoga tipa podataka raspon brojeva i preciznost zavisi od operativnog sistema, mada se najčešće radi o 4 bajtnom prostoru za pohranu.

- DECIMAL

  Kod definicije ovoga tipa podataka specificira se ukupan broj cifri (*eng. precision,* m<=32) i broj cifri koje se nalaze iza decimalne tačke (*eng. scale,* n<=m). Dakle, ako definišemo broj sa DECIMAL (10,3) to znači da se radi o broju od 10 cifri od kojih je 7 ispred decimalne tačke i tri iza (scale=3). Ako „scale“ nije navedeno varijabla se

smatra realnim brojem koji ima preciznost m, a dopušteni interval vrijednosti je od 10^{-130} do 10^{124}. Koliko memorije će zauzeti broj zavisi od odabira broja cifri, potreban prostor je m/2+1 bajtova. Nekada se za ovaj tip koriste i nazivi NUMERIC i DEC.

- MONEY (proširenje standarda)

  DECIMAL tip podataka sa dodatkom valute. Oznaka valute je određena varijablom DBMONEY. Predefinirana vrijednost za DBMONEY je "$.". Ovo znači da ako vi odaberete da je tip podataka MONEY bez ikakvih dodatnih definisanja tada će valuta biti dolar i koristit će se decimalna tačka kao separator.Kako je oznaka valute postavljena ispred decimalne tačke to znači da će valuta biti postavljena ispred cifre, npr. $1234.56 (da je ".$" bilo bi 1234.56$). Ako želite da promijenite valutu (npr. u euro tako da se koristi decimalna tačka i da oznaka valute ide iza cifri) koristit ćete sljedeću naredbu: set DBMONEY=.€

- DATETIME

  Ovaj tip podataka pohranjuje informaciju o trenutku u vremenu. Preciznost se može odrediti korištenjem sintakse DATETIME max_kval TO min_kval, gdje su kvalifikatori:

  YEAR - godina (1 do 9999);
  MONTH - mjesec (1 do 12);
  DAY - dan (redni broj dana u mjesecu, dakle od 1 do 31);
  HOUR - sat (0 – ponoć do 23);
  MINUTE - minuta (0 do 59);
  SECOND - sekunda (0 do 59);
  FRACTION - djelić sekunde (preciznost do 5 cifri).

  Ako definišemo varijablu sa DATETIME YEAR TO FRACTION (2), to znači da će biti pohranjena informcija o: godini, mjesecu, danu, satu, minuti, sekundi i stotim dijelovima sekunde (npr. 2011-05-22 10:25:12.33).

- INTERVAL (proširenje standarda)

  Koristi se za pohranu informacija o trajanju. Preciznost trajanja intervala se definiše korištenjem sintakse INTERVAL max_kval(n) TO min_kval(n) gdje je n<=9, osim za dijelove sekunde (FRACTION) gdje je n<=5. Kvalifikatori su istio kao kod DATETIME tipa podataka. Postoje dvije osnovne grupe:

  - Interval *godina-mjesec*, u ovoj grupi mogu se pohraniti intervali sa određenom godinom i mjesecom, npr. INTERVAL YEAR (3) TO MONTH znači da se može pohraniti informacija o trajanju od maksimalno 999 godina i 12 mjeseci.

- Interval *dan-vrijeme*, u ovoj grupi mogu se pohraniti intervali sa granicama određenim danom, satom, minutom, sekundom i dijelovima sekundi, npr. INTERVAL DAY (4) TO FRACTION (2) pohranjivat će informaciju o trajanju vremena od maksimalno 9999 dana, 24 sata, 60 minuta, 60 sekundi i 99 stotih dijelova sekunde.

- CHARACTER VARYING

  Niz znakova varijabilne dužine pri čemu se minimalna i maksimalna dužina mogu definisati. Količina utrošene memorije zavisi od odabira dužine s tim da se uvijek troši barem onoliko koliko je specificirano minimalnom dužinom. Maksimalna dužina niza je 255.

- BYTE

  Pripada grupi BLOB (*eng. Binary Large OBject*) tipova podataka. Pomoću ovoga tipa podataka moguće je pohraniti bilo koji neprekinuti niz binarnih podataka bez obzira na veličinu. Količina utrošene memorije odgovara veličini stvarno pohranjenog objekta.

- TEXT

  Slično kao BYTE tip podataka s tim da je moguće pohraniti samo tekstualne podatke. I kod ovoga tipa podataka utrošak memorije odgovara stvarnoj veličini pohranjenog objekta.

Napomena: Poglavlja 3.1.1, 3.1.2, 3.2, 3.3, 3.4 i 3.5 su prevedena iz knjige „Spatial databases – A tour“.

3.1.1 Jezik za definisanje podataka – *Data definition language* (DDL)

DDL se koristi za kreiranje, brisanje i modificiranje definicija tabela u bazi podataka. Kreiranje relacijskih šema te dodavanje i brisanje tabela je definisano u DDL komponenti SQL-a. U nastavku će biti definisane relacije Grad i Država.

```
CREATE   TABLE Grad {
         Ime   VARCHAR (35),
         Država   VARCHAR (35),
         Populacija   INT,
         Glavni grad   CHAR (1),
         Oblik   CHAR (13)}
```

Naredba **CREATE TABLE** se koristi da definiše relacije u relacijskoj šemi. Ime tabele je GRAD. Tabela ima četiri kolone, a ime svake kolone i odgovarajući tip podataka se moraju specificirati. U SQL92 mogući tipovi podataka su fiksirani i ne mogu biti korisnički definirani. Atributi *Ime* i *Država* moraju biti ASCII (eng. *American Standard Code for Information Interchange*) znakovi tipa *string* i moraju biti kraći od 35 znakova. Atribut *Populacija* je cjelobrojni tip podataka (*integer*), a *Glavni grad* je atribut koji se označava jednim znakom D ili N. Konačno, atribut *Ime* je primarni ključ relacije. Ovo znači da svaki red u tabeli mora imati jedinstvenu vrijednost za atribut *Ime*.

```
CREATE   TABLE Država {
         Ime   VARCHAR (35),
         Kontinent   VARCHAR (35),
         Populacija   INT,
         BDP   INT,
         Oblik   CHAR (15)}
```

Na same tabele i atribute tabela se mogu stavljati ograničenja. Postoji ukupno 5 ograničenja, to su: UNIQUE KEY, PRIMARY KEY, FOREIGN KEY, NOT NULL i CHECK.

```
CREATE   TABLE Država {
         Ime   VARCHAR (35) NOT NULL,
         Kontinent   VARCHAR (35),
         Populacija   INT,
         BDP   INT,
         Oblik   CHAR (15),
         PRIMARY KEY   (Ime) }
```

Tabele koje se više ne koriste se mogu ukloniti iz baze podataka pomoću komande **DROP TABLE**. Npr. Ako želimo ukloniti tabelu Rijeka to ćemo učiniti na sljedeći način:

```
DROP TABLE Rijeka;
```

Za modificiranje relacijske sheme se koristi komanda **ALTER TABLE**, npr. dodavanje novih atributa, modificiranje postojećih atributa, definisanje default-ne vrijednosti za novi atribut, brisanje atributa.

Ako u tabeli Država želimo izbrisati kolonu BDP to ćemo učiniti na sljedeći način:

```
ALTER TABLE Država
```

DROP BDP;

Ako u tabeli Država želimo dodati ime predsjednika države uradit ćemo sljedeće:

ALTER TABLE Država

ADD Predsjednik (CHAR(30));

3.1.2 Jezik za upravljanje podacima – Data modification language (DML)

Nakon što se tabela napravi kao što je objašnjeno u prethodnom dijelu spremna je za pohranu podataka. Ovaj zadatak, koji se često naziva „naseljavanje tabele“, se vrši pomoću DML-a. Npr., sljedeća naredba dodaje jedan red u tabelu *Rijeka*:

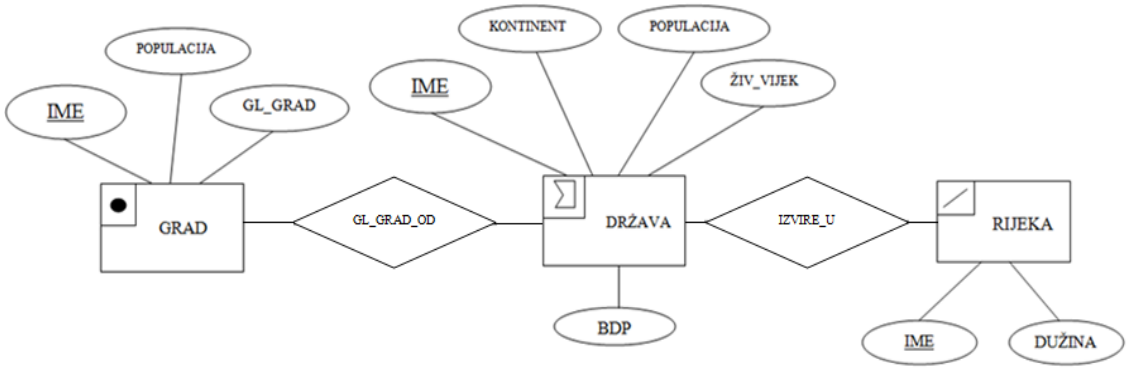
INSERT INTO Rijeka (Ime, Izvor, Dužina)
VALUES („Misisipi“, „SAD“, 6000)

Ako nisu specificirani svi atributi relacije tada im se dodjeljuju predodređene vrijednosti (eng. *default values*). Najčešće korištena predodređena vrijednost je NULA. Pokušaj dodavanja još jednog reda sa imenom „Misisipi“ će biti odbijen od strane SUBP-a zbog ograničenja primarnog ključa koji je specificiran u DDL-u. Osnovni oblik za brisanje redova iz tabele je sljedeći:
DELETE FROM TABLE WHERE < USLOVI >
Npr., sljedeća naredba uklanja red iz tabele Rijeka koji smo prethodno unijeli.
DELETE FROM Rijeka
WHERE Ime=“Misisipi“

3.2 Standardni upitni jezici baze podataka

Korisnici međudjeluju sa podacima ugrađenima u SUBP korištenjem upitnih jezika. Za razliku od tradicionalnih jezika za programiranje, upitni jezici baze podataka su relativno lagani za učenje i korištenje. U ovome poglavlju će biti opisana dva takva upitna jezika. Prvi, relaciona algebra (RA), je formalniji i obično se ne implementira u komercijalnim bazama podataka. Važnost RA leži u činjenici da ona čini srž SQL-a, najpopularnijeg i široko implementiranog upitnog jezika u bazama podataka.

Uvodimo RA i SQL uz pomoć primjera baze podataka. Baza *Svijet* sadrži tri entiteta: *Država, Grad,* i *Rijeka*. EV dijagram baze, shema baze i tabele su prikazane ispod. Primarni ključ je podvučen (npr. *Ime* je primarni ključ tabele *Država*).



Slika 16: EV dijagram baze *Svijet* (Shekhar & Chawla, 2003)

Država	Ime	Kontinent	Populacija (miliona)	BDP (milijarde)	Životni vijek	Oblik
	Kanada	SAM	30.1	658.0	77.08	Polygonid-1
	Meksiko	SAM	107.5	694.3	69.36	Polygonid-2
	Brazil	JAM	183.3	1004.0	65.60	Polygonid-3
	Kuba	SAM	11.7	16.9	75.95	Polygonid-4
	SAD	SAM	270.0	8003.0	75.75	Polygonid-5
	Argentina	JAM	36.3	348.2	70.75	Polygonid-6

Grad	Ime	Država	Populacija (miliona)	Glavni grad	Oblik
	Havana	Kuba	2.1	D	Pointid-1
	Vošington	SAD	3.2	D	Pointid-2
	Monterej	Meksiko	2.0	N	Pointid-3
	Toronto	Kanada	3.4	N	Pointid-4
	Brazilija	Brazil	1.5	D	Pointid-5
	Rosario	Argentina	1.1	N	Pointid-6
	Otava	Kanada	0.8	D	Pointid-7
	Meksiko Siti	Meksiko	14.1	D	Pointid-8
	Buenos Aires	Argentina	10.75	D	Pointid-9

Rijeka	Ime	Izvor	Dužina (kilometara)	Oblik
	Rio Parana	Brazil	2600	LineStringid-1

St. Lovrens	SAD	1200	LineStringid-2
Rio Grande	SAD	3000	LineStringid-3
Misisipi	SAD	6000	LineStringid-4

Tabela 3: Tabele baze *Svijet* sa uzoračkim podacima (Shekhar & Chawla, 2003)

Entitet *Država* ima šest atributa. *Ime* države i kontinenta kojem pripada su tipa *string* maksimalne dužine 35 znakova. Populacija i bruto društveni proizvod (BDP) su tipa *integer*. BDP je ukupna vrijednost dobara i usluga proizvedenih u jednoj državi u toku fiskalne godine. Atribut *Životni vijek* predstavlja očekivanje trajanja života u godinama (zaokruženo na najbliži *integer*) za stanovnike države. Atribut *Oblik* zahtjeva određena pojašnjenja. Geometrija države je predstavljena kolonom Oblik u prethodnoj tabeli. U relacionim bazama gdje su tipovi podataka ograničeni, atribut *Oblik* je strani ključ za tabelu **oblik**. U objektno-relacionim ili objektnoorjentiranim bazama podataka, atribut *Oblik* će biti poligonalni apstraktni tip podataka (ATP).

Relacija Grad ima pet atributa: *Ime*, *Država*, *Populacija*, *Glavni grad* i *Oblik*. Atribut *Država* je strani ključ u tabeli *Država*. Glavni grad je fiksirani znak dužine jedan (ako je neki grad glavni grad države oznaka je D za Da, ako nije onda je oznaka N za Ne). Atribut Oblik je strani ključ u tabeli oblika **tačka**.

Relacija Rijeka ima četiri atributa: *Ime*, *Izvor*, *Dužina* i *Oblik*. Atribut *Izvor* je strani ključ u relaciji *Država* i specificira državu u kojoj rijeka izvire. Atribut Oblik je strani ključ u tabeli oblika **linija**. Za određivanje države u kojoj izvire rijeka, geometrijska informacija atributa *Oblik* nije dovoljna. Preopterećenje imena u tabelama se može riješiti korištenjem notacije **tabela-tačka-atribut**. *Država.Ime, Grad.Ime* i *Rijeka.Ime* jedinstveno identificiraju atribut *Ime* unutar različitih tabela.

3.3 Relacijska algebra

Relacijska algebra je upitni jezik povezan sa relacionim modelom. Algebra je matematička struktura koja se sastoji od dva posebna skupa elemenata, (Ω_a, Ω_o). Ω_a je skup *operatora*, a Ω_o je skup *operacija*. Algebra mora zadovoljiti mnogo aksioma, ali najvažniji je da rezultat operacija na operatoru mora ostati unutar Ω_a. Jednostavan primjer algebre je skup cijelih brojeva. Tu su operatori cijeli brojevi, a operacije su sabiranje i množenje.

U relacijskoj algebri je samo jedan operator i šest osnovnih operacija. Operator je relacija (tabela), a šest operacija su odabir (eng. *select*), izdvajanje (eng. *project*), unija (eng. *union*),

mješoviti produkt (eng. *cross-product*), razlika (eng. *difference*) i presjek (eng. *intersection*). U nastavku će biti opisane ove operacije.

Operacije odabira (eng. select) i izdvajanja (eng. project)

Za manipulaciju podataka u jendoj relaciji, RA obezbjeđuje dvije operacije: *select* i *project*. Operacija *select* vraća podskup redova relacione tabele, a operacija *project* izdvaja podskup kolona. Npr., da bismo prikazali sve države u tabeli Država koje su u Sjevernoj Americi (SA), koristimo sljedeći izraz relacijske algebre:

$$\boldsymbol{\sigma_{Kontinent} = SA(Država)}$$

Rezultat ove operacije je prikazan u tabeli 3a. Vraćeni redovi operacije *select* su specificirani selekcionim operatorom, koji je u ovome slučaju *kontinent="Sjeverna Amerika"*. Uobičajena sintaksa operacije *select* je:

$$\boldsymbol{\sigma_{<selekcioni\ \ operator>}(Relacija)}$$

Podskup kolona za sve redove u relaciji se dobiva korištenjem operacije *project*, π. Npr., da bismo dobili imena svih država u tabeli Država koristimo sljedeći izraz:

$$\boldsymbol{\pi_{Ime}(Država)}$$

Uobičajena sintaksa operacije *project* je:

$$\boldsymbol{\pi_{<lista\ atributa>}(Relacija)}$$

Moguće je kombinirati *select* i *project* operacije. Sljedeći izraz vraća imena država Sjeverne Amerike. Rezultat je prikazan u tabeli 3c.

$$\boldsymbol{\pi_{Ime}(\sigma_{Kontinent} = SA(Država))}$$

Ime	Kontinent	Populacija (miliona)	BDP (milijarde)	Životni vijek	Oblik
Kanada	SAM	30.1	658.0	77.08	Polygonid-1
Meksiko	SAM	107.5	694.3	69.36	Polygonid-2
Kuba	SAM	11.7	16.9	75.95	Polygonid-4
SAD	SAM	270.0	8003.0	75.75	Polygonid-5

a) *Select*

Ime
Kanada
Meksiko
Brazil
Kuba
SAD
Argentina

Ime
Kanada
Meksiko
Kuba
SAD

b) *Project* c) *Select i Project*

Tabela 4: Rezultati dvije osnovne operacije relacijske algebre (*Select* i *Project*)

Skupne operacije

Na najjosnovnijem nivou relacije je skup. Zbog toga su sve skupne operacije punomoćne operacije u relacijskoj algebri. Skupne operacije su primjenjene na relacijama koje su kompatibilne za povezivanje (eng. *union-compatible*)***. Dvije relacije su kompatibilne za povezivanje ako imaju isti broj kolona, dijele istu domenu, i ako se kolone pojavljuju u istom redosljedu idući s lijeva na desno***. (Shekhar & Chawla, 2003)

- Unija (eng. *union*): Ako su R i S relacije, tada unija $R \cup S$ vraća sve zapise koji su u R ili S. Npr., možemo koristiti operaciju *unija* da prikažemo države koje su u Sjevernoj Americi ili ako u njima izvire neka rijeka:
    1. $R = \pi_{Ime}\,(\sigma_{Kontinent} = SA(Država))$
    2. $S = \pi_{Izvor}\,(Rijeka)$
    3. $R \cup S$

Rezultujuća relacija je prikazana u tabeli 4a. Primjetite da atributi R.Ime i R.Izvor imaju istu domenu jer se R.Izvor odnosi na Država.Ime. Ovo je neophodno da bi relacije R i S bile kompatibilne za spajanje.

- Razlika (eng. *difference*): $R - S$ vraća sve zapise u R koji nisu u S. Operacija razlika se može koristiti kada želimo prikazati sve države u Sjevernoj Americi koje nemaju rijeku (navedenu u tabeli Rijeka) koja izvire u njima. Rezultirajuća relacija je prikazana u tabeli 4b.
    1. $R = \pi_{Ime}\,(\sigma_{Kontinent} = SA(Država))$

2. $S = \pi_{Izvor}\,(Rijeka)$
3. $R - S$.

- Presjek (eng. *intersection*): Za dvije operacije R i S koje su kompatibilne za spajanje operacija presjeka $R \cap S$ vraća sve zapise koji se pojavljuju i u R i u S. Primjetite da iako je operacija prikladna da je također i redundantna jer se može izvesti iz operacije razlike, $R \cap S = R - (R - S)$. Da bismo prikazali države koje su u Južnoj Americi i koje također imaju i rijeku koja u njima izvire koristimo operaciju presjeka. Rezultat je prikazan u tabeli 4c.
    1. $R = \pi_{Ime}\,(\sigma_{Kontinent\ } = JA(Država))$
    2. $R = \pi_{Izvor}\,(Rijeka)$
    3. $R \cap S$.

Ime
Kanada
Meksiko
Brazil
Kuba
SAD

a) Unija

Ime
Kanada
Meksiko
Kuba

b) Razlika

Ime
Brazil

c) Presjek

Tabela 5: Rezultat skupnih operacija

- Unakrsni produkt (eng. *cross-product*): Ova operacija se može koristiti na bilo koji par relacija, a ne samo na one koje su kompatibilne za spajanje. $R \times S$ vraća relaciju čija shema sadrži sve atribute relacije R koje slijede atributi iz S. Radi jednostavnosti, apstraktni primjer je prikazan u tabeli 5. Primjetite korištenje tačke u notaciji da bi se razlikovali atributi dvije relacije.

R	R.A	R.B
	A_1	B_1
	A_2	B_2

S	S.C	S.D
	C_1	D_1
	C_2	D_2

R x S	R.A	R.B	S.C	S.D
	A_1	B_1	C_1	D_1
	A_1	B_1	C_2	D_2
	A_2	B_2	C_1	D_1
	A_2	B_2	C_2	D_2

Tabela 6: Unakrsni produkt relacija R i S

Operacija spajanja (eng. join)

Operacije *select* i *project* su korisne za izdvajanje informacija iz jedne relacije. Operacija spajanja (eng. *join*) se koristi za upite kroz više različitih tabela. Operacija *join* se može smatrati kao kombinacija operacija *cross-product* i *select*. Uobičajena operacija *join* se zove uslovno spajanje (eng. *conditional join*). Važan i poseban slučaj uslovnog spajanja se zove prirodno spajanje (eng. *natural join*).

Uslovna spajanja (eng. *conditional joins*)

Uslovno spajanje, **jc** (*j-join, c-conditional*), između dvije relacije R i S se prikazuje na sljedeći način:

$$R \; \boldsymbol{jc} \; S = \sigma_c(R \times S).$$

Uslov **c** se obično odnosi na atribute i R i S. Npr., možemo koristiti *join* operaciju za traženje imena država koje imaju populaciju veću nego Meksiko.

1. $R = \pi_{Ime\,,Populacija}\ (Država)$
2. $S = R$. (S je kopija R)
3. Formiranje unakrsnog produkta $R \times S$. Šema $R \times S$ relacije je:

$R \times S$	R.Ime	R.Populacija	S.Ime	S.Populacija

4. Primjena uslova (populacija države iz relacije S veća nego populacija Meksika).

$$U = R_{\boldsymbol{jc}}S = \sigma_{(R.Ime\,=\text{Meksiko})\wedge(R.Populacija\ >S.Populacija\ )}(R \times S)$$

$R \times S$	R.Ime	R.Populacija	S.Ime	S.Populacija
	⋮	⋮	⋮	⋮
	Meksiko	107.5	Kanada	30.1
	Meksiko	107.5	Meksiko	107.5
	Meksiko	107.5	Brazil	183.3
	Meksiko	107.5	Kuba	11.7
	Meksiko	107.5	SAD	270.0
	Meksiko	107.5	Argentina	36.3
	⋮	⋮	⋮	⋮

a) Dio od $R \times S$

R.Ime	R.Populacija	S.Ime	S.Populacija
Meksiko	107.5	Kanada	30.1
Meksiko	107.5	Kuba	11.7
Meksiko	107.5	Argentina	36.3

b) Operacija select na $R \times S$

Tabela 7: Koraci operacije uslovnog spajanja

Prirodno spajanje (eng. *natural join*)

Važan specijalan slučaj uslovnog spajanja je prirodno spajanje (eng. *natural join*). Kod prirodnog spajanja relacija spajanje se vrši preko zajedničkih atributa, a u rezultirajućoj relaciji se pojavljuju samo one n-torke (redovi) gdje zajednički atributi imaju iste vrijednosti. Npr., prirodno spajanje se može koristiti da se pronađu populacije država koje imaju izvor rijeke. To možemo uraditi na slijedeći način:

1. Preimenovati relaciju Država u D i relaciju Rijeka u R;
2. Napraviti unakrsni produkt $D \times R$;
3. Spojiti dvije relacije po atriburima D.Ime i R.Izvor. Domene ova dva atributa su identične,

$$C\boldsymbol{j}_{\boldsymbol{D.Ime=R.Izvor}}R;$$

4. Kod prirodnog spajanja selekcioni uslov je nedvosmislen; dakle ne mora biti eksplicitno potpisan u formuli spajanja;
5. Konačni rezultat je dobiven izabirom na osnovu atributa Ime i Populacija:

$$\pi_{Ime\,,Populacija}\ \ (C\ \boldsymbol{j}\ R).$$

Operacija dijeljenja

U relacionoj algebri postoji operator koji se zove DIJELJENJE (end. Division). Ovaj operator traži podskup zapisa u jednoj tabeli koji su povezani sa **svim** zapisima u drugoj tabeli. Ovaj operator se označava znakom „/“ ili „÷“.

Dijeljenje se ne može izvršiti nad svim tabelama. Da bi se dijeljenje moglo izvršiti nad tabelama A i B (A/B) potrebno je da se svi atributi iz tabele B nalaze u tabeli A.

Primjer 1 - operacija dijeljenja:

A

X	Y
0150	01
0250	02
0250	03
0250	04
1070	02
1070	04
1175	02
1175	03

B_1

Z
02

B_2

Z
02
03

Rezultat:

$A(X,Y)/B_1(Z)=C_1(X)$

X
0250
1070
1175

$A(X,Y)/B_2(Z)=C_2(X)$

X
0250
1175

Primjer 2 – operacija dijeljenja:

Relacije *r*; *s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

r/*s*:

A	*B*	*C*
α	a	γ
γ	a	γ

Dakle, u rezultujućoj tabeli će se pojaviti samo one kolone iz tabele A koje ne postoje u tabeli B i samo oni redovi koji čiji dijelovi odgovaraju svim redovima tabele B.

Primjer 1:

Date su relacione šeme:
zaposlenik (osoba-ime, ulica, grad);
posao (osoba-ime, firma-ime, plata);
firma (firma-ime, grad);
upravnik (osoba-ime, upravnik-ime).
Zadaci:

a) Pronaći imena osoba koja rade u firmi “Granit”!
Rješenje:

$$R = \pi_{osoba\ -ime}\ (\sigma_{firma\ -ime}\ = \text{Granit}(posao))$$

b) Pronaći imena i prebivalište osoba koje rade u firmi “Granit”!

$$R = \pi_{osoba\ -ime\ ,grad}\ (zaposlenik \bowtie (\sigma_{firma\ -ime}\ = \text{Granit}(posao))$$

ili

$$ZaposleniGranit = \pi_{osoba\ -ime}\ (\sigma_{firma\ -ime}\ = \text{Granit}(posao))$$

$$R = \pi_{osoba\ -ime\ ,grad}\ (zaposlenik \bowtie ZaposleniGranit)$$

c) Pronaći osobe koje rade za “Granit” i imaju platu veću od 700 maraka! Prikazati imena zajedno sa mjestom prebivališta!

$$R = \pi_{osoba\ -ime\ ,ulica\ ,grad}\ (\sigma_{(firma\ -ime\ =\text{"Granit"}\ \wedge\ plata\ >700)} posao \bowtie zaposlenik)$$

ili

$$ZaposleniGranit = \pi_{osoba\ -ime}\ (\sigma_{firma\ -ime}\ = \text{Granit}(posao))$$

$$Plata700 = \pi_{osoba\ -ime}\ \sigma_{plata\ >700}\ (ZaposleniGranit)$$

$$R = Plata700 \bowtie zaposlenik$$

d) Pronaći imena svih zaposlenika koji rade u firmi koja se nalazi u istom gradu u kome oni žive!

$$R = \pi_{osoba\ -ime}\ (zaposlenik \bowtie posao \bowtie firma)$$

e) Pronaći imena svih zaposlenika koji žive u istom gradu i istoj ulici kao njihovi upravnici!

$$R = \pi_{osoba\ -ime}\ ((zaposlenik \bowtie upravnik)$$

$$\bowtie_{(upravnik\ \ -ime\ =zaposlenikK\ \ .osoba\ -ime\ \wedge\ zaposlenik\ \ .ulica\ =zaposlenikK\ \ .ulica\ \wedge zaposlenik\ \ .grad\ =zaposlenikK\ \ .grad\ )}$$

$$(\rho_{zaposlenikK}\ (zaposlenik)))$$

f) Pronaći sve osobe koje ne rade u firmi "Granit"!

Ako osoba može raditi za samo jednu firmu rješenje je:

$$R = \pi_{osoba\ -ime}\ (\sigma_{firma\ -ime}\ \neq \text{Granit}(posao))$$

Međutim, ako baza dozvoljava da se osoba može nalaziti u tabeli „zaposlenik" ali ne i u tabeli „posao" tada je rješenje:

$$R = \pi_{osoba\ -ime}\ (zaposlenik) - \pi_{osoba\ -ime}\ (\sigma_{(firma\ -ime}\ = \text{"Granit"})(posao))$$

g) Pronaći imena svih koji zarađuju više od svakog zaposlenika firme "Energoinvest"!

$$R = \pi_{osoba\ -ime}\ (posao) - (\pi_{posao\ .osoba\ -ime}\ (posao$$

$$\bowtie_{(posao\ .plata\ \leq posaoK\ .plata\ \wedge posaoK\ .firma\ -ime}\ = \text{"Energoinvest"})_{pposaoK}\ (posao)))$$

h) Pretpostavimo da firme imaju poslovnice u više gradova. Pronaći sve firme koje imaju poslovnicu u istom gradu kao i "Energoinvest"!

$$R = \pi_{firma\ -ime}\ (firma \div \left(\pi_{grad}\ \left(\sigma_{firma\ -ime}\ = \text{Energoinvest}(firma)\right)\right))$$

2. Date su relacione šeme:

student (br-indeksa, ime, adresa, apsolvent);

predmet (sifra, naziv);

registriran (br-indeksa, sifra).

a) Pronaći šifre predmeta na kojima je registrovan barem jedan student!

Ako se na neki predmet registruje student tada se šifra predmeta automatski mora pojaviti u tabeli registrovan tako da je rješenje:

π_{sifra}(registrovan)

b) Pronaći nazive svih predmeta na kojima je registrovan barem jedan student!

U tabeli predmeti se nalaze svi predmeti koji se mogu slušati na fakultetu, dakle tu su i oni predmeti koje niko ne sluša (npr. neki dosadni izborni predmeti). Ako tabelu predmet i registrovani prirodno spojimo izvršit će se filtriranje i ostat će samo oni predmeti na kojima ima registrovanih studenata.

π_{naziv}(predmet⋈registrovan)

c) Pronaći šifre predmeta na kojima nema registrovanih studenata!

Rješenje upita pod a je tabela sa šiframa predmeta na kojima ima registrovanih studenata.

Ako sada od svih predmeta oduzmemo one na kojima su registrovani studenti ostat će nam oni predmeti na kojima nema neregistrovanih studenata. Dakle, rješenje je:

$R = \pi_{sifra}(predmet) - \pi_{sifra}(registrovan)$

d) Pronaći nazive predmeta na kojima nema registrovanih studenata!
(prethodno su pronađene šifre, prirodnim spajanjem sa predmetima pronaći nazive).
Isti princip kao u prethodnom slučaju, nađemo one na kojima su registrovani pa oduzmemo od svih – dobijemo one na kojima nema registrovanih.

e) Pronaći imena studenata i nazive predmeta na kojima su registrovani!

Prirodnim spajanjem izvršit će se filtriranje i u istom redu će biti i ime studenta i naziv predmeta na kojem je taj student registrovan. Iz toga izdvojimo onio šta nam treba (ime studenta i naziv predmeta) i dobijemo rješenje. Ovo ćete najbolje vidjeti ako sebi napravite par malih tabela i izvršite prirodno spajanje.

$\pi_{ime,naziv}(student \bowtie registrovan \bowtie predmet)$

f) Pronaći br-indeksa za studente koji su registrovani na predmetu “Baze podataka” ili “Satelitska navigacija”!

Pošto trebamo naći brojeve indeksa za određene nazive predmeta trebaju se spojiti sve tabele jer tabele student i predmet nemaju direktne veze. Iz te novonastale tabele u prvom slučaju treba izdvojiti redove u kojima je predmet Baze podataka i zatim na njih dodati one zapise u kojima je predmet Satelitska navigacija. Dakle imamo dva upita čiji se rezultati spajaju u jedan da bi se dobio konačan rezultat. Dakle, trebat će nam operator UNIJA jer imamo uslov „ili“.

$\pi_{br.indexa}(\sigma_{naziv = "Baze\ podataka\ "}(student \bowtie registrovan \bowtie predmet)) \cup$
$\pi_{br.indexa}(\sigma_{naziv = "Satelitska\ \ navigacija\ "}(student \bowtie registrovan \bowtie predmet))$

g) Pronaći br-indeksa za studente koji su registrirani na predmetima “Baze podataka” i “Satelitska navigacija”!

Skoro isti upit kao prethodni samo što umjeto UNIJA stavljamo PRESJEK.

h) Pronaći predmete na kojima su registrovani svi studenti!

$\pi_{\text{sifra,br\_indexa}}(\text{registrovan}) / \pi_{\text{br\_indexa}}(\text{student})$

Pretpostavimo da imamo samo 3 studenta radi jednostavnosti:

student	Br.indexa	ime	adresa	apsolvent
	121	Amar	Titova 25	NE
	225	Zoran	Kranjčevićeva 7	DA
	287	Sabina	Koševska 75	DA

Također,radi jednostavnosti pretpostavimo da imamo samo 4 predmeta, i to:

Geodezija – šifra 100, matematika – šifra 200, GIS – šifra 300, Ceste – šifra 400

Tabela

registrovan	Br.indexa	Šifra (predmeta)
	121	100
	225	100
	121	200
	121	400
	287	200
	287	100
	121	300
	225	400

A=$\pi_{\text{sifra,br\_indexa}}$(registrovan), rezultat je čitava tabela „registrovan“ samo što se sada zove „A“

B= $\pi_{\text{br\_indexa}}$(student), rezultat će biti:

B	Br.indexa
	121
	225
	287

Dakle, uslovi za operaciju dijeljenja A/B su ispunjeni jer se svi atributi iz tabele B nalaze u tabeli A.

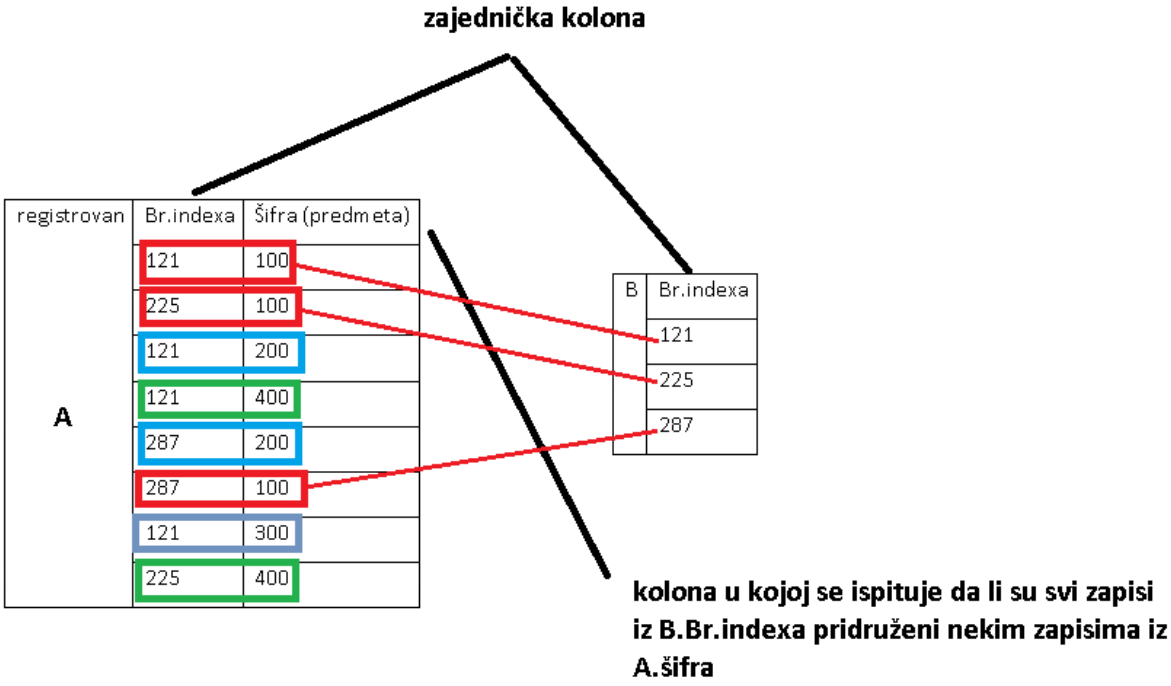
Upit:

$\pi_{\text{sifra,br\_indexa}}(\text{registrovan}) / \pi_{\text{br\_indexa}}(\text{student})$

Odnosno, nakon što smo preimenovali

A/B

Će u tabeli A (i to samo u koloni „šifra“ jer je to jedina kolona u tabeli A koja se razlikuje od tabele B) tražiti zapise kojima su pridruženi svi zapisi iz tabele B.



Jedino je šifri predmeta 100 pridružen svaki broj indexa iz tabele B što znači da su jedino na predmet sa šifrom 100 prijavljeni svi studenti

i) Pronaći spisak predmeta na kojima su svi apsolventi registrovani!

Slično kao u prethodnom primjeru s tim da ovdje imamo još jedan uslov, a to je da se radi o apsolventima.

$\pi_{\text{sifra,br\_indexa}}(\text{registrovan}) / \pi_{\text{br\_indexa}}(\sigma_{apsolvent\ =DA}(\text{student}))$

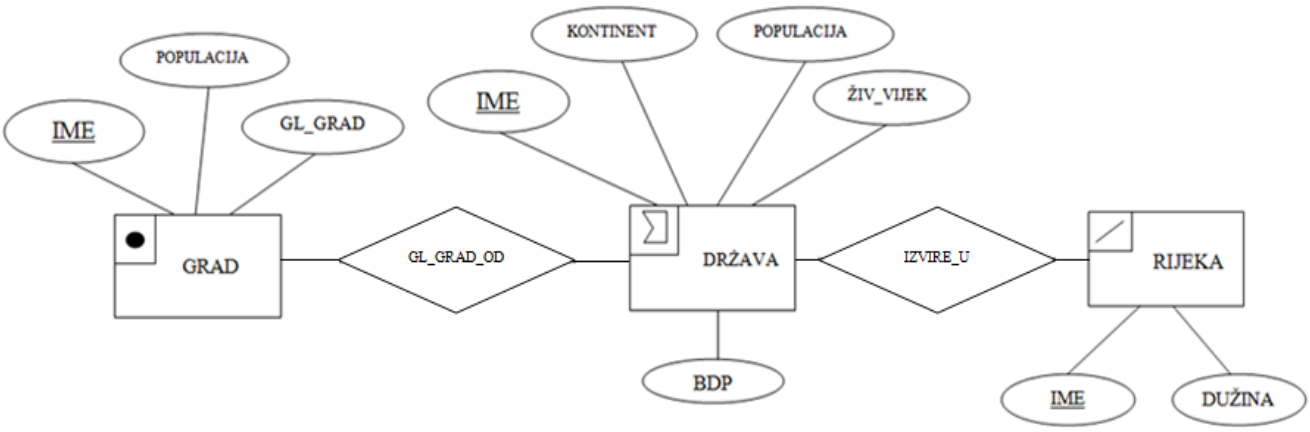
3.4 Relacijski SQL upiti

Kada je shema baze podataka definisana u DDL-u i popunjena, mogu se postavljati upiti u SQL-u da bi se selektovali traženi podaci iz baze podataka. Osnovna sintaksa SQL upita je veoma jednostavna:

SELECT Imena kolona
FROM Relacije
WHERE Uslov

Ovaj oblik je jednak izrazu RA koji se sastoji od *select*, *project* i *join* operacija. Naredba SELECT u SQL-u ima više klauzula koje se odnose na agregaciju (npr. GROUP BY, HAVING), redoslijed rezultata (npr., ORDER BY), itd. Osim navedenog SQL dozvoljava postavljanje *ugnježdenih upita* što će biti prikazano kroz primjere upita. (Shekhar & Chawla, 2003)

Kada je šema baze podataka definirana u DDL-u i „naseljena“ mogu se postavljati upiti u SQL-u da bi se izdvojili potrebi podaci iz baze podataka.



Slika 17: EV dijagram baze *Svijet* (Shekhar & Chawla, 2003)

1. Upit: Prikazati sve gradove iz tabele Grad i države kojima pripadaju.

   SELECT Grad.Ime, Grad.Država

   FROM Grad

Ovaj SQL izraz je jednak operaciji *project* iz relacijske algebre. Izraz WHERE nedostaje u upitu jer nisu specificirani uslovi nego su prikazani svi gradovi (dakle, nema nikakvih uslova nego su jednostavno priakzani svi gradovi iz tabele Grad).

Ime	Država
Havana	Kuba
Vošington	SAD
Monterej	Meksiko
Toronto	Kanada
Brazilija	Brazil
Rosario	Argentina
Otava	Kanada
Meksiko Siti	Meksiko
Buenos Aires	Argentina

Tabela 8: Rezultat 1. Upita

2. Upit: Prikazati imena glavnih gradova sa svim njihovim podacima iz tabele Grad.

   SELECT *

   FROM Grad

   WHERE Glavni_grad="D"

Ovaj SQL izraz je jednak operaciji *select* iz RA. Nažalost u SQL-u operacija *select* je specificirana u WHERE, a ne u SELECT klauzuli. Znak * u SELECT dijelu znači da se moraju prikazati svi atributi iz tabele Grad.

Ime	Država	Populacija (miliona)	Glavni grad	Oblik
Havana	Kuba	2.1	D	Pointid-1
Vošington	SAD	3.2	D	Pointid-2
Monterej	Meksiko	2.0	N	Pointid-3
Toronto	Kanada	3.4	N	Pointid-4
Brazilija	Brazil	1.5	D	Pointid-5

Rosario	Argentina	1.1	N	Pointid-6
Otava	Kanada	0.8	D	Pointid-7
Meksiko Siti	Meksiko	14.1	D	Pointid-8
Buenos Aires	Argentina	10.75	D	Pointid-9

Tabela 9: Rezultat 2. upita

3. Upit: Pronaći imena država u kojima je očekivani životni vijek kraći od 70 godina!

```
SELECT  Država.Ime
FROM    Država
WHERE   Država.Živ_vijek < 70
```

Ako poredimo se relacionom algebrom vidimo da naredba SELECT u SQL-u odgovara π u relacionoj algebri, a naredba WHERE odgovara σ. Klauza FROM definiše mjesto pretraživanja (tabelu/relaciju) što je u relacionoj algebri pisano u zagradama.

4. Upit: Prikazati glavne gradove i populaciju država čiji BDP prelazi bilion dolara!

```
SELECT  Grad.Ime, Država.Populacija
FROM    Grad, Država
WHERE   Grad.Država=Država.Ime AND
        Država.BDP>1000 AND
        Glavni_grad = "D"
```

Ovo je implicitan način za prikaz operacije spajanja. SQL2 i SQL3 također podržavaju i eksplicitnu operaciju spajanja. U ovome slučaju tabele Grad i Država imaju zajedničke atribute, to su imena država i to: u tabeli Grad atribut Država, a u tabeli Država atribut Ime. U nastavku upita dva selekciona uslova su zasebno specificirana na tabelama Grad i Država. Korištenjem notacije „ime tabele-tačka-atribut“ je izbjegnuta bilo kakva zabuna koja se mogla desiti zbog imena atributa u tabelama.

5. Upit: Pronaći ime i populaciju glavnog grada države u kojoj izvire rijeka St.Lovrens!

```
SELECT  Grad.Ime,Grad.Populacija
FROM    Grad, Država, Rijeka
WHERE   Rijeka.Izvor=Država.Ime AND
```

```
Država.Ime=Grad.Država AND
Rijeka.Ime="St.Lovrens" AND
Grad.Glavni_grad="D"
```

Ovaj upit zahtjeva spajanje tri tabele. Tabele Rijeka i Država su spojene preko atributa Izvor i Ime (oba se odnose na ime države), a tabele Država i Grad su spojene atributa Ime i Država (ponovo, oba se odnose na ime države). Postoje dva selekciona uslova koja su izvršena na tabelama Rijeka i Grad.

6. Upit: Kolika je prosječna populacija neglavnih gradova prikazanih u tabeli Grad?

```
SELECT  AVG(Grad.Populacija)
FROM    Grad
WHERE   Glavni_grad="N"
```

Operacija AVG (eng. average – prosjek) je primjer agregatne operacije. Ove operacije nisu dostupne u relacionoj algebri. Osim ove agregatne operacije dostupne su i neke druge, kao što su: COUNT (eng. count – brojati), MAX (eng. maximum), MIN (eng. minimum) i SUM (eng.sum – suma,zbir). Agregatne operacije proširuju funkcionalnost SQL-a jer dopuštaju vršenje proračuna na povratnim podacima.

7. Upit: Za svaki kontinent pronađi prosječan BDP.

```
SELECT      Država.Kontinent, Avg (Država.BDP)  AS  Kontinent BDP
FROM        Država
GROUP BY    Kontinent
```

U ovome upitu vidimo glavno odstupanje od osnovnog SQL formata upita zbog pojavljivanja klauze GROUP BY. Ova klauza dijeli tabelu na osnovu atributa navedenih u klauzuli. U ovome slučaju postoje dvije vrijednosti za Država.Kontinent, to su SAM i JAM. Prema tome tabela DRŽAVA je podijeljena u dvije grupe. Za svaku grupu je sračunat prosječan BDP i zatim je pohranjen pod atributom Kontinent-BDP kao što je specificirano u SELECT klauzuli.

8. Upit: Za svaku državu u kojoj izviru barem dvije rijeke pronaći dužinu manje rijeke.

```
SELECT    Rijeka.Izvor, MIN(Rijeka.Dužina)  AS  MIN-Dužina
FROM      Rijeka
GROUP BY  Rijeka.Izvor
HAVING    COUNT(*) > 1
```

U ovome upitu imamo HAVING kaluzu koja omogućava da uslovi odabira budu primijenjeni na različitim grupama formiranim u klauzi GROUP BY. U primjeru se uzimaju u obzir samo grupe koja imaju više od jednog člana.

9. Upit: Prikaži države čiji je BDP veći od BDP-a Kanade.

```
SELECT    Država.Ime
FROM      Država
WHERE     Država.BDP    >ANY    (SELECT    Država1.BDP
                                 FROM       Država1
                                 WHERE      Država1.Ime="Kanada")
```

Ovo je primjer ugnježdenog upita, tj. upita koji ima druge upite sadržane u sebi. Ugnježdeni upit postaje obavezan kada je potrebna međutabela, koja ne postoji, da bi se mogao izvršiti upit. Ugnježdeni upit se obično pojavljuje u WHERE klauzi, iako se rjeđe može pojaviti i u SELECT i FROM klauzi. ANY je skupni operator za poređenje.

3.5 Prostorni SQL upiti

Iako su jako dobri alati za procesiranje upita, RA i SQL imaju svoje nedostatke. Glavni nedostatak je da vrše upite samo na jednostavnim podacima kao što su brojevi, datumi i slova. Aplikacije prostornih baza moraju biti sposobne rukovati kompleksnim tipovima podataka kao što su tačke, linije i poligoni. Prodavači baza podataka su odgovorili na ovu potrebu na dva načina i to pomoću „blobova[4]" (eng. *blobs*) i hibridnih sistema u kojima su prostorni atributi pohranjeni u fajlove operativnog sistema preko GIS-a. SQL ne može procesirati

[4] Blob (eng. *blob- **b**inary **l**arge **ob**ject*) - odnosi se na veliki broj bita koje treba pohraniti u bazi podataka kao što su slike ili muzički fajlovi. Najvažnije obilježje bloba je to što ga sama baza ne može interpretirati.

podatke koji su pohranjeni kao blobovi tako da je odgovornost aplikacijske tehnike da rukuje podacima u obliku bloba (Stonebraker i Moore, 1997). U hibridnim sistemima prostorni atributi su pohranjeni u odvojene fajlove operativnog sistema i stoga nisu u mogućnosti da iskoriste prednost tradicionalnih usluga baza podataka kao što su upitni jezik, kontrola konkurentnosti i indeksiranje. (Shekhar & Chawla, 2003)

Objektnoorijentisani sistemi su imali najveći uticaj na proširenje mogućnosti SUBP-a za podržavanje prostornih (kompleksnih) objekata. Program proširenja relacijskih baza podataka sa objektno orijentiranim osobinama pada pod okvir OR-SUBP-a. Ključno obilježje OR-SUBP-a je da podržava verziju SQL3/SQL99 koja podržava ideju korisnički definiranih tipova (kao u Javi ili C++).

Glavni zahtjev prostornog SQL-a je da obezbijedi veću apstrakciju prostornih podataka ugrađivanjem koncepata koji su bliži našoj percepciji prostora (Egenhofer, 1994). Ovo je ostvareno ugrađivanjem objektnoorijentisanog koncepta korisnički definiranih apstraknih tipova podataka. Apstraktni tip podataka je korisnički definiran tip sa svojim povezanim funkcijama. Npr., ako smo *zemljišne parcele* pohranili kao poligone u bazi podataka tada koristan ATP može biti kombinacija tipa *poligon* sa nekim povezanim funkcijama (metodama) kao što je *susjedstvo*. Funkcija *susjedstvo* se može primjeniti na *parcele* da odredi da li neke parcele imaju zajedničke granice. Termin apstrakno se koristi jer korisnik ne treba znati detalje implementacije povezanih funkcija. Sve što krajnji korisnici trebaju znati je iterfejs, tj. trebaju znati dostupne funkcije i tipove podataka za unos parametara i izlaz rezultata.

3.5.1 OGIS standard za prošireni SQL

OGIS (eng. *Open Geodata Interoperability Specification*) konzorcij je oformljen od strane najvećih proizvođača GIS softvera radi formuliranja industrijskog standarda vezanog za GIS interoperabilnost. OGIS prostorni model podataka se može ugraditi u različite programske jezike, npr., C, Java, SQL, itd.

Model podataka se sastoji od osnovne klase GEOMETRIJA (eng. *GEOMETRY*) koja je neinstancirana (tj. objekti se ne mogu definirati kao instance klase GEOMETRIJA), ali specificira prostorni referentni sistem koji se koristi za sve njene podklase. Četiri glavne podklase izvedene iz superklase GEOMETRIJA su: Tačka (eng. point), Kriva (eng. *curve*), Površina (eng. *surface*) i KolekcijaGeometrija (eng. *GeometryCollection*). Sa svakom klasom

je povezan skup operacija koje djeluju na instance klase. Operacije specificirane u OGIS standardu se dijele u tri kategorije:

1. Osnovne operacije koje su primjenjive na sve geometrijske tipove podataka;
2. Operacije testiranja topoloških veza među prostornim objektima;
3. Opšte operacije za prostornu analizu.

U nastavku su navedeni neki prostorni operatori dostupni u bazi podataka:

- **SpatialReference** - Vraća kooridnatni sistem geometrije;
- **Envelope** - Vraća minimalni ortogonalni obuhvatni poligon geometrije;
- **Export** - Vraća geometriju u drugačijem prikazu;
- **IsEmpty** - Vraća „tačno“ ako je geometrija prazan skup;
- **IsSimple** - Vraća „tačno“ ako je geometrija jednostavna (nema samopresjecanja);
- **Boundary** - Vraća granicu geometrije;
- **Equal** - Vraća „tačno“ ako su unutrašnjost i granica dvije geometrije prostorno jednaki;
- **Disjoint** - Vraća „tačno“ ako se granice i unutrašnjost dvije geometrije ne presjecaju;
- **Intersect** - Vraća „tačno“ ako se dvije geometrije presjecaju;
- **Cross** - Vraća „tačno“ ako se unutrašnjost površine presjeca sa krivom;
- **Within** - Vraća „tačno“ ako se unutrašnjost date geometrije ne presjeca sa vanjštinom druge geometrije;
- **Contains** - Testira da li data geometrija sadrži neku drugu datu geometriju;
- **Inside** - Suprotno od ***Contains***, *A* ***Inside*** *B* implicira *B* ***Contains*** *A*;
- **Overlap** - Vraća „tačno“ ako unutrašnjosti dvije geometrije imaju neprazan presjek;
- **Distance** - Vraća najkraću udaljenost između dvije geometrije;
- **WithinDistance** - Testira da li su geometrije na specificiranoj udaljenosti;
- **Buffer** - Vraća geometriju koja sadrži sve tačke oko date geometrije koje su na udaljenosti manjoj ili jednakoj od zadate udaljenosti;
- **ConvexHull** - Vraća najmanji konveksni geometrijski skup koji okružuje geometriju;
- **Intersection** - Vraća geometrijski presjek dvije geometrije;
- **Union** - Vraća geometrijsku uniju dvije geometrije;
- **Difference** - Vraća dio geometrije koji se ne siječe sa drugom datom geometrijom;
- **SymDiff** - Vraća dijelove dvije geometrije koje se međusobno ne sijeku;
- **Touch** - Vraća „tačno“ ako se granice ali ne i unutrašnjosti datih geometrija sijeku;
- **Anyinteract** - Vraća „tačno“ ako geometrije nisu razdvojene;

- **Covers** - Vraća „tačno“ ako je unutrašnjost jedne geometrije u potpunosti sadržana u unutrašnjosti ili granici druge geometrije i ako se njihove granice sijeku;
- **CoveredBy** - Suprotno od Covers, *A **CoveredBy** B* implicira *B **Covers** A*;
- **On** - Vraća „tačno“ ako su unutrašnjost i granica jedne geometrije na granici druge geometrije i druga geometrija pokriva prvu geometriju.

OGIS specifikacije su ograničene objektnim modelom prostora. OGIS razvija saglasne modele za operacije i tipove podataka u modelu polja. Čak i sa objektnim modelom OGIS operacije su ograničene na jednostavne SELECT-PROJECT-JOIN upite. Podrška za prostorne agregatne upite koji sadrže klauze GROUP BY i HAVING i dalje zadaju dosta problema. Fokus u OGIS standardu je na osnovnim topološkim i metričkim operacijama dok su npr. one koje se odnose na pravac (gore, dole, sjeverno, istočno i sl.) potpuno izostavljene. Također nisu podržane dinamičke operacije (npr.: reproduce, trasnlate i sl.), operacije bazirane na obliku (npr. ne možemo dobiti odgovor na upit „koji poligoni su kvadratnog oblika?“) i operacije vidljivosti (npr. ne možemo dobiti odgovor na upit „sa kojih tačaka se vidi poligonska tačka 57?“).

U relacijskim bazama podataka skup tipova podataka je fiksiran. U objektno-relacijskim i objektnim bazama podataka ne postoji ovakvo ograničenje jer je omogućeno pravljenje korisnički definiranih tipova podataka. Ova mogućnost daje veliku prednost prilikom netradicionalnih primjena baza podataka kao što je GIS. Sada se teret pravljenja sintatički i semantički ispravnih tipova podataka prebacuje na one koji razvijaju baze podataka. SQL3/SQL99 predloženi SQL standard za OR-SUBP dozvoljava korisnički definirane tipove podataka unutar okvira relacijskih baza podataka. U nastavku su opisane dvije osobine SQL3 standarda koje bi mogle biti korisne prilikom definiranja korsinički definiranih prostornih tipova podataka.

1. **Apstraktni tip podataka** se može kreirati pomoću CREATE TYPE naredbe. Kao klase u objektno orijentiranoj tehnologiji, ATP se sastoji od atributa i pripadajućih funkcija za pristup vrijednostima atributa. Pripadne funkcije mogu promijeniti vrijednosti atributa unutar tipa podataka što može dovesti do promjene stanja baze podataka. ATP se može pojaviti kao tip kolona u relacijskoj šemi. Da bi se pristupilo vrijednosti koji ATP učahuri mora se koristiti pripadna funkcija specificirana u CREATE TYPE. U nastavku je dat primjer kreiranja tipa *Point* sa definicjom pripadne funkcije *Distance*:

```
CREATE TYPE   Point   (
                  x   NUMBER,
```

```
        y   NUMBER,
FUNCTION    Distance (: u Point, : v Point)
            RETURNS NUMBER
                );
```

Dvotačke ispred *u* i *v* označavaju da su ovo lokalne varijable.

2. **Tip red** je tip za relaciju. Tip red specificira shemu relacije. Sljedeća naredba kreira red tip *Point*.

```
CREATE ROW TYPE   Point      (
                      x   NUMBER,
                      y   NUMBER  );
```

Sada možemo kreirati tabelu koja instancira red tip. Npr.:

CREATE TABLE Pointtable of TYPE Point.

U nastavku će biti prikazano kako se prave tri osnovna prostorna tipa podataka (tačka, linija i poligon) u objektno relacijskom SUBP-u Oracle8.

Tačka

```
CREATE   TYPE Point AS OBJECT   (
             x   NUMBER,
             y   NUMBER,
             MEMBER FUNCTION Distance (P2   IN  Point)   RETURN NUMBER,
             PRAGMA RESTRICT_REFERENCES (Distance, WNDS));
```

Tip Tačka (eng. *Point*) ima dva atributa, x i y, i jednu pripadajuću funkciju, Udaljenost (eng. *Distance*). PRAGMA ukazuje na činjenicu da funkcija *Distance* neće mijenjati stanje baze podataka: WNDS (eng. *Write No Database State* - ne zapisuj stanje baze). U OGIS standardu su za tip Tačka specificirane mnoge operacije ali radi jendostavnosti ovdje je prikazana samo jedna. Nakon što je napravljen tip Tačka može se koristiti u relaciji kao atribut. Npr., šema relacije Grad se može definirati na sljedeći način:

```
CREATE   TABLE      Grad   (
             Ime          varchar(30),
             Država       varchar(30),
             Populacija   Integer,
             Gl_grad      Char(1),
```

```
        Oblik       Point   );
```

Kada je relacijska shema definirana možemo popuniti tabelu na uobičajen način. Npr., sljedeća naredba dodaje informacije u bazu podataka vezane za Braziliju, glavni grad Brazila.
INSERT INTO Grad („Brazilija“, „Brazil“, 1.5, „DA“, Tačka(-55.4,-23.2));

Linija

CREATE TYPE LineString AS VARRAY(500) OF Point;
LineType je niz varijabli tipa Point maksimalne dužine 500. Specifične pripadne funkcije se ne mogu definirati ako je tip definiran kao Varray. Zbog toga kreiramo drugi tip LineString.

```
CREATE  TYPE LineString AS OBJECT   (
        Num_of_Points INT,
        Geometry LineType,
        MEMBER FUNCTION Lenght (SELF IN) RETURN NUMBER,
        PRAGMA RESTRICT_REFERENCES (Lenght. WNDS));
```

Atribut Num_of_Points pohranjuje veličinu (u smislu broja tačaka) svake instance tipa LineString. Sada se može definirati shema tabele Rijeka.

```
CREATE  TABLE   Rijeka (
        Ime     varchar(30),
        Izvor   varchar(30),
        Dužina  Number,
        Oblik   LineString  );
```

Prilikom unosa podataka u tabelu Rijeka treba paziti na tipove podataka koji se unose.
INSERT INTO Rijeka („Misisipi“, „SAD“, 6000, LineString (3, LineType (Point (1,1), Point (1,2), Point (2,3)))

Poligon

Prethodno su objašnjena značenja pojedinih naredbi pri kreiranju apstraktnih tipova podataka, relacijskih shema i punjenja tabela podacima, tako da će sada biti dat samo redoslijed kreiranja tabele Država.

CREATE TYPE PolyType AS VARRAY (500) OF Point

```
CREATE   TYPE Polygon AS OBJECT   (
         Num_of_Points    INT,

    CREATE   TABLE        Država (
             Ime          Varchar (30),
             Kontinent    Varchar (30),
             Populacija   Integer,
             BDP          Number,
             Životni_vijek Number,
             Oblik        PolyType   );
         Geometry   PolyType,
         MEMBER FUNCTION Area (SELF IN) RETURN NUMBER,
         PRAGMA RESTRICT_REFERENCES (Lenght, WNDS));
```

INSERT INTO Država („Meksiko“, „JAM“, 107.5, 713.8, 69.36, Polygon (23, PolyType (Point (1,1), ... , Point (1,1)))

3.5.2 Primjeri prostornih SQL upita

Korištenjem OGIS tipova podataka i operacija u nastavku će biti navedeni primjeri SQL upita u bazi Svijet koji ističu prostorne veze između entiteta: Država, Grad i Rijeka. Prije samih upita redefinirat ćemo relacijsku šemu kao što je prikazano u tabelama ispod:

```
CREATE   TABLE     Država (          CREATE   TABLE   Rijeka (
         Ime       varchar (30),               Ime     varchar (30),
         Kontinent varchar (30),               Izvor   varchar (30),
         Populacija Integer,                   Dužina  Number,
         BDP       Number,                     Oblik   LineString);
         Oblik     Polygon);

                   CREATE   TABLE     Grad (
```

Ime	varchar (30),
Država	varchar (30),
Populacija	Integer,
Oblik	Point);

Tabela 10: Izmijenjene tabele baze Svijet

1. Upit: Pronaći imena svih država u tabeli Država koje su susjedi SAD-a.

```
SELECT   Država1.Ime AS „Susjed SAD-a“
FROM     Država1, Država2
WHERE    Touch (Država1.Oblik, Država2.Oblik) = 1 AND
         Država2.Ime = “SAD“
```

Predikat *Touch* provjerava da li su dva geometrijska objekta susjedna bez preklapanja. Ovo je korisna operacija za određivanje susjedstva geometrijskih objekata. Operacija *Touch* jedna od topoloških operacija specificiranih u OGIS standardu. Jako korisna osobina topoloških operacija je to što su nepromjenljive bez obzira na mnoge geometrijske transformacije (npr. odabir koordinatnog sistema za bazu Svijet neće uticati na rezultat topoloških transformacija).

2. Upit: Pronaći imena država kroz kojih prolazi rijeka.

```
SELECT   Država.Ime
FROM     Rijeka, Država
WHERE    Cross (Rijeka.Oblik, Država.Oblik)=1
```

Cross je topološki predikat koji se najčešće koristi za provjeru presjeka između linijskih objekata ili između linijskih i poligonalnih objekata (kao u navedenom primjeru).

3. Upit: Za svaku rijeku pronaći najbliži grad!

```
SELECT   Grad1.Ime, Rijeka1.Ime
FROM     Grad1, Rijeka1
WHERE    Distance (G1.Oblik, R1.Oblik) <
                                    ALL     (SELECT Distance(G2.Oblik,R1.Oblik)
                                    FROM    Grad2
                                    WHERE   Grad1.Ime<>Grad2.Ime )
```

Distance je binarna operacija čiji je rezultat realna vrijednost. U prethodnom primjeru jednom je korištena u WHERE klauzuli i zatim ponovo u SELECT klauzuli podupita. Distance funkcija je definirana za bilo koju kombinaciju geometrijskih objekata.

4. Upit: Rijeka St. Lovrens može vodom snadbijevati gradove u području od 300km oko rijeke. Prikazati gradove koji mogu koristiti vodu iz rijeke.

```
SELECT   Grad.Ime
FROM     Grad, Rijeka
WHERE    Overlap (Grad.Oblik, Buffer (Rijeka.Oblik, 300)) = 1 AND
                  Rijeka.Ime = „St. Lovrens“
```

Bafer (eng. *Buffer*) geometrijskog objekta je geometrijska površina centrirana na objektu koja ima veličinu određenu parametrom u operaciji Bafer. Ova operacija se često koristi u mnogim GIS aplikacijama uključujući nadziranje poplava i zoniranje u urbanim u ruralnim područjima.

5. Upit: Prikazati dužinu rijeka u svakoj državi kroz koju rijeka prolazi.

```
SELECT   Rijeka.Ime, Država.Ime, Lenght (Intersection (Rijeka.Oblik,
         Država.Oblik))
FROM     Rijeka, Država
WHERE    Cross (Rijeka.Oblik, Država.Oblik) = 1
```

Operacija Intersection se razlikuje od funkcije Intersects koja je topološki predikat koji određuje da li se dvije geometrije sijeku. Intersection tipova LineString i Polygon tipova podataka može biti Point ili LineString tip. Ako rijeka prolazi kroz neku državu tada će

rezultat biti LineString i tada će funkcija vratiti vrijednost dužine koja je različita od nule za svaku državu kroz koju rijeka prolazi.

6. Prikazati BDP i udaljenost glavnog grada države od ekvatora za sve države.

```
SELECT   Država.BDP, Distance (Point (0, Grad.Oblik.y), Grad.Oblik)
FROM     Država, Grad
WHERE    Država.Ime = Grad.Država AND
         Grad.Gl_Grad = „DA“
```

Traženje implicitne veze između skupova podataka pohranjenih u bazi podataka je van dohvata standardne funkcionalnosti baze podataka. Trenutni SUBP-ovi su usmjereni prema on-line procesiranju transakcija (eng. *on-line transaction processing - OLTP*), dok je ovaj upit u oblasti on-line analitičkog procesiranja (eng. *on-line analytical processing - OLAP*). Sam OLAP pada u oblast „rudarenja podataka“ (eng. *data mining*). U ovome trenutku najbolje što možemo je prikazati svaki glavni grad i njegovun udaljenost od ekvatora.

Grad.Ime	Država.BDP	Udaljenost od ekvatora (km)
Havana	16.9	2562
Vošington	8003	4324
Brazilija	1004	1756
Otava	658	5005
Meksiko Siti	694.3	2161
Buenos Aires	348.2	3854

Tabela 11: Rezultat upita br. 6

7. Upit: Prikazati države koje imaju samo jednog susjeda. Država je susjed drugoj državi ako države dijele granicu. Prema ovoj definiciji ostrvske države kao što je Island nemaju susjede.

```
SELECT      Država.Ime
FROM        Država, Država1
WHERE       Touch (Država.Oblik, Država1.Oblik)=1
GROUP BY    Država.Ime
HAVING      Count (Država1.Ime) = 1
```

Ponovo imamo GROUP BY i HAVING klauze. GROUP BY klauza dijeli tabelu na osnovu imena država. HAVING klauza prisiljava selekciju da suzi izbor samo na one države koje imaju jednog susjeda. HAVING klauza ima sličnu ulogu klauzi WHERE, razlika je u tome što mora imati agregatne funkcije kao što su *count, sum, max* i *min*.

8. Koja država ima maksimalan broj susjeda?

```
CREATE WIEW Susjed AS
SELECT      Država.Ime, Count (Država1.Ime) AS br_susjeda
FROM        Država, Država1
WHERE       Touch (Država.Oblik, Država1.Oblik)
GROUP BY    Država.Ime

SELECT  Država.Ime, br_susjeda
FROM    Susjed
WHERE   br_susjeda = (SELECT Max (br_susjeda)
                      FROM Susjed)
```

Ovaj upit pokazuje korištenje pogleda radi pojednostavljenja kompleksnih upita. Prvi upit (pogled - eng. *view*) računa broj susjeda za svaku državu. Ovaj pogled pravi virtuelnu tabelu koja se može koristiti kao normalna tabela u uzastopnim upitima. Drugi upit selektuje države sa najvećim brojem susjeda iz pogleda Susjed.

Primjeri za bazu podataka „Katastar nekretnina“

Neka su kreirane tabele „parcela“ i „zgrada“ na sljedeći način:

```
CREATE TABLE Parcela (
Broj Varchar (10),
Posjednik Varchar (40),
Povrsina Double,
Oblik   PolyType);

CREATE TABLE Zgrada {
Adresa   Varchar (40) NOT NULL,
Grad   Varchar (30),
Broj_stanara   INT (3),
Spratnost   INT (2),
Geometrija   Polygon }
```

1. Izračunati površinu svih parcela koje sadrže zgrade!

```
SELECT Sum(Parcela.Povrsina)
FROM Parcela, Zgrada
WHERE WITHIN (Zgrada.Geometrija , Parcela.Geometrija)=TRUE
```

2. Pronaći sve susjede parcele 256!

```
SELECT Parcela.Broj
FROM Parcela, Parcela1
WHERE TOUCH (Parcela.Geometrija , Parcela1.Geometrija)=TRUE AND
Parcela1.broj = „256“
```

3. Pronaći udaljenost između parcela i zgrada i sortirati rezultate prema broju parcele!

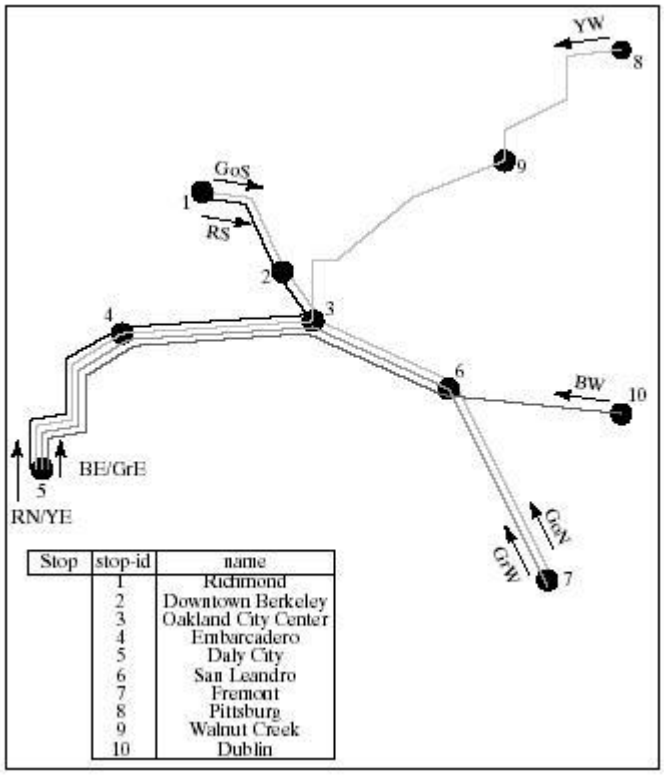
```
SELECT Parcela.broj, KAT_Zgrada.broj, Distance(Parcela.Geometrija, Zgrada.Geometrija),
FROM Parcela, Zgrada
ORDER BY KAT_Parcela.broj
```

4. Pronaći sve parcele koje se nalaze u krugu od 100m od puta M-16.

```
SELECT        Parcela.Broj
FROM Parcela, Put
WHERE         Overlap (Parcela.Oblik, Buffer (Put.Oblik, 100)) = 1 AND
Put.Naziv = „M-16“
```

4. TEORIJA GRAFOVA

Veliki broj prostornih problema se može prikazati pomoću prostornih mreža. Npr. sistem puteva ili željezničkih veza je često prikazan u obliku mreže. Korištenjem prostornih mreža se često rješava problem najkraće rute (npr. pomoću Dijkstra algoritma).



Stop	stop-id	name
	1	Richmond
	2	Downtown Berkeley
	3	Oakland City Center
	4	Embarcadero
	5	Daly City
	6	San Leandro
	7	Fremont
	8	Pittsburg
	9	Walnut Creek
	10	Dublin

Slika 1: Željeznička mreža u San Francisku(Shekhar & Chawla, 2003)

Prostorne mreže imaju veliku primjenu u oblastima navigacije i transporta. Npr. pomoću prostornih mreža omogućene su sljedeće usluge:

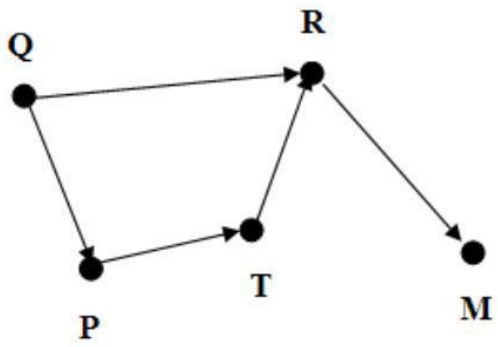
- Vođenje do mjesta na osnovu kućne adrese;
- Računanje najkraćeg puta do odredišta;
- Praćenje odabrane rute.

Teroija grafova predstavlja osnovu mrežnog modela pa će u nastavku biti govora o osnovnim konceptima grafova.

4.1 Grafovi

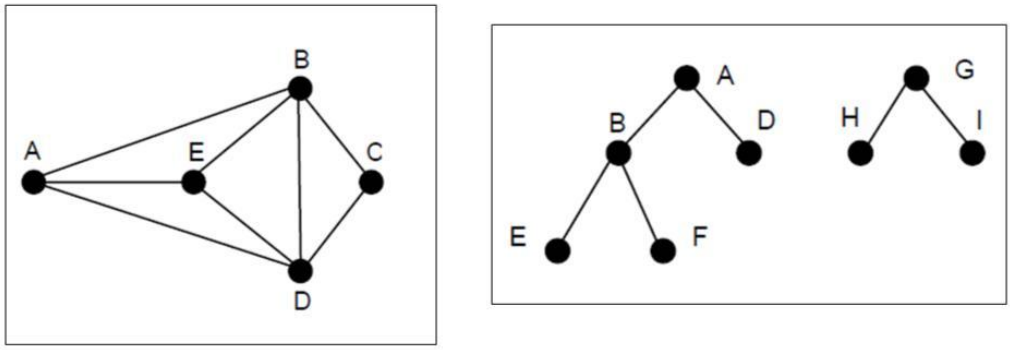
Graf, $G = (V, E)$ se sastoji od konačnog skupa vrhova V i skupa rubova E između vrhova u V. Tako je skup rubova E u binarnoj relaciji sa V. *Graf* je **usmjeren** ako je uređen par vrhova koji predstavljaju rub, inače graf je **neusmjeren**. Vrhovi i rubovi nazivaju i čvorovi i veze. Prvi element uređenog para vrhova se ponekad naziva prethodnik ili izvor, a drugi sljedbenik ili odredište.

Ponekad su **oznake** (*label*) i **težine** (*weight*) vežu za čvorove i veze grafa, za dodavanje dodatnih informacija. Npr. ime, geografske koordinate ili oboje može biti priloženo čvorovima grafa, a udaljenosti između čvorova mogu biti odabrani kao težine na rubovima.



Slika 2: Usmjereni graf sa oznakama

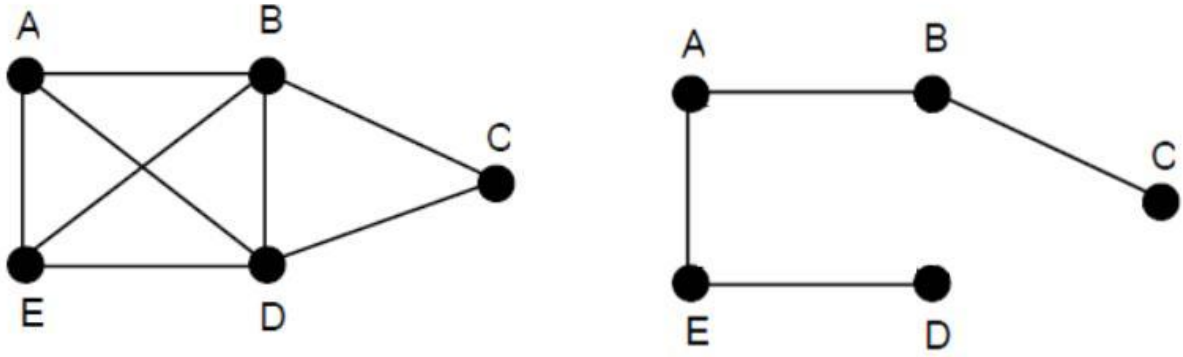
Dva ruba su susjedna ako imaju zajednički čvor. Redoslijed susjednih rubova predstavlja putanju (*path*). Na primjer, sekvenca (v_0, v_1), (v_1, v_2),..., (v_{n-2}, v_{n-1}), (v_{n-1}, v_n) predstavlja putanju jer svaki rub imazajednički čvor s prethodnim ili narednimrubom. Ako postoji putanja između bilo koja dva čvora u grafu tada za graf kažemo da je „**povezan**“, u suprotnom je „**nepovezan**“.



Slika 3: Povezani (lijevo) i nepovezani (desno) grafovi

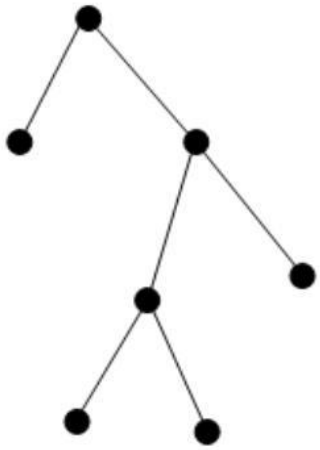
Ako su krajnje tačke v_0 i v_1 jednake, takva putanja se naziva **ciklus**. Ciklus u kome se svaki čvor posjećuje samo jednom se naziva **Hamiltonovom kružnom putanjom**. Ako

graf nema ciklusa onda je graf **acikličan**. Npr. u riječnoj mreži nema ciklusa (aciklični graf), ali povratno putovanje u željezničkom sistemu predstavlja ciklus.



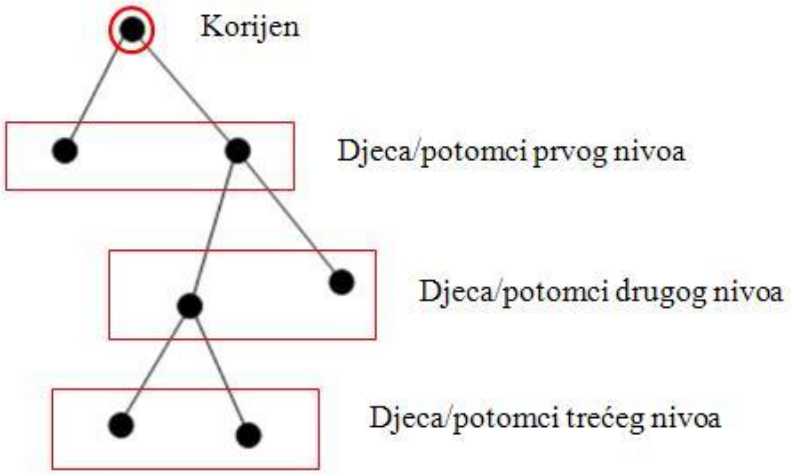
Slika 4: Cikličan (lijevo) i acikličan (desno) graf

Posebno korisnu klasu grafova predstavlja stablo. Stablo je povezani aciklični graf.



Slika 5: Aciklični povezani graf – stablo

Ukorijenjeno stablo je stablo kod koga jedan od njegovih čvorova predstavlja korijen, po čemu se i razlikuje od ostalih čvorova. Ukorijenjena stabla se po dogovoru crtaju sa korijenom na vrhu dok se ostali čvorovi crtaju ispod – na nižim nivoima. Čvorovi koji su najbliži korjenu nazivaju se **djecom/potomcima prvog nivoa**. Ovi čvorovi mogu i sami imati svoje potomke koji su za njih potomci prvog nivoa, ali su u isto vrijeme i potomci drugog nivoa za korijen. Ako čvor nema potomka naziva se **listom**.



Slika 6: Ukorijenjeno stablo

4.2 Metode predstave grafa

Graf se može predstaviti na više načina. Najčešće graf predstavljamo **grafički** kao što je prikazano na slici 7a.

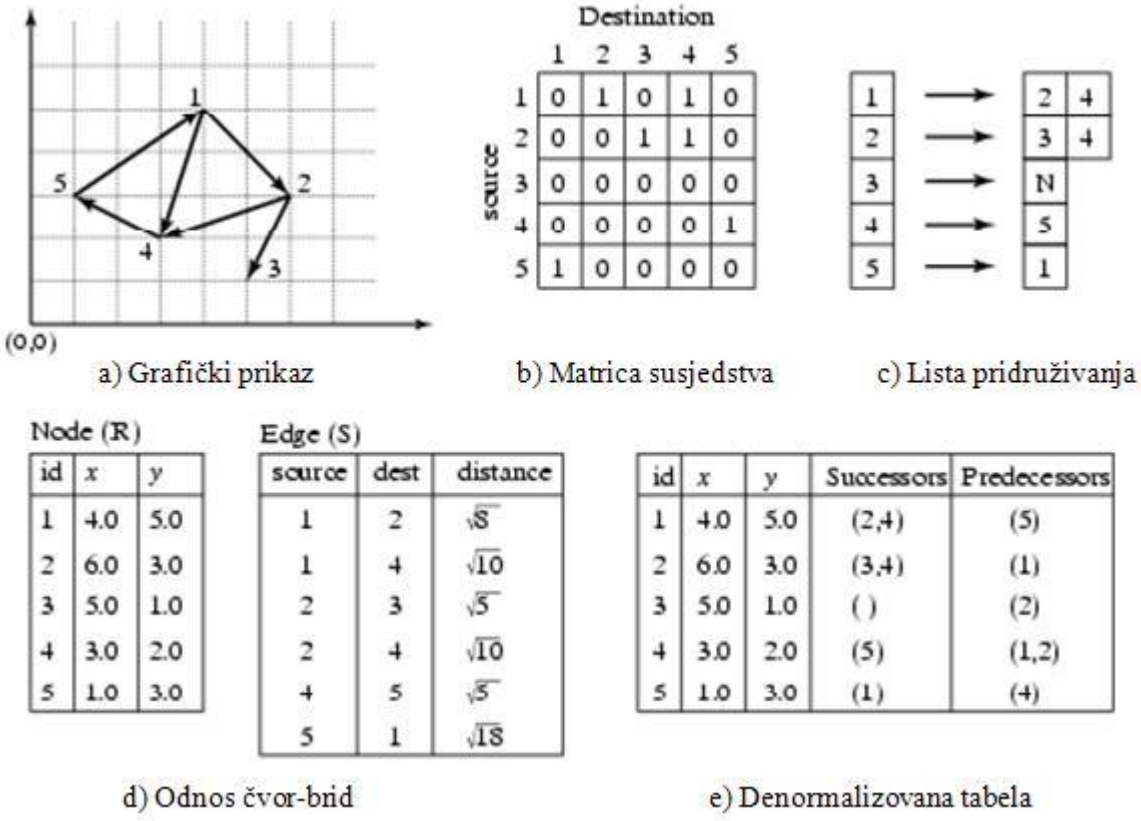
Matrica susjedstva (7b) i **lista pridruživanja** (7c) su također dvije dobro poznate metode za prikaz grafova. U matrici susjedstva redovi i kolone grafa predstavljaju vrhove na grafu. Matrični unos može biti ili 1 ili 0, ovisno o tome da li postoji rub između dva vrha, kao što je prikazano na slici 7b. Ako je graf neusmjeren, onda je rezultirajuća matrica simetrična. Struktura matrice susjedstva je efikasna za odgovaranje na upite o rubu.

Struktura liste pridruživanja je efikasna za upite koji uključuju nabrajanje vrhova grafa, npr. pronaći sve susjede od *v*. Struktura podataka liste pridruživanja je niz pokazivača. Svaki element polja odgovara vrhu grafa, a pokazivač pokazuje na listu neposrednog nasljednika vrha kao što je prikazano na slici 7c.

Usmjereni grafovi mogu biti implementirani u relacioni model pomoću para relacija R i S za čvorove i rubove grafa. Relacija čvora R i relacija ruba S su prikazane na slici 7d. Primijetite

da se rubovi iz relacije S mogu fizički grupisati na osnovu vrijednosti prvog atributa. Također smo uključili koordinate tačaka čvorova u relaciji R i udaljenost između čvorova (težina) u relaciji S.

Denormalizirana tabela je data na slici 7e i često se koristi da ubrza računanje najkraćeg puta. Ovaj prikaz tabele čvorova sadrži koordinate, popis nasljednika i popis prethodnika. Mogući su i drugi tablični prikazi grafova. Na primjer, uz tabelu čvorova, može se koristiti tabela ruta za čuvanje relacija između čvorova i ruta koje nastaju u domeni primjene javnog prijevoza.



id	x	y
1	4.0	5.0
2	6.0	3.0
3	5.0	1.0
4	3.0	2.0
5	1.0	3.0

source	dest	distance
1	2	$\sqrt{8}$
1	4	$\sqrt{10}$
2	3	$\sqrt{5}$
2	4	$\sqrt{10}$
4	5	$\sqrt{5}$
5	1	$\sqrt{18}$

id	x	y	Successors	Predecessors
1	4.0	5.0	(2,4)	(5)
2	6.0	3.0	(3,4)	(1)
3	5.0	1.0	( )	(2)
4	3.0	2.0	(5)	(1,2)
5	1.0	3.0	(1)	(4)

Slika 7: Metode predstave grafova

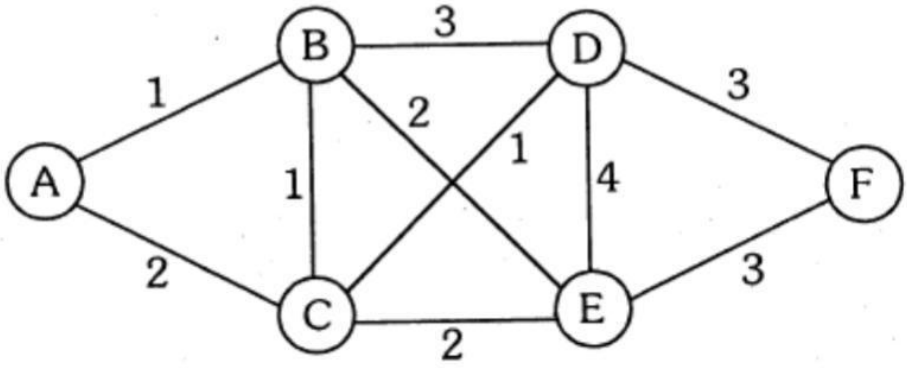
4.3 Najkraći put u grafu

4.3.1 Dijkstra algoritam

Ideja Dijkstra algoritma je gradi stablo koje se sastoji od bridova koji čine minimalne puteve

od početnog vrha do svih ostalih vrhova. U jednom koraku biramo brid koji spaja vrh koji nije u stablu sa stablom, ali pri tome za svaki vrh računamo udaljenost od početnog vrha. Od svih bridova koje u jednom koraku možemo izabrati uzimamo onaj za koji je pripadni vrh, koji nije u stablu, najmanje udaljen od početnog vrha.

Primjer: Za graf na slici ispod pronaći najkraći put od vrha A do vrha F



Prvi korak:

-Postavljamo da je udaljenost do A=0 jer je to početni čvor, a za sve ostale pišemo beskonačno.
-Iz A možemo doći do B i C. AB=1, a AC=2. Pišemo privremene vrijednosti za čvorove B(1) i C(2). Odaberemo najmanju privremenu vrijednost što je u ovom slučaju B(1) i ta vrijednost postaje konačna.
-Nakon što je završen prvi korak imamo dvije konačne vrijednosti i to: **A(0) i B(1)** i jednu privremenu *C(2)* – beskonačne ne računamo.

Drugi korak:

-Iz B možemo doći do C(1+1=2), D(1+3=4) i E(1+2=3). C već ima privremenu vrijednost 2 tako da nema promjene vrijednosti, za D upisujemo 4 jer je manje od beskonačno, za E pišemo 3 jer je manje od beskonačno. Odaberemo najmanju provremenu vrijednost što je u ovome slučaju C(2) koja postaje konačna.
-Nakon što je završen drugi korak imamo 3 konačne vrijednosti: **A(0), B(1), C(2)** i dvije privremene *D(4) i E(3)*.

Treći korak:

-Iz C možemo doći do B (meĎutim B već ima konačnu vrijednost pa nema potrebe za računanjem), D(2+1=3) i E(2+2=4). Za D pišemo novu privremenu vrijednost jer je novodobivena manja od postojeće, dakle D(3), a za E nema promjena. Sada imamo dvije privremne vrijednosti koje su jednake, D(3) i E(3), i odabiremo koju želimo kao konačnu. Mi odabiremo E(3) kao konačnu.

-Nakon završenog trećeg koraka imamo 4 konačne vrijednosti: **A(0), B(1), C(2), E(3)** i jednu privremenu *D(3).*

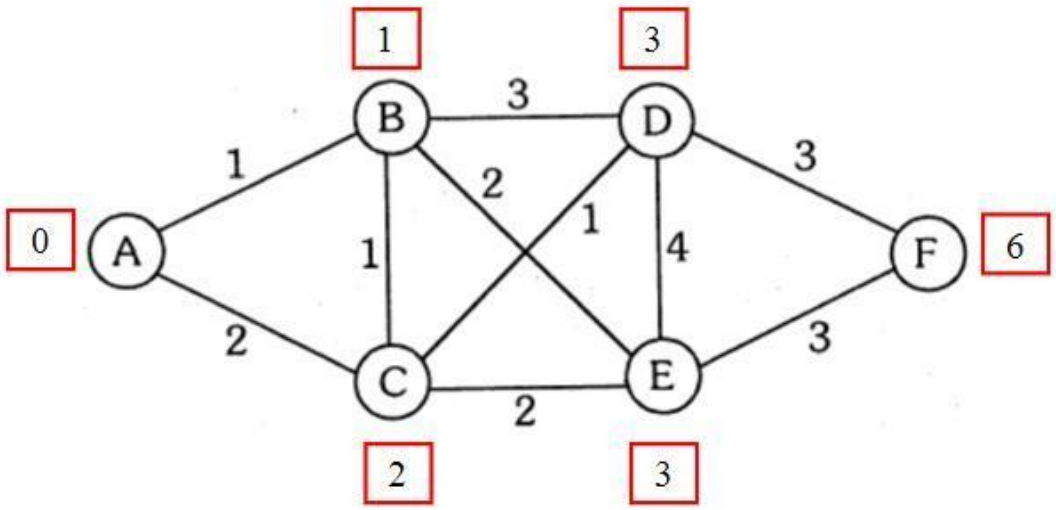
Četvrti korak:

-Iz E možemo doći do D(3+4=7) i F(3+3=6). Za D nećemo pisati novodobivenu vrijednost jer je veća od postojeće, a za F pišemo novu privremenu vrijednost F(6). -Nakon završenog četvrtog koraka imamo 5 konačnih vrijednosti: **A(0), B(1), C(2),E(3), D(3)** i jednu privremenu*F(6).*

Peti korak:

-Iz D možemo doći do F(3+3=6). Ova vrijednost je jednaka postojećoj tako da nema promjena ali možemo zaključiti da ćemo imati dvije najkraće putanje u grafu od čvora A do čvora F.

-U petom koraku smo dobili sve najkraće udaljenosti od A do ostalih čvorova u grafu, dakle dobili smo: **B(1), C(2), E(3), D(3) i F(6).**



Odredimo najkraće putanje (napomena: obično bude samo jedna putanja):

Da bismo odredili najkraću putanju idemo unazad, od čvora F prema A. Za odreĎivanje najkraćeg puta smo vršili sabiranje težina, a za odreĎivanje putanje ćemo vršiti oduzimanje. Dakle ispitujemo da li je vrijednost u F umanjena za vrijednost težine do susjednog čvora jednaka vrijednosti u odabranom čvoru. Ako jeste taj brid je dio najkraće putanje, ako nije taj brid nije dio najkraće putanje. Ovaj postupak ponavljamo dok ne dođemo do čvora A.

U našem slučaju imamo:
Prvi korak:

1. FD jer je 6-3=3
2. FE jer je 6-3=3 Drugi korak: 1.DC jer je 3-1=2 2.EB jer je 3-2=1

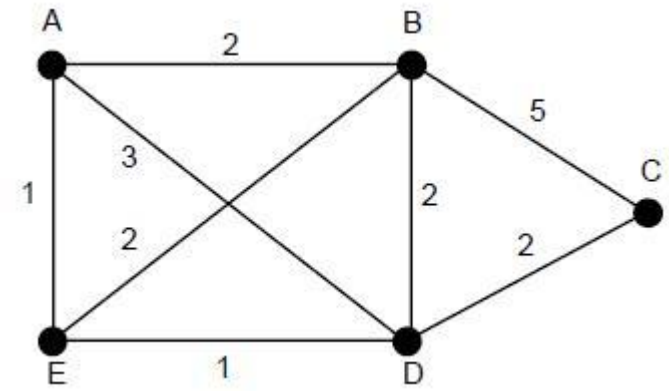
Treći korak:
1.CA jer je 2-2=0 2.BA jer je 1-1=0

Dakle najkraće putanje su:
1.ACDF
2.ABEF.

4.3.2 Floyd-ov algoritam

Floydovim algoritmom pronalazimo najmanje udaljenosti između svih parova vrhova ugrafu. Ideja algoritma je ispitivanje svih mogućih puteva u grafu, ali se pri takvom ispitivanjukoristi činjenica da ovakav problem ima optimalnu substrukturu te da se do ukupnog minimuma može doći spajanjem minimuma problema manjeg reda.

Primjer: Za graf na slici ispod pronaći najkraće putanje pomoću Floyd-ovog algoritma!



Matrica susjedstva za graf sa slike je:

	A	B	C	D	E
A	0	2		3	1
B	2	0	5	2	2
C		5	0	2	
D	3	2	2	0	1
E	1	2		1	0

U prvom koraku uzimamo vrh A za vrh kroz koji ćemo pokušati poboljšati put izmeĎu ostalih vrhova te redom zapisujemo puteve koje želimo poboljšati.

B→C – Ne možemo poboljšati jer trenutačno nemamo put od A do C B→D – Ne poboljšavamo jer je BA + AD jednak 5 a tu već imamo 2 B→E – Ne možemo poboljšati već postojeći put

C→D – Ne možemo poboljšati jer trenutačno nemamo put od C do A C→E – Ne možemo poboljšati jer trenutačno nemamo put od C do A D→E – Ne možemo pobolkjšati već postojeći put

Ovdje vidimo da prvi korak uopće nije promijenio matricu incidencije te da s vrhom A ne možemo poboljšati već postojeće puteve. Prelazimo na drugi korak s vrhom B. A→C – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 7

A→D – Ne možemo poboljšati već postojeći put A→E – Ne možemo poboljšati već postojeći put
C→D – Ne možemo poboljšati već postojeći put

C→E – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 7 D→E – Ne možemo poboljšati već postojeći put

Nova matrica susjedstva je:

	A	B	C	D	E
A	0	2	7	3	1
B	2	0	5	2	2
C	7	5	0	2	
D	3	2	2	0	1
E	1	2		1	0

U trećem koraku koristimo vrh C.

A→B – Ne možemo poboljšati već postojeći put A→D – Ne možemo poboljšati već postojeći put A→E – Ne možemo poboljšati već postojeći put B→D – Ne možemo poboljšati već postojeći put B→E – Ne možemo poboljšati već postojeći put D→E – Ne možemo poboljšati već postojeći put

Treći korak također nije ništa promijenio, a algoritam se nastavlja s vrhom D A→B – Ne možemo poboljšati već postojeći put

A→C – Udaljenost AD+DC je 5 što je manje od 7 te mjenjamo tablicu incidencije A→E – Ne možemo poboljšati već postojeći put

B→C – Udaljenost BD+DC je 4 što je manje od 5 te mjenjamo tablicu incidencije B→E – Ne možemo poboljšati već postojeći put

C→E – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 3 Matrica susjedstva sada je:

	A	B	C	D	E
A	0	2	5	3	1
B	2	0	4	2	2
C	5	4	0	2	3
D	3	2	2	0	1
E	1	2	3	1	0

U posljedenjem koraku koristimo vrh E.

A→B – Ne možemo poboljšati već postojeći put

A→C – Udaljenost AE+EC je 4 što je manje od 5 te mjenjamo tablicu incidencije A→D – Udaljenost AE+ED je 2 što je manje od 3 te mjenjamo tablicu incidencije B→C – Ne možemo poboljšati već postojeći put

B→D – Ne možemo poboljšati već postojeći put

C→D – Ne možemo poboljšati već postojeći put

Matrica susjedstva sada je:

	A	B	C	D	E
A	0	2	4	2	1
B	2	0	4	2	2
C	4	4	0	2	3
D	2	2	2	0	1
E	1	2	3	1	0

Kao što je iz matrice vidljivo sad imamo vrijednosti minimalnih udaljenosti za sve parove vrhova u grafu, meĎutim ne znamo kako se te minimalne udaljenosti ostvaruju. Pratimo li

izvoĎenje algoritma možemo rekonstruirati te puteve tako da pratimo kako smo promjenili težine u matrici incidencije u svakom koraku.

2: A→C ≡ A→B→C
C→E ≡ C→B→E

4: A→C ≡ A→D→C
B→C ≡ B→D→C
C→E ≡ C→D→E

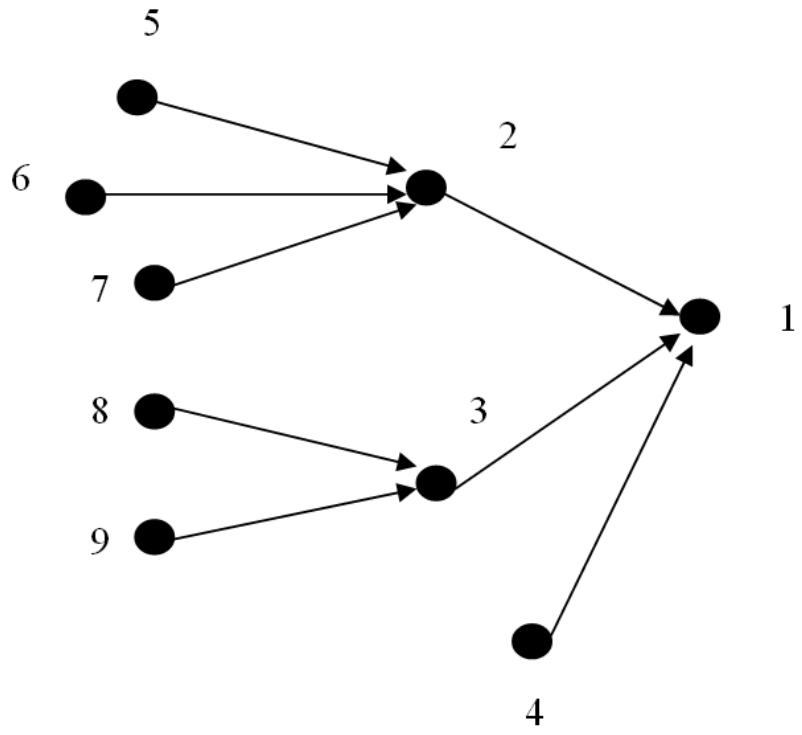
5: A→C ≡ A→E→C ≡ A→E→D→C
A→D ≡ A→E→D

Ostali putevi koji nisu navedeni ostvareni su direktnom vezom, tj. samo jednim bridom.

4.4 SQL upiti i grafovi

SQL pravila kod grafova:

- Prolaz grafom se kontroliše pomoću "START WITH" i "CONECT BY" klauzule.
- Odnos *roditelj - dijete* je definisan "CONNECT BY" klauzulom.
- Smjer traženja se upravlja korištenjem 'PRIOR' operatora.



Slika 18: Graf „Rijeke“

Operator „Prior“

SELECT Izvor

FROM Rijeke

CONNECT BY PRIOR Izvor= Odredište

START WITH Odredište= 1

Ovaj upit služi za pretraživanje grafa, a riječ PRIOR tj. mjesto na koje je postavljena određuje pravac pretraživanja - određuje hoće li se tražiti prethodnici (djeca) ili sljedbenici (roditelji) od odabranog čvora. Dakle:

CONNECT BY PRIOR source = dest traži djecu od čvora koji smo odabrali u START WITH dijelu.

CONNECT BY source = PRIOR dest traži roditelje od čvora koji smo odabrali u START WITH dijelu.

Primjer 1:

Pronaći koje rijeke će biti zagađene ako je došlo do curenja u rijeci 6?

Napomena: Ustvari treba pronaći roditelje od 6, pa roditelja od roditelja od 6 itd.

SELECT Izvor

FROM Rijeke

CONNECT BY Izvor= PRIOR Odredište

START WITH Odredište= 6

Primjer 2:

Znamo da je zagađena rijeka 3. Odrediti potencijalne zagađivače!

SELECT Izvor

FROM Rijeke

CONNECT BY PRIOR Izvor= Odredište

START WITH Odredište= 3

Primjer 3:

Prikazati spisak svih direktnih i indirektnih pritoka rijeke 1 !

SELECT Izvor

FROM Rijeke

CONECT BY PRIOR Izvor=Odredište

START WITH Odredište= 1

Primjer 4:

Koliko rijeka može biti zagađeno ako je zagađena rijeka 1?

Napomena: Dodajemo tabelu "Rijeka" sa kolonama "ID" i "Ime"!

```
SELECT  COUNT(Izvor)
FROM  Rijeke
CONNECT  BY Izvor= PRIOR dest
START  WITH Izvor IN
(SELECT  Rijeka ID
FROM Rijeka
WHERE  Ime='R1')
```

5. PROCESIRANJE I OPTIMIZACIJA UPITA

Upiti bazi podataka su izraženi preko deklarativnih jezika kao što je SQL. Odgovornost softvera baze podataka je da preslika upit u niz operacija podržanih prostornim indeksima i strukturama za pohranjivanje. Glavni cilj je precizno procesiranje upita u najkraćem vremenskom roku. Ovo poglavlje će se baviti tehnikama koje se koriste za procesiranje i optimizaciju upita u prostornim bazama podataka. U relacionim bazama, većina upita je sastavljena od fiksiranog skupa osnovnih operacija. Ove osnovne relacione operacije formiraju gradivne blokove za postavljanje svih složenih upita. Stoga je procesiranje i optimizacija upita podjeljena na dva dijela:

- Kreiranje i fino podešavanje algoritama za svaki od osnovnih relacionih operatora;
- Preslikavanje upita visokog nivoa u kompoziciju ovih osnovnih relacionih operatora i njihova optimizacija korištenjem informacija iz prvog koraka.

Isti obrazac, uz par važnih razlika, se može primjeniti kod prostornih baza podataka. Prvo, domena primjene prostorne baze je raznolikija, i ne postoji opća saglasnost oko skupa prostornih blokova (tj. Prostornih operacija) koji mogu pokriti sve slučajeve koji se mogu pojaviti. Ono oko čega se svi slažu su klase prostornih operacija. Drugo, algoritni za prostorne operacije su intenzive u pogledu ulaza/izlata podataka i korištenja procesora (*eng. CPU – central process unit*). Zbog ovoga je kreiranje procesa složenije nego kod tradicionalnih baza podataka gdje je uobičajena pretpostavka da cijena ulaz/izlaz dominira nad cijenom procesora te da je dobar algoritam taj koji minimizira broj pristupa disku.

Procjena prostornih operacija

Iz perspektive procesiranja upita tri značajna problema karakterišu razlike između prostornih i relacionih baza podataka (Brinkhoff i dr., 1993), to su:

1. Za razliku od relacionih baza podataka, prostorne baze podataka nemaju fiksiran skup operatora koji se koriste kao gradivni blokovi za procjenu upita.
2. Prostone baze podataka se bave ekstremno velikim količinama kompleksnih objekata. Ovi objekti imaju prostorne ekstenzije i ne mogu se prirodno sortirati u jednodimenzionalni poredak.
3. Računski skupi algoritmi se zahtjevaju za testiranje prostornih predikata, te pretpostavka da cijena ulaz/izlaz dominira nad cijenom procesiranja u procesoru više ne vrijedi.

Prostorne operacije

Prostorne operacije se mogu klasificirati u četiri grupe (Gaede i Gunther, 1998):

1. Operacije osvježavanja (*eng. update operaions*) - standardne operacije kao što su modificiraj, kreiraj, itd.
2. Operacije odabira (*eng. selection operations*) - mogu biti dvije vrste:
   a) Tačkasti upit (*eng. point query – PQ*): data je tačka p, pronaći sve prostorne objekte O koji je sadržavaju:
   $$PQ(p) = \{O \mid p \in O.G \neq \emptyset\}$$
   Gdje je *O.G* geometrija objekta O.
   Npr. „Pronaći sve plavne ravni koje sadržavaju svetišta!“ Svetište je konstanta tipa *tačka*.
   b) Poligonalni ili regijski upiti (*eng. range or regional query – RQ*):dat je poligon P, pronaći sve prostorne objekte O koji sijeku P. Kada je upitni poligon pravougaonik tada se upit zove prozorni upit (*eng. window query*). Ovi upiti se nekada zovu i poligonalni upiti.
   $$RQ(P) = \{O \mid O.G \cap P.G \neq \emptyset\}$$
   Npr. „Pronaći šumska područja koja se preklapaju sa plavnim područjem rijeke Nil!“.
3. Prostorno spajanje (eng. spatial join) – baš kao i kod relacionih baza podataka, prostorno spajanje je jedan od važnijih operatora. Kada su dvije tabele R i S spojene prostornim predikatom θ, takvo spajanje se zove prostorno spajanje. Varijanta prostornog spajanja i važan operator u GIS-u je prevlačenje karata (*eng. map overlay*). Ova operacija kombinira dva skupa prostornih objekata i kreira jedan novi skup. Granice skupa ovih novih objekata su određene neprostornim atributima pridruženim operacijom prevlačenja. Npr., ako operacija pridruži istu vrijednost neprostornog atributa dva susjedna objekta tada se ti objekti spajaju.
   $$R \bowtie_\theta S\{o, o' \mid o \in R, o' \in S, \theta(o.G, o'.G)\}$$
   Neki primjeri predikata θ su:
   - Presjeca (eng. intersect);
   - Sadrži (eng. contains);
   - Okruženo je (eng. is_enclosed_by);
   - Udaljenost (eng. distance);

- Smjer (npr. sjeverozapadno, istočno, itd.);
- Susjedni (eng. adjecent);
- Susreće (eng. meet);
- Dodiruje (eng. touch);
- Preklapa (eng. overlap), itd.

Primjer prostornog spajanja je upit:“Pronaći mjesta gdje se preklapaju šumska područja sa plavnim ravnima!“.

4. Prostorno pridruživanje (eng. spatial aggregate) – primjer prostornog pridruživanja je upit:“Pronaći rijeku koja je najbliža kampu!“ prostorno pridruživanje je obično neka varijanta traženja najbližeg susjeda. Za dati objekt *O* pronađi sve objekte *o* na minimalnoj udaljenosti *o'* od objekta *O*.

$$NNQ(o')=\{o \mid \forall o'' : dist(o'.G, o.G) \leq dist(o'.G, o''.G)\}$$

Dva koraka pri procesiranju upita

Procesiranje prostornih upita uključuje kompleksne tipove podataka, npr. granica jezera može imati hiljade čvorova zbog tačnog prikaza. Prostorne operacije obično slijede algoritam od dva koraka kojim efikasno procesiraju velike objekte:

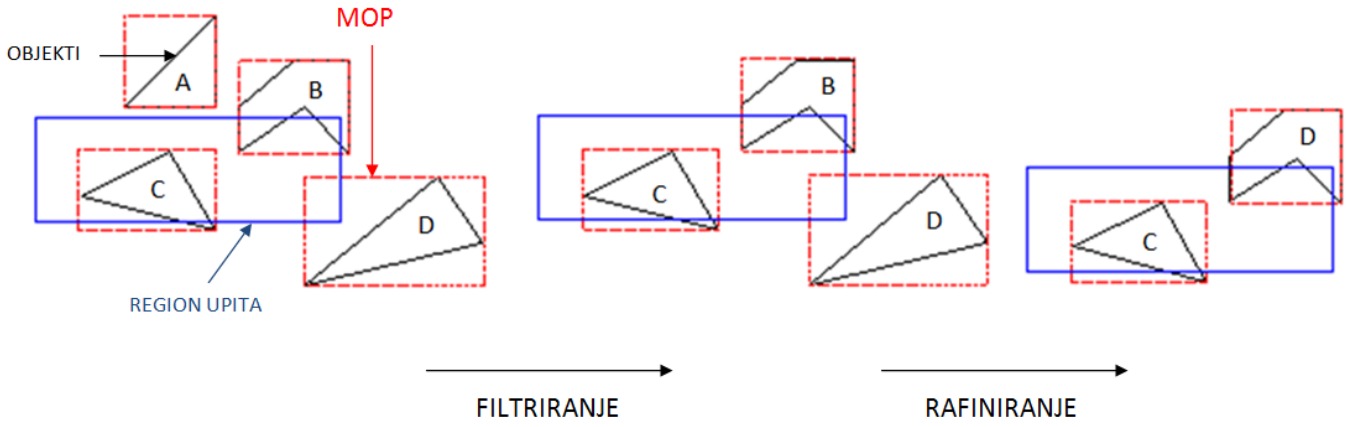
1. **Korak filtriranja**: U ovome koraku prostorni objekti se prikazuju jednostavnijim aproksimacijama kao što su minimalni obuhvatni pravougaonik ili MOP (*eng. minimum bounding rectangle – MBR*). Npr.:“Pronađi sve rijeke čije plavne ravni preklapaju svetišta.“ U SQL-u ovaj upt nbi izgledao ovako:

   SELECT Rijeka.Ime
   FROM Rijeka
   WHERE overlap(Rijeka.Plovna-ravan, :Svetište)

   Prije korisnički definiranih parametara se stavlja dvotačka, da bi se naznačilo da se radi o korisnički definiranom parametru, npr. :Svetište. Ako aproksimiramo plavne ravni svih rijeka korištenjem MOP-a, tada će određivanje da li se tačka nalazi unutar ili izvan MOP-a biti jeftinije nego provjeravanje da li se tačka nalazi u nepravilnom poligonu, što je stvarni oblik plovne ravni. Odgovor ovoga aproksimativnog testa je nadskup skupa stvarnih rješenja. Ovaj nadskup se nekada zove i *skup kandidata*. I prostorni predikat može biti zamjenjen aproksimacijom da bi se pojednostavio optimizator upita.

Npr., touch(Rijeka.Plovna-ravan, :Svetište) se može zamjeniti sa overlap(MBR(Rijeka.Plovna-ravan, :Svetište), MBR(:Svetište)) u koraku filtriranja. Mnogi prostorni operatori kao što su: inside, bafer, sjeverno od i sl. mogu biti aproksimirane pomoću preklapanja između odgovarajućih MOP-a. Ovakva transformacija garantuje da će negativne n-torke iz konačnog odgovora, u kome se koriste tačne geometrije, biti eliminirane u koraku filtriranja.

2. **Korak rafiniranja (pročišćavanja)**: Ovdje se za ispitivanje koristi stvarna geometrija iz skupa kandidata i pravi prostorni predikat. Ovo obično zahtjeva korištenje algoritama koji su teški za procesor. Ovaj korak se iz tih razloga nekada procesira izvan prostorne baze podataka u nekom od GIS programa, pri tome se koristi skup kandidata koji je odredila baza podataka u koraku filtriranja.



Prostorna selekcija

U najgeneralnijem slučaju, selekcioni uslov može biti kombinacija više primitivnih selekcionih uslova. U tradicionalnim bazama uslovi se prvo izražavaju u konjuktivnoj normalnoj formi[5] (CNF – conjuctive normal form), a zatim se koristi kombinacija heširanja (hashing) i indeksiranja (index trees) za ubrzavanje procesiranja upita. Redoslijed procesiranja uslova je bitan jer različiti uslovi imaju različite cijene procesiranja. Ovo nije slučaj u

[5]Atomarnu formulu i njezinu negaciju nazivamo literal. Formulu oblika $A1 \wedge A2 \wedge A3 \ldots \wedge An$ nazivamo konjukcija (Ai su proizvoljne formule), a formulu oblika $A1 \vee A2 \vee A3 \ldots \vee An$ nazivamo disjunkcijom. Elementarna konjunkcija je konjunkcija literala, a elementarna disjunkcija je disjunkcijaliterala. ***Konjunktivna normalna forma je konjunkcija elementarnih disjunkcija***. Disjunktivna normalna forma je disjunkcija elementarnih konjunkcija.

tradicionalnim bazama podataka gdje je uobičajena pretpostavka da je, u prosjeku, trošak svih neprostonih uslova jednak.

Optimizacija upita

Upiti se obično izražavaju deklarativnim jezikom visokog nivoa kao što je SQL. Ovo znači da se specificira šta se želi dobiti kao rezultata ali ne i način na koji se dođe do rezultata – to je ostavljeno bazi. Metrika strategije, ili plan procjene, je vrijeme koje je potrebno da se izvrši upit. U tradicionalnim bazama metrika je uveliko funkcija ulazno/izlaznih troškova jer su dostupni tipovi podataka i funkcije koje operiraju relativno jednostavni za računanje. Kod prostornih baza podataka situacija je drukčija jer su uključeni kompleksni tipovi podataka i funkcije koje su zahtjevne za procesor. Zbog toga je zadatak odabira optimalne strategije više zastupljen u prostornim bazama nego u tradicionalnim bazama.

Optimizator upita, modul u softveru baze podataka, generiše različite planove evaluacije i određuje odgovarajuću strategiju izvršavanja. Optimizator upita izvlači informacije iz sistemskog kataloga i zatim ih kombinira sa određenom heuristikom[6] i tehnikama iz dinamičkog programiranja da bi došao do odgovarajuće strategije. Optimizator upita rijetko, ili čak nikada, izvršava najbolji plan. Razlog tome je kompleksnost računanja opštih troškova optimizacije (Selinger i dr., 1997). osnovna ideja optimizacije je da se izbjegnu najgori planovi i izabere neki od dobrih. Procedura koju vrši optimizator se može podijeliti u dva dijela: logička transformacija i dinamičko programiranje (Adam i Gangopadhyay, 1997).

Logička transformacija

Prije nego optimizator može početi raditi na upitu, deklarativna izjava visokog nivoa se mora skenirati kroz **parser**[7]. Parser provjerava sintaksu i transformiše upit u **drvo upita**. Kod klasičnih baza podataka parser je prilično jednostavan, s druge strane kod prostornih baza parser je znatno sofisticiraniji jer se radi o proširenim bazama podataka u koje se mogu dodavati korisnički definisani tipovi podataka i metode. Umjesto osnovnih tipova podataka sada parser treba preslikati ove korisnički definisane podatke i metode u sintatički ispravno drvo upita.

[6] **Heuristika** je tehnika rješavanja ili brže od klasičnih metoda, ili nalaženja približnog rješenja kada klasični metodi ne mogu da nađu tačno rješenje.
[7] **Parser** (od engleske riječi to parse "analizirati" odnosno latinske pars "dio") jeste računarski program ili komponenta računarskog programa koja analizira neki sadržaj tako što utvrdi hijerarhiju među elementima.Analiza se vrši na osnovu zadane gramatike.

U drvetu upita **listovi čvorovi** odgovaraju relacijama koje su uključene u upit, a unutrašnji čvorovi odgovaraju osnovnim operacijama (SELECT, PROJECT, JOIN i skupne operacije) koje čine upit. Procesiranje upita počinje na čvorovima listovima i ide drvetom dok se ne izvrši operacija u korijenskom čvoru.

Neka je data šema baze podataka:

Jezero (JID, ime, dubina, oblik=poligon)

Park (PID, ime, oblik=poligon)

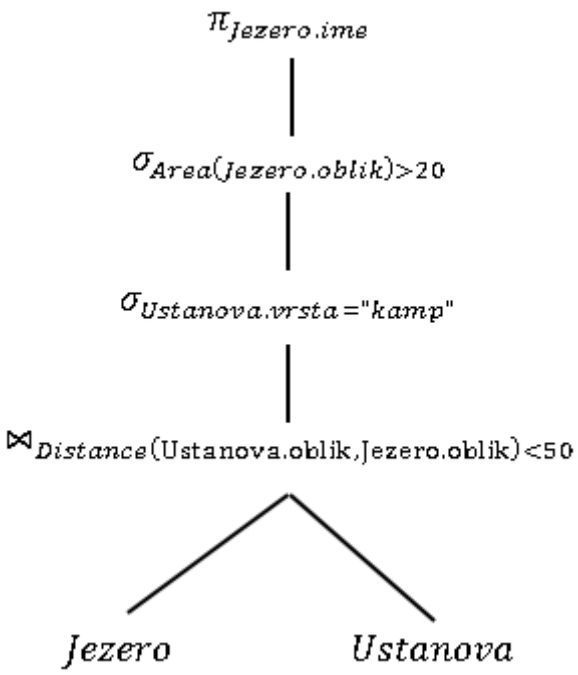
Ustanova (UID, vrsta, oblik=tačka)

Upit: pronaći sva jezera čija je površina veća od 20km kvadratnih i koja se nalaze u krugu od 50km od kampa!

Podrazumijeva se da površine i udaljenosti nisu poznate već ih treba izračunati!

SELECT Jezero.ime
FROM Jezero, Ustanova
WHERE **Area** (Jezero.oblik) > 20 AND
Distance (Ustanova.oblik, Jezero.oblik) < 50 AND
Ustanova.vrsta = „kamp“

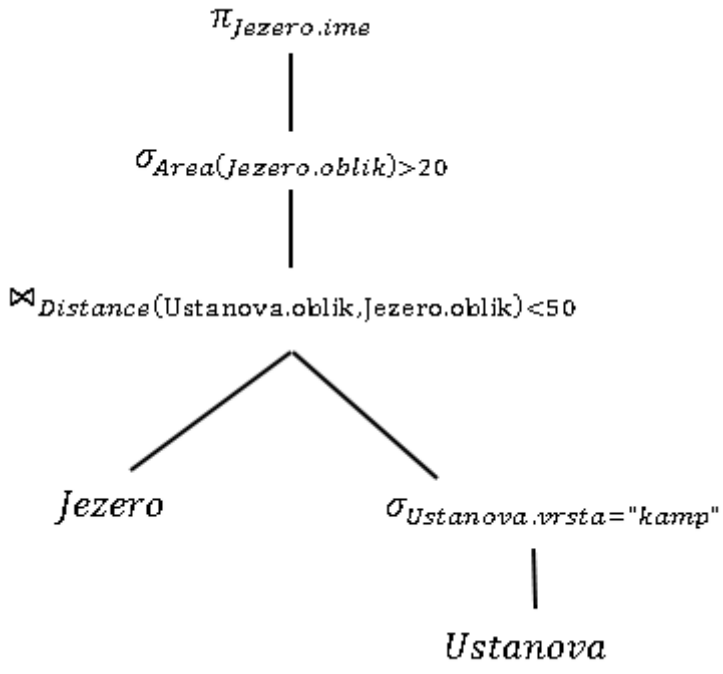
Za ovaj upit drvo upita može imati sljedeći oblik.



Strategija upita koja je prikazana na prethodnom drvetu upita je loša. Razlog je to što je operacija spajanja skupa, a vrši se na samom početku upita i u njoj učestvuje velika količina podataka. Da bi se optimizovao upit potrebno je smanjiti količinu podataka koja učestvuje u spajanju. To se može uraditi ako se neprostorni operator selekcije

$$\sigma_{Ustanova\ .vrsta="kamp"}$$

pomjeri ispred operacije spajanja. Tada bi drvo upita dobilo sljedeći oblik:



U koraku logičke transformacije drvo upita koje generiše parser se preslikava u ekvivalentna drva upita. Ova ekvivalentnost je rezultat formalnog skupa pravila koja su nasljeđena iz relacione algebre. Nakon što se ekvivalentna drva upita numerišu, mogu se filtrirati ona drva upita koja očigledno nisu kandidati za strategiju izvršenja upita. Logička transformacija je analogna koraku filtriranja u procesiranju upita. To je brz način eliminisanja drva upita koja nisu optimalna. Pravila kod neprostornih baza podataka su jasna, međutim pravila optimizacije za prostorne baze mogu biti dvosmislena. Npr., razmotrimo šta bi se desilo ako bi se prostorni opertor

$$\sigma_{Area\ (Jezero\ .oblik\ )>20}$$

pomjerio prema listovima drveta upita.

To bi značilo da se prostorno spajanje izvršava nakon prostornog predikata. Međutim, sada postoje dvije korisnički definisane funkcije čije cijene izvršenja treba međusobno rankirati. Ako je funkcija Area računarski zahtjevnija (skuplja) od Distance funkcije tada nije preporučljivo vršiti prostornu selekciju prije spajanja. Iz tog razloga glavno pravilo optimizacije relacionih baza podataka koje kaže „da se *select* i *project* operacije vrše prije spajanja i binarnih operacija“ više nije bezuslovno. Sve što možemo reći je da se *select* i *project* operacije trebaju pomjeriti prema listovima drveta upita.

Pravila ekvivalentnosti kod operatora

Postoje dvije važne ekvivalencije za operator selekcije.

1. $\sigma_{C1 \wedge C2 \wedge \ldots Cn}(R) \equiv \sigma_{C1}(\sigma_{C2}(\ldots(\sigma_{Cn}(R))\ldots))$

   Ovo pravilo dozovljava kombinovanje više selekcija u jednu selekciju. Za procesiranje prostornih upita, gdje selekcioni uslovi mogu biti kombinacija prostornih i neprostornih predikata bilo bi korisno pomjeriti sve neprostorne uslove prema desno kako bi se ti uslovi izvršili prije prostonih.

2. $\sigma_{C1}(\sigma_{C2}(R)) \equiv \sigma_{C2}(\sigma_{C1}(R))$

   Ovo pravilo govori da se selekcioni uslovi mogu testirati bilo kojim redoslijedom. Ponovo, najbolje je izvršavati neprostorne uslove prije prostornih zbog cijene izvršavanja.

Za projekcioni operator vrijedi pravilo:

Ako je a_i skup atributa takvih da je $a_i \subset a_{i+1}$ za $i = 1 \ldots . n - 1$, tada vrijedi:

$$\pi_{a1} \equiv \pi_{a1}(\pi_{a2}(\pi_{a3} \ldots (\pi_{an}(R)) \ldots))$$

Za unakrsni produkt i spajanje vrijede pravila komutativnosti, asocijativnosti i ekvivalencije.

Za dva ili više operatora vrijede slijedeća pravila ekvivalencije:

- Ako selekcioni uslov c uključuje atribute sadržane u projekcionom operatoru tada vrijedi:
$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$
- Ako selekcioni uslov c uključuje jedan atribut koji se pojavljuje samo u R ali ne u S, tada vrijedi:
$$\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$$

Optimizacija bazirana na troškovima: Dinamičko programiranje

Dinamičko programiranje je tehnika određivanja najboljeg plana izvršenja iz skupa rješenja. Optimalno rješenje se izvodi na osnovu troška funkcije. Svaki plan se procjenjuje na osnovu troška funkcije, a plan sa minimalnim troškom je optimalni plan. U koraku dinamičkog programiranja fokus je na svakom čvoru drveta upita i vrši se numeracija različitih strategija izvršenja koje su dostupne za svaki čvor. Različite strategije procesiranja za svaki čvor u kombinaciji sa čitavim drvetom upita čine **prostor plana**. Kardinalnost prostora plana je obično jako velika, a brzina izvršenja zavisi od reda veličine prostora (što je prostor veći više vremena treba za izvršenje). Dodatno, pretpostavke o parametrima koje koriste funkcije su često pojednostavljene,a nekada čak i pogrešne. Nadalje, vrijeme optimizacije (vrijeme potrebno da se odabere plan izvršenja) također mora biti minimalno. Uzimajući sve ovo u obzir, kao što je i prethodno naglašeno, **cilj je odabrati dobar plan, a ne neophodno najbolji**.

U srcu dinamičkog programiranja je korištenje funkcija troška za procjenu strategija izvršenja plana. Faktori koje funkcije troška uzimaju u obzir su (Elmasri i Navathe, 2000):

- Trošak pristupa: trošak traženja i prenosa podataka iz sekundarnih skladišta.
- Trošak skladištenja: trošak pohranjivanja privremenih međurelacija koje kreira strategija izvršavanja upita.

- Trošak računanja: trošak procesora (CPU) za izvršenje operacija.
- Trošak komunikacije: trošak prenosa podataka između klijenta i servera.

Sistemski katalog

Informacije koje su potrebne funkciji troška da bi napravila optimalnu strategiju izvršenja se čuvaju u sistemskom katalogu. Ove informacije sadrže veličinu svakog fajla, broj pohranjivanja u fajlu i broj blokova preko kojih se šire sačuvani fajlovi. Takođe treba da postoji informacija o indeksima i atributima indeksa. Selektivnost i diferencijalni trošak predikata takođe mora biti uključena. U nekim slučajevima korisno je materijalizirati skupe, korisnički definisane funkcije i indeksirati njihove vrijednosti za brže preuzimanje. Npr., funkcija Area (Geometry) može biti sračunata unaprijed i te vrijednosti mogu biti indeksirane. Ovo ubrzava procjenu upita i sistemski katalog može spremati informacije za ove „teške“ funkcije (teške u smislu proceiranja, znači da crpe mnogo resursa).

Funkcije troška

Funkcije troška koje se koriste u relacionim bazama podataka (Stonebraker and Moore, 1997) su varijacije od:

trošak = Očekivanje(ispitani zapisi) + K*Očekivanje(očitane stranice),

gdje je Očekivanje(ispitani zapisi) očekivani broj čitanja pohranjivanja i samim tim mjera CPU vremena i Očekivanje(očitane stranice) je očekivani broj stranica očitanih iz memorije i mjera ulazn/izlaz vremena. Faktor K je mjera važnosti CPU resursa u odnosu na ulaz/izlaz resurse.

Numerisanje alternativnih planova

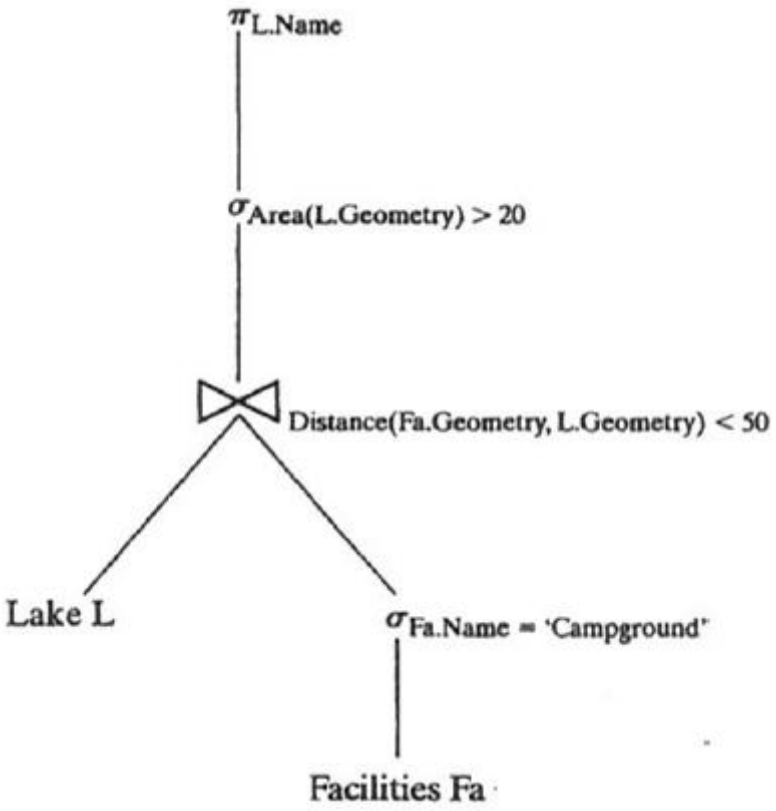
Nakon formulacije upita optimizator numeriše skup planova i zatim izvršava plan sa najnižom procijenjenom cijenom. Za numeraciju različitih planova se koristi kombinacija implementacijskih tehnika (koje su dozvoljene za svaki prostorni i neprostorni operator) i pravila jednakosti relacione algebre. Plan upita se sastoji od drveta upita sa dodatnim oznakama na svakom čvoru koji određuje metodu pristupa koja će se koristiti za svaki čvorni operator.

Rastavljanje i spajanje

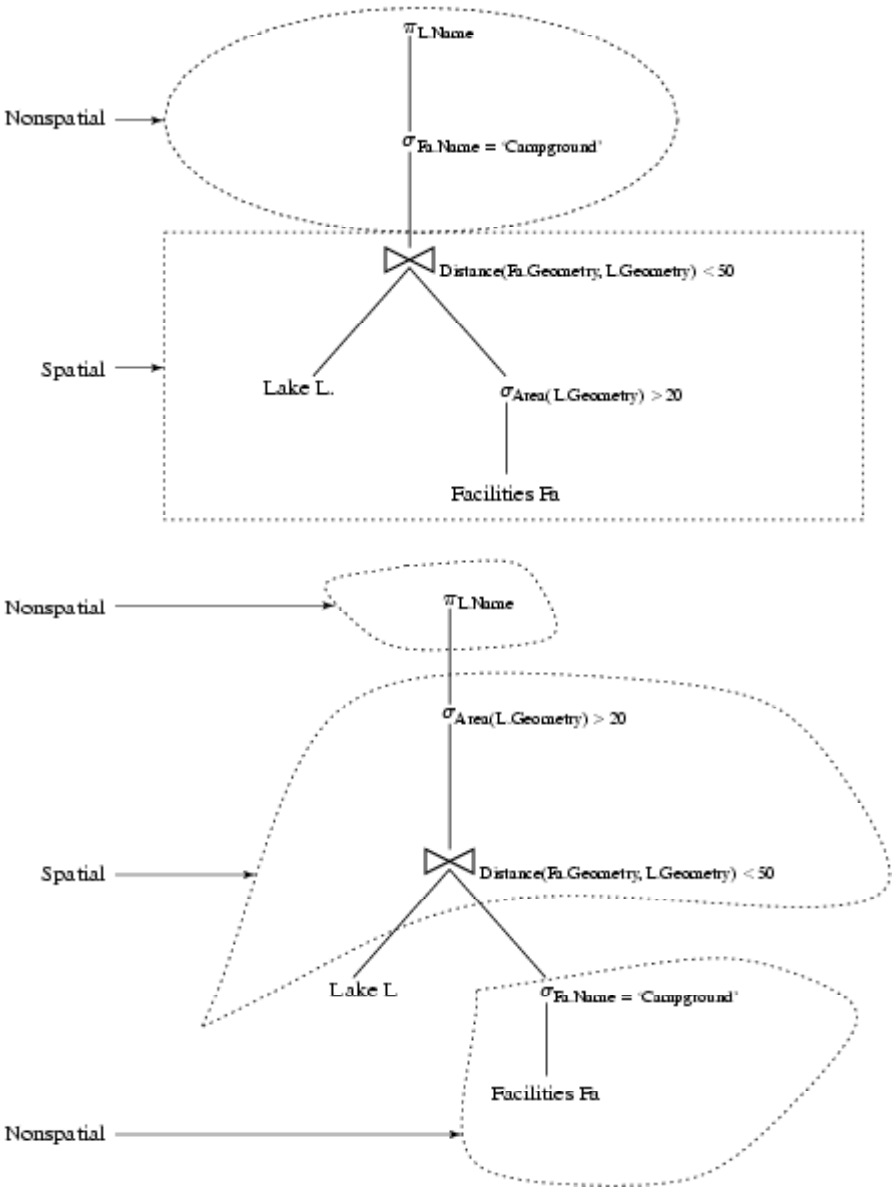
Upiti u sistemima za upravljanje prostornim bazama podataka zasnovani na hibridnoj arhitekturi su **rastavljeni** na prostorni i neprostorni dio. Podupiti su optimizirani u posebnim modulima određenim za prostornu i neprostornu optimizaciju i na kraju **spojeni** da osiguraju željeni rezultat. Rastavljanje se bazira na tri kriterija:

1. kreira se podupit za svaki prostorni predikat,
2. kreira se podupit za svaki neprostorni predikat povezan sa veznicima za prostorni predikat, i
3. kreira se podupit za sve neprostorne predikate.

Razmotrimo stablo upita prikazan na sljedećoj slici.



Postoje mnogo načina dekompozicije ovog stabla upita na prostorne i neprostorne dijelove. Na slici ispod će biti prikazane dvije metode dekompozicije. Svaka od ovih metoda rastavljanja može dati različite rezultate, a još jedan sloj softvera za određivanje optimalne dekompozicije mora biti uvršten u optimizator upita. Uz rastavljanje, moraju biti naglašeni problemi u vezi sa rasporedom podupita. Na primjer, podupiti mogu biti procesirani sekvencijalno ili paralelno, ovisno kako su upareni.


Nonspatial
Spatial
Nonspatial
Spatial
Nonspatial
Lake L.
Facilities Fa
Lake L
Facilities Fa

Literatura

Egenhofer, M., & Herring, J. (1994). Categorizing binary topological relations between regions, lines and points in geographic database.

Galić, Z. (2006). *Geoprostorne baze podataka.* Zagreb: Golden marketing - Tehnička knjiga.

Galić, Z., & Govedarica, M. (2007). Preuzeto od http://bib.irb.hr/datoteka/299861.Model_podataka_katastra_BiH.pdf

Matijević, H. (2004). Modeliranje podataka katastra. Zagreb, Hrvatska: Sveučilište u Zagrebu, Geodetski fakultet.

Rigaux, P., Scholl, M., & Voisard, A. (2002). *Spatial databases with application to GIS.* San Francisco: Elsevier.

Roić, M., & Cetl, V. (2003). Preuzeto od http://www.geof.hr/~vcetl/radovi/RoicCetl.pdf

Roić, M., Matijević, H., & Cetl, V. (2002). URL: http://www.geof.hr/~vcetl/radovi/RoicCetlMatijevic.pdf

Shekhar, S., & Chawla, S. (2003). *Spatial Databases: A Tour.* New Jersey: Pearson Eduction.

Ponjavić M.: Osnovi geoinformacija, 2011.

URL 1: http://www.ce.utexas.edu/prof/maidment/visual/dallas/david1/sld027.htm (preuzeto 20.09.2017.)

URL 2: http://www.cs.toronto.edu/~nn/csc309/guide/pointbase/docs/html/htmlfiles/dev_datatypesandconversionsFIN.html (preuzeto 30.10.2017.)

URL3: http://cs.elfak.ni.ac.rs/nastava/pluginfile.php/1959/mod_resource/content/1/grafovi_1_.pdf (preuzeto: 01.07.2013.)