

Relacione baze podataka

Skripta za školsku 2023/2024. godinu

Mirjana Maljković i Ivana Tanasijević

Sadržaj

1	Uvod	11
2	Uvod u baze podataka	13
2.1	Pre baza podataka	13
2.2	Sistem baza podataka	13
2.3	Entiteti i odnosi	15
2.4	Informacije o studentima	15
2.5	Model podataka	17
2.6	Prednosti rada sa bazom podataka	17
2.7	Nezavisnost podataka	18
2.8	Neformalni pogled na relacioni model	19
2.8.1	Domen	19
2.8.2	Relacije	19
2.8.3	Terminologija	20
2.8.4	Primeri operatora za obradu	21
2.8.5	Nedostajuće vrednosti	22

2.8.6	Osnovna ograničenja u relacionim bazama podataka	22
2.9	Formalni pogled na relacioni model	23
3	Arhitektura sistema baza podataka	25
3.1	ANSI/SPARC arhitektura	25
3.1.1	Spoljašnji nivo	26
3.1.2	Konceptualni nivo	27
3.1.3	Unutrašnji nivo	27
3.1.4	Preslikavanje nivoa	27
3.2	Funkcije SUBP	28
3.3	Klijent-server arhitektura	29
3.4	Utility programi	29
3.5	Distribuirana obrada	29
4	Relaciona algebra	31
4.1	Manipulativni deo relacionog modela podataka	31
4.2	Relaciona algebra	31
4.3	Relacioni izraz algebre i relaciono zatvorenje	32
4.4	Semantika originalne algebre	32
4.4.1	Restrikcija	32
4.4.2	Projekcija	33
4.4.3	Dekartov proizvod	34
4.4.4	Prirodno spajanje	34
4.4.5	Slobodno spajanje	35
4.4.6	Unija	36
4.4.7	Presek	36
4.4.8	Razlika	37
4.4.9	Deljenje	38
4.5	Dodatni operatori	39

4.6	Minimalni skup operatora	40
4.7	Svrha relacije algebre	40
4.8	Relaciona kompletnost	40
4.9	Algebarski zakoni	41
4.10	Prioritet operatora	41
5	Relacioni račun	43
5.1	Relacioni račun n-torki	43
5.2	Relacioni račun domena	48
5.3	Relaciona algebra ili relacioni račun	49
6	Integritet i sigurnost	51
6.1	Klasifikacija ograničenja integriteta	51
6.2	Ključevi	52
6.3	Pojam sigurnosti podataka	54
6.4	Ostali aspekti sigurnosti	54
7	OLAP	55
7.1	Uvod	55
7.2	Arhitektura	56
8	Teorija zavisnosti	57
8.1	Funkcionalne zavisnosti	57
8.1.1	Trivijalne i netrivialne funkcionalne zavisnosti	58
8.1.2	Pravila izvođenja	58
8.1.3	Skupovi funkcionalnih zavisnosti	59
8.1.4	Zatvorenje skupa funkcionalnih zavisnosti	63
8.1.5	Zatvorenje skupa atributa	66

8.1.6	Izračunavanje kandidata za ključ preko zatvorenja skupa atributa .	71
8.1.7	Pokrivač skupa funkcionalnih zavisnosti	77
8.1.8	Nereducibilni skup funkcionalnih zavisnosti	77
8.1.9	Projekcija funkcionalnih zavisnosti	84
8.2	Višeznačne zavisnosti	86
8.3	Zavisnosti spajanja	88
9	Normalizacija	91
9.1	Redudantnost i anomalije	91
9.2	Vrste funkcionalnih zavisnosti	92
9.3	Kandidati za ključ, ključni i neključni atributi	92
9.4	Normalne forme	93
9.4.1	Prva normalna forma	93
9.4.2	Druga normalna forma	95
9.4.3	Treća normalna forma	96
9.4.4	Bojs-Kodova normalna forma	98
9.4.5	Dodatni primeri	98
9.4.6	Četvrta normalna forma	112
9.4.7	Peta normalna forma	112
9.5	Dobre i loše strane dekompozicije	112
9.6	Denormalizacija	112
10	Pretraživanje podataka	113
10.1	Uvod	113
10.2	Indeksi	115
10.2.1	Primarni i sekundarni indeks	116
10.2.2	Klasterovan indeks	116
10.2.3	Gusti i retki indeksi	117
10.2.4	Indeksi na više nivoa	119

10.2.5	Ažuriranje indeksa	120
10.2.6	Pretraživanje po dupliranim vrednostima	120
10.2.7	Uputstva pri kreiranju indeksa	123
10.3	B+ drveća	123
10.3.1	Pretraživanje podataka	124
10.3.2	Unošenje podataka	126
10.3.3	Brisanje podataka	127
10.3.4	Primene	128
10.3.5	Efikasnost	129
11	Upravljanje transakcijama	131
11.1	Transakcije	131
11.2	Oporavak	132
11.2.1	Oporavak sistema	133
11.2.2	Oporavak medija	135
11.2.3	Dvofazni COMMIT	135
11.3	Konkurentnost	136
11.3.1	Problemi u konkurentnom radu	136
11.3.2	Zaključavanje	138
11.3.3	Rešenje problema u konkurentnom radu	139
11.3.4	Mrtva petlja	140
11.3.5	Izbegavanje mrtve petlje	142
11.3.6	Serijabilnost	142
11.3.7	Nivoi izolacije	143
11.3.8	Namera zaključavanja	144
11.4	Zadaci	145
12	Optimizacija	149
12.1	Uvod	149

12.2	Faze obrade upita	150
12.2.1	Parsiranje upita i provera semantike	150
12.2.2	Konverzija upita u kanonički oblik	151
12.2.3	Analiza i izbor kandidata za procedure niskog nivoa	151
12.2.4	Formiranje planova upita i izbor najjeftinijeg	151
12.3	Transformacija izraza	151
12.4	Statistika u bazi podataka	154
12.5	Implementacija operatora spajanja	155
12.5.1	Gruba sila	155
12.5.2	Korišćenje indeksa	155
12.5.3	Korišćenje hash-a	155
12.5.4	Merge metoda	155
12.5.5	Hash metoda	156
13	Relacioni upitni jezik SQL	157
13.1	Predavanje 1	158
13.1.1	Jezik za manipulaciju podacima	160
13.2	Predavanje 2	162
13.2.1	Uklanjanje dupliranih redova iz rezultata	162
13.2.2	Dodatni operatori za postavljanje uslova	163
13.2.3	Izvedene kolone	165
13.2.4	Uređivanje rezultata upita	167
13.2.5	Proizvod tabela	169
13.2.6	Spajanje tabela	170
13.3	Predavanje 3	174
13.3.1	Alias	174
13.3.2	Skupovni operatori	175
13.3.3	Podupiti	178
13.4	Predavanje 4	180
13.4.1	Kvantifikovana poredenja	180

13.4.2	Egzistencijalni kvantifikator	180
13.4.3	Tabela sa konstantnim redovima	183
13.4.4	Specijalni registri	184
13.4.5	Skalarne funkcije	184
13.5	Predavanje 5	189
13.5.1	Agregatne funkcije	195
13.5.2	Grupisanje rezultata	197
13.5.3	Izdvajanje rezultata za željene grupe	198
13.5.4	Imenovanje međurezultata	200
13.5.5	Izrazi case	201
13.5.6	Skalarna funkcija raise_error	202
13.5.7	Korisnički definisane funkcije	203
13.5.8	Dodavanje entiteta	204
13.5.9	Promena vrednosti tabele	206
13.6	Predavanje 6	207
13.6.1	Uklanjanje entiteta	207
13.6.2	Menjanje entiteta tabele na osnovu sadržaja druge tabele/tabela	207
13.6.3	Prikaz željenog broja redova	209
13.6.4	DDL	210
13.6.5	Pravljenje baze podataka	211
13.6.6	Pravljenje šeme baze podataka	212
13.6.7	Pravljenje tabele	213
13.6.8	Korisnički definisani tipovi	215
13.6.9	Promena bazne tabele	217
13.6.10	Indeksi	218
13.6.11	Dodatni primeri	218
13.6.12	Okidači	220
13.6.13	Sigurnost u SQL-u	222
13.6.14	Pogledi	223
13.6.15	Kreiranje pogleda u SQL	223
13.6.16	Brisanje pogleda u SQL	224

13.6.17	Ažuriranje pogleda u SQL-u	224
13.6.18	WITH CHECK OPTION	226
13.6.19	Okidači nad pogledima	229
13.6.20	Podsistem za autorizaciju	231
13.6.21	GRANT naredba	231
13.6.22	REVOKE naredba	232
13.6.23	GRANT naredba - WITH GRANT OPTION	233
13.7	Predavanje 7	234
13.7.1	OLAP	234
13.7.2	Unakrsno tabeliranje	234
13.7.3	Analitičke funkcije	240
13.7.4	Rekurzivni SQL	255
13.7.5	LOB	269
13.7.6	Materijalizovani upiti	270
13.8	Predavanje 10	273
13.8.1	Katalog	273
13.9	Predavanje 11	279
13.9.1	Transakcije	279
13.10	Predavanje 12	281
13.10.1	Optimizacija	281
13.11	Predavanje 13	284
13.11.1	XML	284
13.11.2	XQUERY	285
13.11.3	XQUERY FLWOR	288
13.11.4	Formiranje nove XML strukture pomoću konstruktora	290
13.11.5	Operatori	291
13.11.6	Korišćenje agregatnih funkcija	292
13.11.7	SQL/XML	292
13.11.8	Upotreba XMLTABLE	296
13.11.9	Upotreba XMLEXISTS	297

1. Uvod

Rukopis sadrži materijal koji prati izlaganja na predavanjima u okviru kursa Relacione baze podataka na Matematičkom fakultetu za školsku 2023/2024. godinu. U rukopisu je sažeto predstavljen materijal iz glavne literature za predmet - knjiga *Uvod u relacione baze podataka* profesorke dr Gordane Pavlović Lažetić, materijali profesora dr Nenada Mitića i izdvojena poglavlja iz knjiga *An Introduction to Database Systems*, *Database Design and Relational Theory: Normal Forms and All That Jazz*, *SQL and Relational Theory: How to Write Accurate SQL Code*, autora C.J.Date. Pored navedenih materijala, korišćena je i druga literatura navedena u bibliografiji.

U poslednjem poglavlju je opisan upitni jezik SQL i RSUBP DB2. Poglavlje je podeljeno prema materijalu koje je obrađivano po nedeljama. U okviru opisa SQL naredbi klauzule/opcije koje ne moraju obavezno da se navedu su uokvirene zagradama [], nakon kojih je označeno koliko puta može navedena opcija da se ponovi u naredbi. Ako je nakon opisa opcije naveden karakter:

- ? opcija može da se navede nula ili jedanput;
- + opcija može da se navede jedanput ili više puta;
- * opcija može da se navede nula ili više puta.

2. Uvod u baze podataka

Ovo poglavlje sadrži pojednostavljen opis pojmova i koncepata sistema za baze podataka, kako bi student stekao širu sliku o njima. U narednim poglavljima će oni biti detaljnije opisani.

2.1 Pre baza podataka

Dok nisu postojale baze podataka datoteke su se koristile za čuvanje podataka, a programski sistemi su im pristupali i koristili podatke u njima za obradu. Posebni programi su pisani za rad sa tim podacima. Neki od nedostataka ovakvog pristupa su: ponavljanje istih podataka za potrebe različitih aplikacija, nekonzistentnost podataka (ako se isti podatak čuva na više mesta, može se desiti da se promeni samo na jednom mestu), otežan pristup više korisnika istoj datoteci. Da bi se prevazišao problem čuvanja podataka u datotekama, 60-tih godina prošlog veka se pojavio pojam baze podataka [6].

2.2 Sistem baza podataka

Sistem baza podataka je u osnovi sistem za računarsko zapisivanje i čuvanje slogova, tj. sistem čija je svrha da čuva informacije i dozvoli korisniku da te informacije dobije i menja [5]. Informacija može biti bilo šta što je bitno za korisnike baze podataka. Informacija i podatak se nekad koriste kao sinonimi, a nekad podatak označava ono što je sačuvano u bazi podataka, a informacija predstavlja njeno značenje ([1], [3]).

U komponente sistema baze podataka spadaju:

- podaci;

- hardver;
- softver;
- korisnici.

Za podatke u bazi podataka je potrebno obezbediti da budu integrisani i deljivi. Pošto podaci mogu biti smešteni u jednoj bazi podataka ili rapoređeni u više baza podataka, bitno je da podaci budu integrisani, odnosno da u podacima ima što manje ili nimalo redundantnosti (ponavljanja). Sistemi koji koriste bazu podataka mogu biti takvi da omogućavaju jednokorisnički ili višekorisnički pristup bazi. Kod jednokorisničkih sistema samo jedan korisnik može u jednom trenutku da pristupi bazi podataka koja se obično čuva na ličnom računaru. Kod višekorisničkog pristupa bazi podataka, više korisnika u istom trenutku može da pristupi bazi podataka, te je potrebno obezbediti konkurentan pristup podacima, odnosno da svaki korisnik radi sa bazom podataka kao da je samo on koristi. Retko kada je dozvoljeno da svaki korisnik kod višekorisničkog pristupa bazi podataka vidi sav sadržaj baze podataka, već se za različite korisnike ili grupe korisnika prave *pogledi* preko kojih se definiše kom delu baze podataka korisnik ili grupa može da pristupi. U hardver sistema baze podataka spadaju spoljašnji memorijski uređaji, procesor i glavna memorija. Spoljašnji memorijski uređaji se koriste za čuvanje podataka. Procesori i glavna memorija se koriste za izvršavanje softvera sistema baza podataka.

Upravljač bazom podataka ili sistem za upravljanje bazom podataka (SUBP, eng. database management system - DBMS) je najznačajnija softverska komponenta u sistemu baza podataka. SUBP se nalazi između fizički sačuvanih podataka i korisnika i on obrađuje sve zahteve za pristup bazi podataka koje upućuje korisnik. SUBP se može uporediti sa programskim jezikom. Kako programski jezik štiti programera od detalja na hardverskom nivou, tako SUBP štiti korisnika od načina čuvanja podataka. Ostale softverske komponente su alati za razvoj aplikacija, upravljač transakcijama,...

Korisnici sistema baza podataka se mogu podeliti u tri grupe:

- aplikativni programeri;
- krajnji korisnici;
- administratori.

Aplikativni programeri su odgovorni za pisanje programa u nekom programskom jeziku (C++, Java, Python ...) koji pristupaju bazi podataka preko SUBP. Krajnji korisnici interaktivno pristupaju bazi podataka preko neke specifične aplikacije koju je napravio aplikativni programer ili preko interfejsa aplikacije koja je sastavni deo sistema i koja se naziva obrađivač upitnog jezika (eng. query language processor). Preko obrađivača upitnog jezika korisnik jednostavno zadaje naredbe SUBP, npr. za izdvajanje podataka ili menjanje podataka. SQL je upitni jezik koji se koristi kod relacionih baza podataka za zadavanje naredbe.

U administratore se ubrajaju administrator podataka i administrator baze podataka. Administrator podataka (eng. data administrator, DA) je osoba koja razume postojeće podatke, odlučuje koji podaci će biti čuvani u bazi podataka i ustanovljava pravila za održavanje i rad sa podacima po njihovom čuvanju u bazi podataka. Administrator podataka nije tehničko

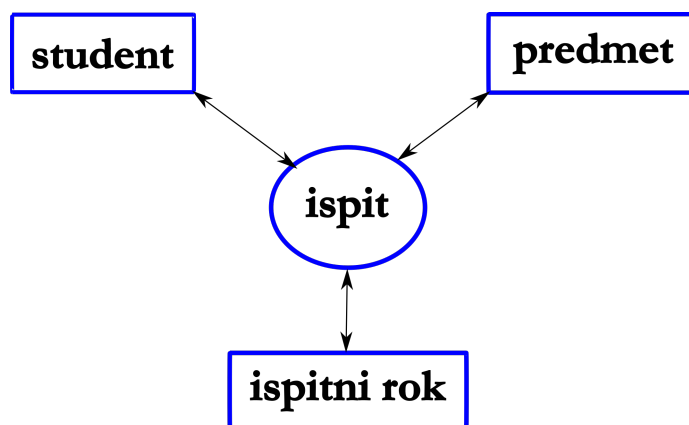
lice, već pripada upravljačkim strukturama. Administrator baze podataka (eng. database administrator, DBA) je tehničko lice koje pravi bazu i implementira kontrolne strukture. DBA je odgovoran za implementaciju odluka DA kao i za rad sistema, performanse, ...

Baza podataka je skup postojećih podataka koji se koriste od strane aplikativnih sistema u nekom okruženju. Da bi podaci bili postojani mora da važi da kada se jednom nađu u bazi ne mogu da budu uklonjeni iz baze bez eksplicitnog zahteva SUBP.

2.3 Entiteti i odnosi

Entitet je bilo koji objekat o kom želimo da čuvamo informacije u bazi podataka. Na primer, osnovni entiteti studentske baze podataka su studenti, predmeti, nastavnici, ispitni rokovi ... Između dva ili više osnovnih entiteta mogu postojati veze. Na primer, student sluša neki predmet, nastavnik predaje neki predmet, student u nekom ispitnom roku polaže neki predmet kod nekog nastavnika ... Odnosni se mogu posmatrati kao posebna vrsta entiteta. Svaki od entiteta ima svojstva koja odgovaraju informacijama koje je potrebno sačuvati o njima.

Za predstavljanje osnovnih entiteta i odnosa koristi se E/R dijagram (eng. entity/relationship diagram). E/R dijagram za malu studentsku bazu je prikazan na slici 13.1. Osnovni entiteti su student, predmet i ispitni rok i oni su na dijagramu uokvireni pomoću pravugaonika. Ispit predstavlja vezu između navedenih osnovnih entiteta, te je on primer odnosnog entiteta i na slici je uokviren pomoću elipse.



Slika 2.1: E/R dijagram za malu studentsku bazu podataka

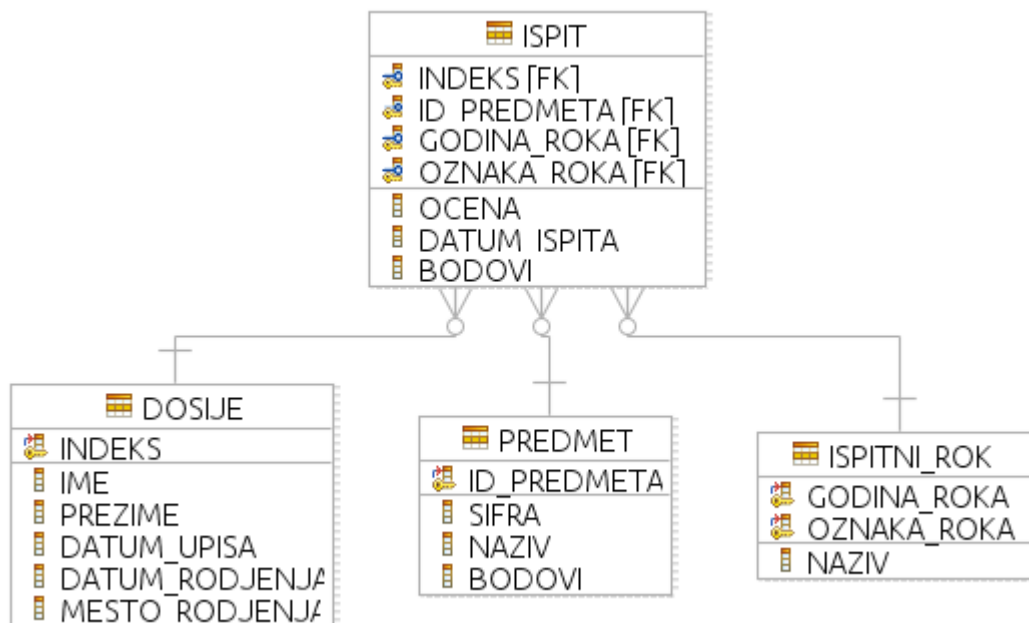
2.4 Informacije o studentima

Da bi fakultet mogao uspešno da obavlja svoju primarnu delatnost, tj. uspešno da obrazuje svoje polaznike, potrebno je da beleži i čuva informacije o njima i njihovim

uspesima tokom studiranja. Analizom bitnih podataka o studentima iz ugla fakulteta, moglo bi se zaključiti da je potrebno sačuvati informacije o sledećim entitetima:

- **Dosije.** Svojstva ovog entiteta su lični podaci o studentu koji se prikupljaju prilikom upisa studenta na fakultet, npr. ime, prezime, mesto rođenja, datum rođenja, indeks i datum upisa na fakultet. Svaki student ima jedinstveni broj indeksa.
- **Predmet.** Svojstva ovog entiteta su: identifikator predmeta, šifra predmeta, naziv i njegov broj espb bodova. Svaki predmet ima jedinstveni identifikator.
- **Ispitni_rok.** Svojstva ovog entiteta su: oznaka ispitnog roka (npr. jan, feb, apr), školska godina u kojoj je održan ispitni rok i naziv ispitnog roka. Svaki ispitni rok jedinstveno određuje oznaka i školska godina, npr. oznaka roka je jan, a školska godina 2021.
- **Ispit,** odnosni entitet koji povezuje prethodno opisane osnovne entitete. Za svaki ispit je nepohodno znati: indeks studenta koji je polagao ispit, identifikator predmeta koji je polagan, oznaku ispitnog roka i školsku godinu u kojoj je ispit polagan. Pored ovih svojstva, ispit može biti opisan i sa bodovima i ocenom koji su dobijeni na ispitu, kao i sa datumom polaganja ispita.

Dijagram opisanih entiteta u maloj studentskoj bazi sa njihovim svojstvima je prikazan na slici 2.2. Na dalje će mala studentska baza biti označena sa MSTUD.



Slika 2.2: Dijagram entiteta u MSTUD

2.5 Model podataka

Model podataka je apstraktna definicija objekata i operatora koji zajedno čine apstraktnu mašinu sa kojom korisnik komunicira. Objekti dopuštaju modeliranje strukture podataka, dok operatori koji se primanjuju nad objektima dopuštaju modeliranje ponašanja. Implementacija modela podataka je fizička realizacija na stvarnoj mašini komponenata apstraktne mašine koje zajedno čine model. Korisnik ne mora da zna detalje implementacije, ali je potrebno da poznaje model.

Relacione baze podataka se zasnivaju na formalnoj teoriji koja se zove relacioni model podataka. Bitne osobine relacionog modela su:

- podaci se predstavljaju kao n-torke (ili samo torke) u relacijama (ili kao redovi u tabelama).
- obezbeđeni su operatori koji se primenjuju nad torkama (redovima) relacije (tabele). Npr. operator za restrikciju izdvaja torke iz relacije koje zadovoljavaju zadati uslov.

2.6 Prednosti rada sa bazom podataka

Rad sa bazom podataka obezbeđuje mnoge prednosti, a neke od njih su:

- podaci se mogu deliti između postojećih aplikacija, a takođe se mogu praviti nove aplikacije koje će koristiti postojeću bazu podataka bez njene izmene.
- smanjuje se redundantnost podataka, odnosno podaci se ponavljaju samo u onoj meri koja je neophodna za efikasan rad.
- uklanjanjem redundantnosti podataka eliminiše se i nekonzistentnost, tj. ne dolazi do situacije da jedna kopija podatka bude promenjena, a druga ne. Npr. neka postoji osoba koja je student na fakultetu, a takođe je i zaposlena na fakultetu kao saradnik u nastavi. Ako baza podataka nije dobro napravljena, lični podaci o toj osobi će postojati na dva mesta, u delu koji se odnosi na nastavno osoblje i u delu koji se odnosi na studente. Ako se za tu osobu u nekom trenutku promeni vrednost koja se odnosi na lične podatke, npr. broj telefona, sa ovako napravljenom bazom podataka biće potrebno da se promene podaci na dva mesta. Ako je baza podataka dobro napravljena, lični podaci o osobi će biti samo na jednom mestu i oni će biti povezani sa delom koji se odnosi na nastavno osoblje, kao i sa delom koji se odnosi na studente, te će biti dovoljna samo jedna izmena koja se odnosi na lične podatke.
- postoji podrška za rad sa transakcijama. Transakcija je logička jedinica posla nad bazom podataka i sastoji se od jedne ili više operacija nad bazom podataka. Za transakciju je bitno da je obezbeđena osobina koja se zove atomičnost, tj. da su sve operacije u okviru jedne transakcije uspešno izvršene ili nijedna nije. Tipičan primer transakcije je prebacivanje novca sa računa X na račun Y. Operacije od kojih se sastoji takva transakcija su: 1) skini iznos N sa računa X i 2) dodaj iznos N na račun Y. Zbog atomičnosti transakcije, obe operacije moraju biti uspešno izvršene ili ni prva ni druga ne smeju biti izvršene.

- održavanje integriteta, tj. obezbeđivanje da su podaci u bazi podataka tačni. Ova prednost se postiže definisanjem pravila koja podaci u bazi podataka moraju da zadovolje u svakom trenutku. Npr. mogu se definisati pravila: a) školsku godinu ne može da upiše student koji je diplomirao ili b) na ispitu može da se dobije ocena koja je između 5 i 10.
- primena zaštite podataka. Moguće je definisati za različite korisnike koje podatke iz baze podataka mogu da vide, te osetljive podatke ne mogu svi korisnici da vide ili da ih menjaju. Npr. student iz baze podataka može da vidi samo one podatke koji se odnose na njega (lične podatke, upisane godine, upisane kurseve ...)
- balansiranje između konfliktnih zahteva. Podaci u bazi podataka moraju da budu strukturirani tako da budu optimalni za zadovoljavanje potreba svih korisnika, a ne pojedinačnih korisnika.
- primena standarda - od internacionalnih, industrijskih do onih koji su definisani u okviru organizacije kojoj pripada baza podataka.

2.7 Nezavisnost podataka

Nezavisnost podataka je otpornost aplikacije na promene fizičke reprezentacije podatka i pristupnih tehnika.

Aplikacije implementirane na starim sistemima su bile zavisne od fizičke reprezentacije podataka i tehnika za pristup. Npr. aplikacija je bila svesna postojanja indeksa (jednostavan opis indeksa: pokazivači na redove u tabeli koji su uređeni prema vrednosti neke kolone ili više njih, npr. broju poena predmeta) i koristila bi ga za pristup određenim redovima. To nije pogodno jer različite aplikacije mogu da zahtevaju različite poglede nad istim podacima, a i DBA mora da ima slobodu da promeni fizicku reprezentaciju ili pristupne tehnike radi performansi.

Pojmovi na koje se odnose promene koje bi DBA mogao da izvrši su:

- sačuvano polje (eng. stored field) je najmanja jedinica podataka koja može da se čuva;
- sačuvani slog (eng. stored record) je skup sačuvanih polja;
- sačuvana datoteka (eng. stored file) je skup svih trenutno postojećih pojava sačuvanih slogova istog tipa

Potrebno je obezbediti i da DBA može da promeni reprezentaciju podataka, a da je aplikacija uvek vidi na isti način. Npr. podatak je sačuvan kao decimalan broj, a aplikacija ga vidi kao nisku. Potrebno je obezbediti da taj podatak aplikacija uvek vidi kao nisku, čak i ako se način njegovog čuvanja promeni.

Neki aspekti sačuvanih reprezentacija koji mogu da budu predmet promena od strane DBA su:

- reprezentacija brojevanih podataka - npr. da li se koristi zapis u fiksnom ili pokretnom zarezu, koja je preciznost (broj cifara), ...
- reprezentacija znakovnih podataka

- jedinice za brožane podatke, npr. promena iz mm u cm

Potrebno je obezbediti mogućnost širenja baze podataka bez promene postojećih aplikacija i bez negativnog uticaja na postojeće aplikacije.

2.8 Neformalni pogled na relacioni model

Edgar Codd je predstavio relacioni model podataka 1970. godine u radu *A Relational Model of Data for Large Shared Data Banks* koji predstavlja jedan od najznačajnijih događaja u oblasti baza podataka. Proizvodi zasnovani na relacionom modelu podataka su počeli da se pojavljuju krajem 1970-tih i početkom 1980-tih.

Intuitivno, relacioni model predstavlja jedan način gledanja na podatke i sadrži pravila za predstavljanje podataka i pravila za rad sa tim podacima.

Relacioni model se često opisuje sa tri aspekta:

- aspekt strukture: svi podaci u bazi se korisniku prikazuju isključivo u obliku relacija;
- aspekt integriteta: tabele zadovoljavaju izvesna ograničenja (primarni i spoljašnji ključevi, ...);
- aspekt obrade: operatori koji su na raspolaganju korisnicima za obradu relacija su takvi da izvode relacije iz relacija.

2.8.1 Domen

Domen predstavlja skup svih mogućih vrednosti. Naziva se i tip podataka. U implementiranim sistemima postoje ugrađeni tipovi podataka (integer za cele brojeve, char za niske ...), ali i korisnik može definisati nove tipove koji se nazivaju korisnički definisani tipovi (npr. tip indeks, tip ime, ...)

SQL podržava sledeće relacione domene:

- brojevi (numbers)
- niske karaktera (character strings)
- niske bitova (bit strings)
- datumi (dates)
- vremena (times)
- kombinacija datuma i vremena (timestamps)
- intervali godina/mesec (year/month intervals)
- intervali dan/vreme (day/time intervals)
- ...

2.8.2 Relacije

Neka je dat skup od n tipova ili domena T_i ($i = 1, 2, \dots, n$), pri čemu ne moraju svi tipovi da budu međusobno različiti. R je **relacija** nad tim domenima ako se sastoji od dva dela,

zaglavlja i tela, gde važi:

- Zaglavlje je skup od n atributa oblika $A_i : T_i$ ($i = 1, 2, \dots, n$) gde su A_i imena atributa relacije R , a T_i odgovarajuća imena domena.
- Telo je skup od m n -torki (torki) t gde je t skup komponenti oblika $A_i : v_i$ u kojima je v_i vrednost iz domena T_i .

m se naziva kardinalnost, a n stepen (arnost) relacije R .

Bitne osobine relacije su:

- nema ponovljenih torki;
- torke su neuređene u relaciji, od vrha ka dnu;
- atributi su neuređeni u relaciji, sleva u desno;
- svaka torka sadrži tačno jednu vrednost za svaki atribut. Za relaciju koja zadovoljava ovu osobinu se kaže da je normalizovana, odnosno da je u prvoj normalnoj formi.

Relacija i tabela se obično koriste kao sinonimi, ali nisu jer

- tabela može da sadrži duplirane redove dok relacija ne može da sadrži duplirane torke;
- redovi u tabeli su uređeni u redosledu od vrha ka dnu, dok za relaciju to ne važi;
- kolone u tabeli su uređene u redosledu sleva u desno, dok za relaciju to ne važi;

Definicija relacije ima sledeći oblik:

relation {lista-atributa-razdvojenih-zarezima}

gde je atribut uređen par oblika

ime-atributa ime-domena

Na primer: dosije { Indeks INDEKS, Ime IME, Prezime PREZIME, God_rodjenja GOD_RODJENJA, Mesto_rodjenja MESTO_RODJENJA }

Relaciona baza podataka je skup relacija, a relaciona šema je opis strukture relacija.

2.8.3 Terminologija

Codd je pri formulisanju principa relacionih baza podataka uveo novu terminologiju. U tabeli 2.1 je dat pregled termina u relacionim bazama podataka i njihov opis.

Neka je data tabela dosije kao na slici 2.3. Tada je

- relacija kompletna tabela;
- torke je svaki red u tabeli;
- atributi su kolone Indeks, Ime, Prezime, Datum_rodjenja, Mesto_rodjenja, Datum_upisa;
- kardinalnost tabele je 7 (tabela ima 7 redova);
- stepen tabele je 6 jer ima 6 kolona;

Termin	Značenje
Domen	skup važećih vrednosti (tipova)
Relacija	matematički termin za tabelu
Torka	red u tabeli
Atribut	kolona u tabeli
Kardinalnost	broj torki
Stepen	broj atributa
Primarni ključ	atribut ili kombinacija atributa čije vrednosti jedinstveno identifikuju torku

Tabela 2.1: Termini u relacionim bazama podataka

- primarni ključ tabele je kolona Indeks;
- domeni u tabeli su skup mogućih brojeva indeksa (tip INDEKS), skup mogućih imena (tip IME), ..

INDEKS	IME	PREZIME	DATUM_RODZENJA	MESTO_RODZENJA	DATUM_UPISA
20140021	Milos	Peric	20.01.1995	Beograd	06.07.2014
20140022	Marijana	Savkovic	11.03.1995	Kraljevo	05.07.2014
20130023	Sanja	Terzic	09.11.1994	Beograd	04.07.2013
20130024	Nikola	Vukovic	17.09.1994	NULL	04.07.2013
20140025	Marijana	Savkovic	04.02.1995	Kraljevo	06.07.2014
20140026	Zorica	Miladinovic	08.10.1995	Vranje	06.07.2014
20130027	Milena	Stankovic	NULL	NULL	NULL

Slika 2.3: Tabela Dosije

2.8.4 Primeri operatora za obradu

Neki od operatora koji mogu da se primene nad relacijama su:

- projekcija koja izdvaja pojedinačne attribute iz relacije, npr. dosije [indeks, ime, prezime]
- restrikcija (selekcija) koja izdvaja pojedinačne torke iz relacije, npr. dosije where mesto_rodjenja='Beograd'

Relacioni operatori se primenjuju nad relacijama (tabelama), a kao rezultat se dobija relacija (tabela). Ova osobina se naziva **zatvorenje relacionih sistema**. Pošto je rezultat primene operatora istog tipa kao i njegov argument (relacija/tabela) mogu da se pišu **ugneždjeni relacioni izrazi**. Takođe, sve operacije ce primenjuju na ceo skup istovremeno, a ne samo na pojedinačnu torku (red). Rezultat operacije nije nikada pojedinačna torka (red)

već je uvek kompletna relacija (tabela) koja sadrži skup torki (redova). Relacija (tabela) može da sadrži samo jedan red ili da bude prazna.

2.8.5 Nedostajuće vrednosti

U svakodnevnoj praksi se često javlja problem nedostatka nekih podataka. Na primer, za neku osobu o kojoj se čuvaju podaci datum rođenja može biti nepoznat ili je ime supružnika nepoznato zato što osoba nije u braku. Potrebno je da u bazi podataka postoji indikator o nedostatku vrednosti, kao i da se na odgovarajući način vrši obrada takvih podataka. Najčešći pristup, koji je prihvaćen i u praksi, je korišćenje *nedostajuće vrednosti* (NULL), odnosno trovalentne (3VL) logike. U 3VL logici pored vrednosti tačno (eng. true, T) i netačno (eng. false, F), postoji i vrednost nepoznato (eng. unknown, U). U nastavku su prikazani rezultati operatora I, ILI, NE (eng. AND, OR, NOT) u 3VL logici.

AND	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

OR	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

NOT	
T	F
U	U
F	T

Tabela 2.2: Operatori I (AND), ILI (OR) i NE (NOT) u 3VL logici

U 3VL logici postoji i operator MOŽDA (MAYBE) koji je potreban da bi mogli da se napišu upiti koji uključuju i nedostajuće podatke, npr. zbog ovakvih upita: Prikazati sve zaposlene koji su možda bili, ali za koje nije sasvim pouzdano da su bili, programeri rođeni pre 25. januara 1991. godine sa platom manjom od 50000 na dan 30.09.2011.

MAYBE	
T	F
U	T
F	F

Tabela 2.3: Operator MOŽDA (MAYBE) u 3VL logici

2.8.6 Osnovna ograničenja u relacionim bazama podataka

Skup atributa (kolona) čije vrednosti mogu jedinstveno da odrede n-torku (red) u relaciji (tabeli) se naziva **primarni ključ** relacije (tabele). Na primer, atribut indeks je primarni ključ relacije dosije jer svaki student ima jedinstven indeks. U relaciji ispitni_rok primarni ključ čine atributi oznaka_roka i skolska_godina. Veza između dve relacije se može postaviti pomoću stranog ključa nad atributima jedne relacije, kojim označavamo da vrednosti u tim atributima mogu biti samo one koje se javljaju u drugoj (baznoj) relaciji sa kojom želimo

da povežemo relaciju u kojoj postavljamo strani ključ. Na primer, u relaciji ispit se može postaviti strani ključ nad atributom indeks koji referiše na relaciju dosije (u okviru koje je definisano da je atribut indeks primarni ključ). Na taj način se postavlja uslov da u relaciji ispit u atributu indeks može da bude samo vrednost koja se pojavljuje u atributu indeks relacije dosije, tj. da ispit ne može da prijavi/polaže student o kome nema podataka u relaciji dosije.

2.9 Formalni pogled na relacioni model

Relacioni model se sastoji od:

- otvorenog skupa skalarnih tipova (koji uključuje i tip logičkih vrednosti boolean);
- generatora relacionih tipova i njihove odgovarajuće interpretacije;
- mogućnosti definisanja relacionih promenljivih za generisane relacione tipove;
- operacije relacione dodele kojom se dodeljuju relacione vrednosti definisanim relacionim promenljivim;
- otvorenog skupa opštih relacionih operatora ("relaciona algebra") za izvođenje relacionih vrednosti iz drugih relacionih vrednosti.

3. Arhitektura sistema baza podataka

Arhitektura sistema baza podataka je apstraktni opis komponenti i njihovih interakcija.

3.1 ANSI/SPARC arhitektura

Arhitektura baza podataka koja je predložena od strane ANSI¹/SPARC² studijske grupe za sisteme za upravljanje podacima se sastoji od tri nivoa[6]:

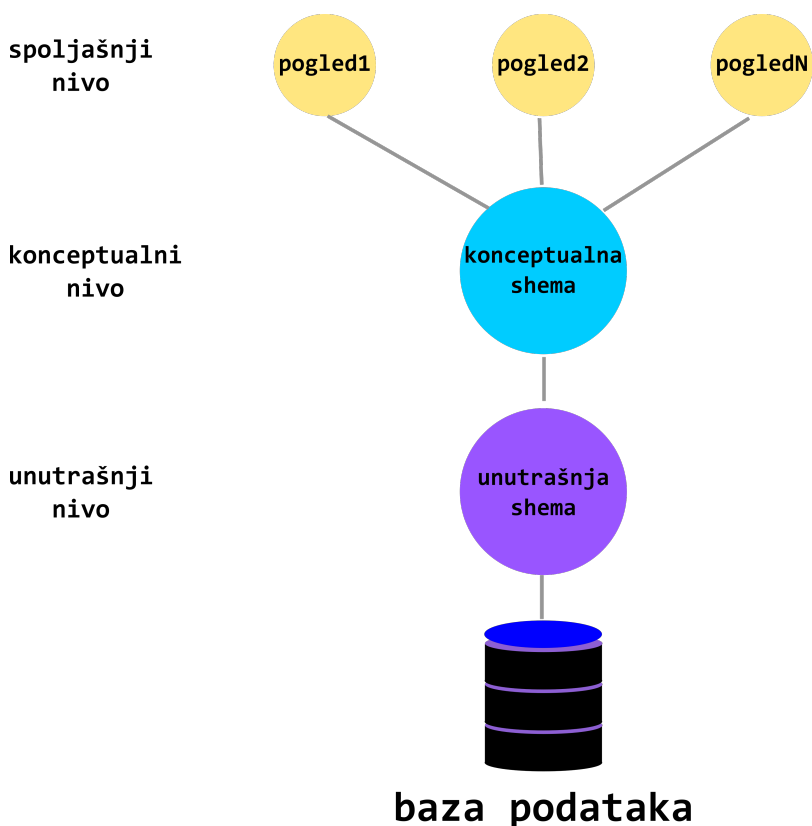
- unutrašnji nivo (interni nivo, fizički nivo) koji se odnosi na način kako se podaci čuvaju u sistemu;
- konceptualni nivo (zajednički logički izgled, model podataka) definiše kako se podaci uopšteno prikazuju korisniku;
- spoljašnji nivo (eksterni nivo, individualni korisnički izgled) koji se odnosi na način na koji podatke vidi individualni korisnik ili grupa korisnika.

Šema ANSI/SPARC arhitekture baze podataka je prikazana na slici 3.1

Postoji samo samo jedan *konceptualni izgled* podataka kojim se izražava apstraktna reprezentacija podataka, odnosno kako se podaci prikazuju korisniku. U relacionim bazama podataka, podaci se prikazuju preko relacija. Takođe, postoji samo jedan *unutrašnji izgled* podataka koji predstavlja način na koji se podaci čuvaju interno. Za razliku od konceptualnog i unutrašnjeg izgleda, postoji više *spoljašnjih izgleda* za podatke i svaki od njih predstavlja apstraktno predstavljanje jednog dela baze podataka koji je vidljiv određenom

¹ANSI - American National Standards Institute

²ANSI/SPARC - ANSI/System Planning and Requirements Committee



Slika 3.1: ANSI/SPARC arhitektura baza podataka

korisniku. Konceptualni i spoljašnji nivoi se definišu pomoću pojmova koji su orijentisani ka korisniku (slogovi, polja), a unutrašnji nivo se naziva i nivo implementacije jer se definiše pomoću pojmova orijentisanih ka mašinama (bitovi i bajtovi).

3.1.1 Spoljašnji nivo

Spoljašnji nivo je individualan korisnički nivo, a korisnik može da bude programer aplikacije ili krajnji korisnik. Programer aplikacije za izražavanje zahteva nad podacima ima na raspolaganju matični (eng. host) jezik u koji se ugrađuje jezik podataka (eng. data sublanguage, DSL) koji je kombinacija bar dva podjezika - jezika za definisanje objekata baze podataka (DDL) i jezika za rad i obradu objekata iz baze podataka (DML). U matične jezike spadaju npr. Java, C, C++, Python. Ako matični jezik ne može jasno da se odvoji od jezika podataka tada se za njih kaže da su čvrsto vezani, a ako mogu jasno i lako da se razdvoje tada se za njih kaže da su labavo vezani.

Krajnji korisnici koriste upitni jezik (eng. query language) za postavljanje zahteva. Primer upitnog jezika je SQL. Pojedinačnog korisnika obično interesuje samo jedan deo ukupne baze podataka, te on ima svoj spoljašnji izgled baze podataka koji se sastoji od spoljašnjih slogova, koji ne odgovaraju nužno sačuvanim slogovima u bazi podataka. Svaki

spoljašnji izgled je definisan preko spoljašnje šeme koja sadrži definicije svakog od različitih tipova slogova u spoljašnjem izgledu. Mora da postoji preslikavanje između konceptualne šeme i spoljašnje šeme.

3.1.2 Konceptualni nivo

Konceptualni nivo predstavlja informacioni kontekst celokupne baze podataka u obliku koji je nezavisan od načina na koji se podaci čuvaju na fizičkom nivou. Konceptualni slogovi različitih tipova se nalaze na konceptualnom nivou koji je definisan konceptualnom šemom. **Konceptualna šema uključuje definicije svakog od tipova konceptualnih slogova, kao i ograničenja koja se odnose na sigurnost i integritet.** Za njen zapis koristi se konceptualni DDL. Da bi se postigla nezavisnost podataka, definicije na konceptualnom DDL ne smeju ni na koji način da uključe fizičku reprezentaciju podataka ili pristupne tehnike.

3.1.3 Unutrašnji nivo

Unutrašnji nivo je reprezentacija baze podataka na niskom nivou. Sastoji se od pojava različitih tipova unutrašnjih slogova (ANSI/SPARC termin za sačuvani slog) čije su karakteristike definisane unutrašnjom šemom i zapisane pomoću unutrašnjeg DDL. Unutrašnji izgled je iznad fizičkog nivoa (ne radi sa adresama, blokovima podataka ili stranicama u memoriji). Umesto termina unutrašnji nivo ili izgled obično se koristi intuitivniji termin *sačuvana baza podataka*, a umesto termina unutrašnja šema termin *definicija sačuvanih struktura*. Zbog performansi, nekad se dopušta da neki aplikativni programi rade nad unutrašnjim izgledom baze podataka umesto nad spoljašnjim izgledom baze iako to nije preporučljivo zbog mogućeg narušavanja bezbednosti i integriteta.

3.1.4 Preslikavanje nivoa

Pored tri opisana nivoa, arhitektura sistema baza podataka obuhvata i preslikavanja između nivoa. Preslikavanje je opis povezanosti dva nivoa i postoji

- jedno konceptualno/unutrašnje preslikavanje;
- više spoljašnje/konceptualnih preslikavanja.

Konceptualno/unutrašnje preslikavanje precizira kako su konceptualni slogovi i polja predstavljeni na unutrašnjem nivou. Ukoliko se promeni unutrašnji izgled onda mora da se promeni i konceptualno/unutrašnje preslikavanje, kako bi se sačuvala nezavisnost podataka od promene fizičke strukture. Spoljašnje/konceptualno preslikavanje precizira vezu između određenog spoljašnjeg izgleda i konceptualnog izgleda. Ukoliko se promeni konceptualni izgled onda mora da se promeni i spoljašnje/konceptualno preslikavanje da bi se sačuvala nezavisnost podataka od promene logičke strukture.

3.2 Funkcije SUBP

Sistem za upravljanje bazama podataka (SUBP) je softver koji upravlja svim pristupima bazi podataka. Osnovni koraci u pristupu bazi podataka su:

- korisnik ispostavlja zahtev za pristupom koristeći jezik podataka (npr. SQL upit);
- SUBP prihvata zahtev i analizira ga;
- da bi odredio potrebne operacije SUBP proverava spoljašnju šemu korisnika, odgovarajuće spoljašnje/konceptualno preslikavanje, konceptualnu šemu, konceptualno/unutrašnje preslikavanje i definiciju sačuvane baze podataka;
- SUBP izvršava potrebne operacije (tj. zahtev korisnika) nad bazom podataka.

SUBP prvo mora da izdvoji tražene sačuvane slogove na osnovu kojih konstruiše tražene konceptualne slogove i onda na osnovu njih konstruiše tražene spoljašnje slogove.

U funkcije koje obavlja SUBP ubrajaju se i

- **Definisanje podataka**

SUBP mora biti sposoban da prihvati definiciju podataka u izvornom obliku i da je pretvori u odgovarajući oblik objekta, što postiže preko DDL procesora za svaki od DDL jezika.

- **Obrada podataka**

SUBP mora biti sposoban da obradi zahteve za izdvajanje, ažuriranje i brisanje postojećih podataka u bazi podataka, kao i za dodavanje novih podataka u bazu podataka. Ova funkciju se postiže pomoću DML procesora za DML.

- **Optimizacija izvršavanja upita**

Komponenta koja se naziva optimizator obrađuje zahteve napisane korišćenjem DML da bi odredila efikasan način za izvršavanje zahteva.

- **Obezbeđivanje zaštite i integriteta podataka**

SUBP mora da nadgleda zahteve korisnika i spreči bilo koji pokušaj narušavanja ograničenja vezanih za zaštitu i integritet podataka koje je definisao DBA.

- **Obezbeđivanje konkurentnog pristupa podacima i oporavak**

Deo SUBP koji se naziva upravljač transakcijama obezbeđuje transakcijama konkurentan pristup podacima i oporavak podataka u slučaju pada transakcije (prekida njenog izvršavanja).

- **Formiranje rečnika podataka**

SUBP mora da obezbedi formiranje rečnika podataka (repozitorijuma podataka, kataloga) koji sadrži metapodatke (podatke o podacima), odnosno sadrži informacije o definiciji svih objekata u sistemu (šeme, preslikavanja, ograničenja, ...). Rečnik takođe sadrži i opširnije informacije, na primer koji program koristi koji deo baze podataka, koji korisnik zahteva koji izveštaj ...

- **Obezbeđivanje što efikasnijeg rada**

- **Korisnički interfejs ka sistemu baza podataka**

Svrha SUBP je da omogući korisnički interfejs ka sistemu baza podataka na spoljašnjem nivou.

3.3 Klijent-server arhitektura

Svrha sistema baze podataka je da podrži razvoj i izvršavanje aplikacija koje rade sa bazom podataka. Takav sistem može da se posmatra kao da ima dve komponente:

- **server**
Server je u suštini SUBP sa obezbeđenim svim osnovnim funkcijama (definicija podataka, manipulacija podacima, ...)
- **klijent**
Klijenti su razne aplikacije (koje je napisao korisnik ili ih je obezbedio proizvođač SUBP) koje koriste isti interfejs za ispostavljanje zahteva serveru. Korisnički napisane aplikacije su regularne aplikacije napisane u nekom programskom jeziku koje koriste neki jezik podataka. Aplikacije koje obezbeđuje proizvođač SUBP (zovu se i alati) se koriste pri pravljenju i izvršavanju drugih aplikacija od strane korisnika bez korišćenja konvencionalnih programskih jezika. Ovim alatima pripadaju procesori upitnih jezika, alati za pisanje izveštaja, statistički paketi, alati za istraživanje podataka i vizuelizaciju...

3.4 Utility programi

Utility programi se prave sa ciljem da olakšaju DBA različite administratorske poslove. Mogu se podeliti na spoljašnje i unutrašnje. Spoljašnji *utility* programi su aplikacije specijalne namene koje rade na spoljašnjem nivou sistema. Unutrašnji *utility* programi rade na unutrašnjem nivou sistema te moraju biti deo servera i obezbeđuje ih proizvođač SUBP. Primeri *utility* programa su:

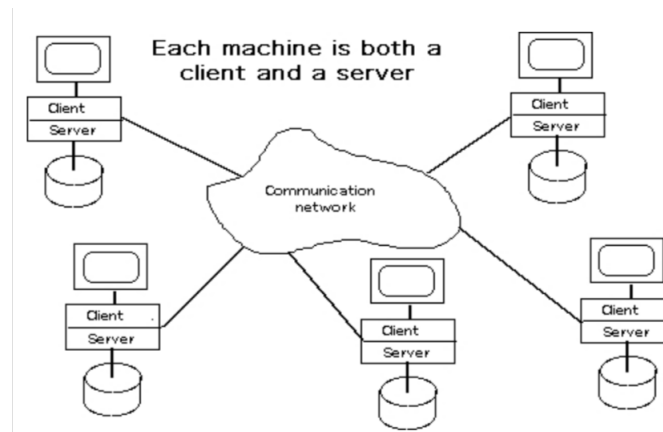
- LOAD rutina koja obezbeđuje pravljenje inicijalne verzije baze podataka iz regularnih datoteka
- UNLOAD/RELOAD rutine koje obezbeđuju čuvanje cele ili dela baze podataka i ponovno učitavanje podataka iz rezervne kopije
- programi za računanje raznih statistika
- programi za analizu statistika

3.5 Distribuirana obrada

Distribuirana obrada znači da se obrada podataka izvršava na različitim računarima koji su spojeni u mrežu. Postoji više tipova distributivne obrade, od kojih je najjenostavniji onaj u kome je SUPB (server) na jednoj mašini, a klijenti na drugoj. U većim preduzećima su podaci obično smešteni na dva ili više servera, te klijent nekad mora da pristupi podacima sa više mašina. Na slici 3.2 je dat primer povezanosti više mašina koje mogu da budu i klijent i server.

Pristup podacima na različitim mašinama se obično obezbeđuje na dva načina[1]:

- **Klijent može da pristupi većem broju servera, ali u jednom trenutku samo jednom.** Sa ovakvim pristupom ne mogu da se kombinuju podaci sa različitih servera kao odgovor na jedan zahtev. Takođe, klijent mora da zna na kom serveru se nalaze željeni podaci da bi mogao da ih dobije.
- **Klijent može istovremeno da pristupi većem broju servera, te podaci sa različitih mašina mogu da se kombinuju kao odgovor na jedan zahtev.** Klijent ne mora da zna na kom serveru se nalaze koji podaci, preciznije on ima doživljaj kao da njima upravlja jedan SUBP na jednom serveru. Ovaj način pristupa podacima koji se nalaze na različitim mašinama se naziva i distribuirani sistem baza podataka.



Slika 3.2: Distribuirana obrada podataka

Osnovna prednost relacionog modela u odnosu na hijerarhijski i mrežni je u tome što se u potpunosti oslanja na matematiku, konkretnije na relacionu algebru čime je omogućena računarska podrška, razvoj specifičnog softvera i obrada uz zagarantovanu konzistentnost podataka i rezultata.

4. Relaciona algebra

4.1 Manipulativni deo relacionog modela podataka

Manipulativni deo relacionog modela podataka čini skup operatora koji se koriste za rad sa relacijama. Definisana su dva formalna jezika za manipulisanje podacima koji se koriste za pravljenje relacionog izraza - relaciona algebra i relacioni račun. U relacionom izrazu napisanom na relacionoj algebri se zadaje redosled izvršavanja relacionih operatora nad relacijama, dok se u relacionom izrazu napisanom na relacionom računa opisuju osobine koja mora da zadovolji relacija koja se dobija kao rezultat. Upitni jezici u SUBP se zasnivaju na jednom od ovih formalizma, ili u njihovoj kombinaciji[6].

4.2 Relaciona algebra

Relaciona algebra se koristi za modeliranje relacija (objekata) smeštenih u relacionoj bazi podataka i za definisanje upita nad njima. Sastoji se od skupa operatora čiji su operandi, kao i rezultat, relacije. Prvu verziju relacione algebre definisao je Codd 1972. godine, i ona ima 8 operatora za:

- projekciju
- restrikciju
- Dekartov proizvod
- uniju
- presek
- razliku
- (prirodno) spajanje
- deljenje

Bitna osobina ovih operatora je da služe samo za čitanje (eng. read-only), odnosno da mogu samo da čitaju operande, tj. ne mogu da menjaju njihove vrednosti. Kasnije su drugi autori dodavali nove operatore relacione algebre.

4.3 Relacioni izraz algebre i relaciono zatvorenje

Relacioni izraz relacione algebre je kompozicija relacionih operatora i relacija nad kojima se primenjuju operatori. Rezultat relacionog izraza je uvek relacija. Osobina da su i argumenti i rezultat primene bilo kog relacionog operatora relacije se naziva **relaciono zatvorenje**. Zahvaljujući relacionom zatvorenju mogu da se pišu ugneždeni relacioni izrazi, tj. relacioni izrazi čiji su operandi takođe relacioni izrazi. Pošto je rezultat relacionog izraza relacija, potrebni je obezbediti da i novodobijene relacije imaju odgovarajuće zaglavlje (sa jedinstvenim nazivima atributa) i odgovarajuće telo.

4.4 Semantika originalne algebre

4.4.1 Restrikcija `where`

Neka relacija R ima attribute X, Y, \dots, Z i neka je u istinitosna funkcija čiji su parametri neki podskup od X, Y, \dots, Z . Tada je restrikcija relacije R prema u relacija sa istim zaglavljem kao relacija R i sa telom koje sadrži torke relacije R za koje u vraća tačnu vrednost. Sintaksa operatora za restrikciju je:

```
R where uslov
```

Restrikcija se naziva i horizontalno sečenje relacije (videti tabelu 4.1).

X_1	X_2	X_3	X_4	X_5

Tabela 4.1: Ilustracija primene restrikcije nad relacijom

■ **Primer 4.1** Izdvojiti podatke o studentima koji su rođeni u Beogradu.

```
dosije where mesto_rodjenja='Beograd'
```

■

4.4.2 Projekcija kao select ali izmedju uglastih zagrada

Neka relacija R ima bar attribute X, Y, \dots, Z . Projekcija relacije R na X, Y, \dots, Z je relacija čije

- zaglavlje je izvedeno iz relacije R uklanjanjem svih atributa koji se ne nalaze u skupu $\{X, Y, \dots, Z\}$;
- telo se sastoji od svih torki $\{X : x, Y : y, \dots, Z : z\}$ pri čemu se svaka torka javlja u R sa X vrednošću x , Y vrednošću y , ..., Z vrednošću z .

Sintaksa operatora za projekciju je:

```
R[X, Y, ..., Z]
```

Projekcija se naziva i vertikalno sečenje relacije (videti tabelu 4.2).

X_1	X_2	X_3	X_4	X_5

Tabela 4.2: Ilustracija primene projekcije nad relacijom

■ **Primer 4.2** Izdvojiti imena i prezimena studenata.

```
dosije[ime, prezime]
```

■

Umesto liste atributa koje je potrebno izdvojiti, može se navesti lista atributi koje relacija koja se dobija kao rezultat ne treba da sadrži:

```
R[all but X, Y, ..., Z]
```

Relacija koja se dobija kao rezultat u zaglavlju ima sve attribute relacije R bez atributa X, Y, \dots, Z .

■ **Primer 4.3** Izdvojiti imena i prezimena studenata koji su rođeni u Beogradu.

```
(dosije where mesto_rodjenja='Beograd')[ime, prezime]
```

PRVO IDE RESTRIKCIJA

PA ONDA PROJEKCIJA

■

4.4.3 Dekartov proizvod TIMES - svaki red sa svakim redom

Neka relacije R_1 i R_2 imaju sledeća zaglavlja

$$R_1 : X$$

$$R_2 : Y$$

Dekartov proizvod relacija R_1 i R_2 je uparivanje svake torke iz relacije R_1 sa svakom torkom iz relacije R_2 . Relacija koja se dobija kao rezultat u zaglavlju ima sve atribute iz relacije R_1 i sve atribute iz relacije R_2 i može se opisati sa $\{ \{X : x, Y : y\} | \{X : x\} \in R_1 \wedge \{Y : y\} \in R_2 \}$.

Sintaksa operatora za Dekartov proizvod je:

```
R1 times R2
```

Ilustracija primene Dekartovog proizvoda nad relacijama je prikazana u tabeli 4.3.

X_1	X_2		Y_1	Y_2		X_1	X_2	Y_1	Y_2
a	b	times	1	2	=	a	b	1	2
c	d		3	4		c	d	3	4

Tabela 4.3: Ilustracija primene Dekartovog proizvoda nad relacijama

■ **Primer 4.4** Izdvojiti podatke o studentima i predmetima.

```
dosije times predmet
```

■

■ **Primer 4.5** Izdvojiti podatke o studentima i ispitima.

```
dosije times ispit
```

U relaciji koja se dobija kao rezultat svaki student je uparen sa svakim ispitom, a ne samo sa svojim ispitima.

■

4.4.4 Prirodno spajanje JOIN

Neka relacije R_1 i R_2 imaju sledeća zaglavlja

$$R_1 : X_1, X_2, \dots, X_m, Y_1, Y_2, \dots, Y_n$$

,

$$R_2 : Y_1, Y_2, \dots, Y_n, Z_1, Z_2, \dots, Z_p$$

Neka su atributi $\{X_1, X_2, \dots, X_m\}$ označeni sa X , atributi $\{Y_1, Y_2, \dots, Y_n\}$ sa Y , a atributi $\{Z_1, Z_2, \dots, Z_p\}$ sa Z .

Prirodno spajanje relacija R_1 i R_2 je uparivanje svake torke iz relacije R_1 sa svakom torkom iz relacije R_2 koje imaju iste vrednosti u **svim** atributima koji imaju isto ime u relacijama i definisani su nad istim domenom. Relacija koja se dobija kao rezultat će u zaglavlju imati sve atribute iz relacije R_1 i sve atribute iz relacije R_2 , ali nema ponavljanja atributa sa istim imenom. Relacija koja se dobija kao rezultat primene prirodnog spajanja na relacije R_1 i R_2 se može opisati sa:

$$\{\{X : x, Y : y, Z : z\} | \{X : x, Y : y\} \in R_1 \wedge \{Y : y, Z : z\} \in R_2\}$$

Sintaksa operatora za Dekartov proizvod je:

`R1 join R2`

Ilustracija primene prirodnog spajanja nad relacijama je prikazana u tabeli 4.4.

X_1	X_2	join	Y_1	X_2	=	X_1	X_2	Y_1
a	b		1	b		a	b	1
c	d		3	f				

Tabela 4.4: Ilustracija primene prirodnog spajanja nad relacijama

■ **Primer 4.6** Izdvojiti podatke o studentima i njihovim ispitima.

dosije join ispit

■

■ **Primer 4.7** Izdvojiti podatke o predmetima i ispitima na kojima su polagani.

predmet join ispit

■

4.4.5 Slobodno spajanje

Pored prirodnog, postoji i slobodno (Θ) spajanje za torke čiji atributi zadovoljavaju uslov $X \Theta Y$. Ako je $\Theta = '='$ tada se ovo spajanje naziva jednakosno spajanje. Za slobodno spajanje ne postoji poseban operator, već se primenjuje Dekartov proizvod, a zatim restrikcija za zadavanje uslova spajanja.

4.4.6 Unija

Za relacije R_1 i R_2 koje su istog tipa, unija predstavlja relaciju koja je istog tipa kao i one, sa telom koje sadrži sve torke t koje se pojavljuju u relaciji R_1 ili R_2 .

Sintaksa operatora za uniju je:

$R_1 \text{ union } R_2$

Ilustracija primene unije nad relacijama je prikazana u tabeli 4.5.

X_1	X_2			X_1	X_2			X_1	X_2
a	b	union		c	d	=		a	b
c	d			e	f			c	d
								e	f

Tabela 4.5: Ilustracija primene unije nad relacijama

■ **Primer 4.8** Izdvojiti podatke o ispitima koji su održani u januarskim ispitnim rokovima ili na kojima je dobijena ocena 10.

```
ispit where oznaka_roka='jan'
union
ispit where ocena=10
```

■

4.4.7 Presek

Za relacije R_1 i R_2 koje su istog tipa, presek predstavlja relaciju koja je istog tipa kao i one, sa telom koje sadrži sve torke t koje se pojavljuju i u relaciji R_1 i u relaciji R_2 .

Sintaksa operatora za presek je:

$R_1 \text{ intersect } R_2$

Ilustracija primene unije nad relacijama je prikazana u tabeli 4.6.

■ **Primer 4.9** Izdvojiti podatke o ispitima koji su održani u januarskim ispitnim rokovima i na kojima je dobijena ocena 10.

```
ispit where oznaka_roka='jan'
intersect
ispit where ocena=10
```

X_1	X_2	intersect	X_1	X_2	=	X_1	X_2
a	b		c	d		c	d
c	d		e	f			

Tabela 4.6: Ilustracija primene preseka nad relacijama

■

4.4.8 Razlika

Za relacije R_1 i R_2 koje su istog tipa, razlika relacija R_1 i R_2 predstavlja relaciju koja je istog tipa kao i one, sa telom koje sadrži sve torke t koje se pojavljuju u relaciji R_1 , ali se ne pojavljuju u relaciji R_2 .

Sintaksa operatora za razliku je:

```
R1 minus R2
```

Ilustracija primene razlike nad relacijama je prikazana u tabeli 4.7.

X_1	X_2	minus	X_1	X_2	=	X_1	X_2
a	b		c	d		a	b
c	d		e	f			

Tabela 4.7: Ilustracija primene razlike nad relacijama

■ **Primer 4.10** Izdvojiti podatke o ispitima koji su održani u januarskim ispitnim rokovima na kojima nije dobijena ocena 10.

```
ispit where oznaka_roka='jan'
minus
ispit where ocena=10
```

■

Primetiti da važe sledeće jednakosti:

- $R \text{ where } u_1 \text{ or } u_2 = (R \text{ where } u_1) \text{ union } (R \text{ where } u_2)$
- $R \text{ where } u_1 \text{ and } u_2 = (R \text{ where } u_1) \text{ intersect } (R \text{ where } u_2)$

- $R \text{ where not } (u) = R \text{ minus } (R \text{ where } u)$

4.4.9 Deljenje

Neka relacije R_1 i R_2 imaju sledeća zaglavlja

$R_1 : X, Y$

$R_2 : Y$

Rezultat deljenja relacije R_1 sa relacijom R_2 je relacija sa zaglavljem $\{X\}$ i telom koje sadrži sve torke $\{X : x\}$ koje se u relaciji R_1 pojavljuju uparene sa svim torkama relacije $R_2[Y]$.

Sintaksa operatora za deljenje je:

R1 divideby R2

Ilustracija primene deljenja nad relacijama je prikazana u tabelama 4.8 i 4.9.

X_1	X_2	divideby	X_2	=	X_1
a	1		1		a
a	2		2		c
b	1				
c	1				
c	2				
d	2				

Tabela 4.8: Ilustracija primene deljenja nad relacijama

X_1	X_2	X_3	X_4	divideby	X_3	X_4	=	X_1	X_2
a	b	1	2		1	2		a	b
c	d	1	2		3	4		g	h
a	b	3	4						
e	f	3	4						
g	h	1	2						
g	h	3	4						

Tabela 4.9: Ilustracija primene deljenja nad relacijama

■ **Primer 4.11** Pronaći studenta koji je polagao sve predmete.

```
ispit[indeks, id_predmeta]
divideby
predmet[id_predmeta]
```

■ **Primer 4.12** Pronaći studenta koji je polagao sve predmete od 6 espb u jednom ispitnom roku.

```
ispit[indeks, oznaka_roka, godina_roka, id_predmeta]
divideby
(predmet where bodovi=6)[id_predmeta]
```

4.5 Dodatni operatori

Osim operatora relacione algebre koje je uveo Codd, relaciona algebra se može proširiti sa dodatnim operatorima. Neki od njih su:

- operator za definisanje aliasa koji se koristi kada je potrebno u relacionom izrazu koristiti istu relaciju više puta u različitim kontekstima

```
define alias novo-ime for ime-relacije
```

- operator za preimenovanje atributa koji se obično koristi da bi se obezbedilo da svaki atribut u relaciji ima jedinstveno ime

```
ime-relacije rename X as novo-ime-atributa
```

■ **Primer 4.13** Izdvojiti podatke o predmetima i njihovim ispitima.

```
(predmet rename bodovi as espb) join ispit
```

■ **Primer 4.14** Izdvojiti parove studenata koji su rođeni u istom mestu. Izdvojiti indekse studenata.

```
define alias d1 for dosije
define alias d2 for dosije
((d1 times d2)
where d1.mesto_rodjenja=d2.mesto_rodjenja
and d1.indeks<d2.indeks)
[d1.indeks, d2.indeks]
```

ili

```
define alias d1 for dosije
((d1 times dosije)
where d1.mesto_rodjenja=dosije.mesto_rodjenja and d1.indeks<dosije.indeks)
[d1.indeks, dosije.indeks]
```

■

4.6 Minimalni skup operatora

Osam operatora koje je Codd definisao ne čine minimalan skup operatora, jer je neke od njih moguće definisati koristeći ostale. Minimalan skup operatora čine:

- restrikcija
- projekcija
- proizvod
- unija
- razlika

Na primer, presek može da se definiše preko unije i razlike.

4.7 Svrha relacione algebre

Iako su prikazani samo operatori za dohvatanje podataka, osnovna svrha relacione algebre je da omogući pisanje relacionih izraza koji omogućavaju:

- definisanje prostora za dohvatanje podataka;
- definisanje prostora za ažuriranje podataka (unos novih, menjanje i brisanje postojećih);
- definisanje pravila integriteta, odnosno ograničenja koje baza podataka mora da zadovolji;
- definisanje izvedenih relacija;
- definisanje pravila zaštite;
- ...

4.8 Relaciona kompletnost

Za upitni jezik je potrebno da je relaciono kompletan, odnosno da je moćan isto kao i algebra, tj. da bilo koja relacija predstavljiva u relacionoj algebri može da se predstavi i u upitnom jeziku. Upitni jezik SQL je relaciono kompletan jer postoje SQL izrazi za svaki od 5 primitivnih operatora relacione algebre (videti 4.10).

Relaciona algebra	SQL
A WHERE uslov	SELECT * FROM A WHERE uslov
A[x, y, ..., z]	SELECT DISTINCT x, y, ..., z FROM A
A TIMES B	A CROSS JOIN B
	SELECT * FROM A
A UNION B	UNION
	SELECT * FROM B
	SELECT * FROM A
A MINUS B	EXCEPT
	SELECT * FROM B
A RENAME x AS y	SELECT x AS y FROM A

Tabela 4.10: SQL izrazi minimalnog skupa operatora relacione algebre

4.9 Algebarski zakoni

Za operatore relacione algebre važe sledeći algebarski zakoni:

- **zakon asocijacije:**
 - (A UNION B) UNION C = A UNION (B UNION C)
 - (A INTERSECT B) INTERSECT C = A INTERSECT (B INTERSECT C)
 - (A TIMES B) TIMES C = A TIMES (B TIMES C)
 - (A JOIN B) JOIN C = A JOIN (B JOIN C)
- **zakon komutacije:**
 - A UNION B = B UNION A
 - A INTERSECT B = B INTERSECT A
 - A TIMES B = B TIMES A
 - A JOIN B = B JOIN A

4.10 Prioritet operatora

U relacionom izrazu napisanom na relacionoj algebri važe sledeći prioriteti, od višeg ka nižem, među operatorima:

- unarni operatori (restrikcija, projekcija)
- times, join
- intersect, divideby
- union, minus

5. Relacioni račun

Relaciona račun je drugi formalizam kojim se može opisati manipulativni deo relacionog modela. Ako se posmatra deo relacionog modela podataka za obradu podataka, relacioni račun je logički ekvivalent relacione algebre, tj. svaki izraz koji može da se napiše na relacionoj algebri može da se napiše i na relacionom računu. Relacioni račun je zasnovan na predikatskom računu. Edgar Codd je definisao dve varijante relacionog računa:

- relaciona račun n-torki;
- relaciona račun domena.

5.1 Relacioni račun n-torki

Osnovna osobina obe varijante relacionog računa je **promenljiva**. Promenljiva u relacionom računu n-torki se naziva *promenljiva n-torki*. Promenljiva n-torki je promenljiva čije vrednosti mogu biti torke iz relacije nad kojom je deklarisan. Sintaksa za deklarisanje promenljive je

```
range of ime-promenljive is relacija
```

Na primer, sa

```
range of d is dosije  
range of i is ispit
```

deklariše se promenljiva n-torki d čiji su opseg vrednosti torke iz relacije dosije. Svaka

n-torna promenljiva koja se koristi u okviru izraza relacionog računa n-torki mora biti prvo deklarirana. Relacioni izraz u relacionom računu n-torki ima oblik

```
promenljiva.ime-atributa [, promenljiva.ime-atributa]*
[where uslovni-izraz]?
```

Na početku se navodi lista atributa promenljivih iz kojih je potrebno izdvojiti podatke. Navođenje liste atributa odgovara projekciji u relacionoj algebri.

■ **Primer 5.1** Izdvojiti imena i prezimena studenata.

```
range of d is dosije
d.ime, d.prezime
```

■

Kada je potrebno izdvojiti sve attribute iz neke relacije, umesto liste atributa može se navesti promenljiva.*.

■ **Primer 5.2** Izdvojiti podatke o studentima.

```
range of d is dosije
d.*
```

■

U okviru klauzule where opisuju se svojstva, odnosno uslovi koji moraju da važe za torke koje će se naći u rezultatu. U okviru uslovnog izraza mogu se navoditi poređenja u obliku $a\Theta b$ pri čemu su a i b vrednosti nekog od atributa promenljivih koje su navedene u okviru liste atributa (napomena: ti atributi ne moraju biti u listi atributa), konstante ili izrazi, a Θ je operator =, <, >, >=, <= ili <>. Više uslova je moguće spojiti sa and (logičko i) ili or (logičko ili), a moguće je vršiti i negaciju uslova sa not. U rezultatu će se naći torke koje zadovoljavaju zadati uslovni izraz.

■ **Primer 5.3** Izdvojiti imena i prezimena studenata koji su rođeni u Beogradu.

```
range of d is dosije
d.ime, d.prezime
where mesto_rodjenja='Beograd'
```

■

■ **Primer 5.4** Izdvojiti imena i prezimena studenata koji su rođeni u Beogradu posle 1.1.1995.

```
range of d is dosije
d.ime, d.prezime
where mesto_rodjenja='Beograd' and datum_rodjenja > '1.1.1995'
```

■

Navođenje atributa iz dve ili više promenljivih u listi atributa odgovara Dekartovom proizvodu relacija nad kojima su definisane promenljive iz liste atributa.

■ **Primer 5.5** Izdvojiti indekse studenata i nazive predmeta.

```
range of d is dosije  
range of p is predmet  
d.indeks, p.naziv
```

■

Spajanje relacija označava se navođenjem atributa tih promenljivih u listi promenljivih i uslova spajanja.

■ **Primer 5.6** Izdvojiti imena i prezimena studenta i identifikatore predmeta koje su položili.

```
range of d is dosije  
range of i is ispit  
d.ime, d.prezime, i.id_predmeta  
where d.indeks=i.indeks and ocena > 5
```

■

U relacionom računu postoje dva kvantifikatora - egzistencijalni (exists) i univerzalni (forall). Pomoću kvantifikatora se mogu uvesti promenljive koje se ne nalaze u listi atributa i mogu se postaviti uslovi nad njihovim torkama.

Uslovni izraz

```
exists promenljiva(uslov)
```

izračunava tačno ako za neku torku promenljive važi zadati uslov, a netačno inače.

Uslovni izraz

```
forall promenljiva(uslov)
```

izračunava tačno ako za sve torke promenljive važi zadati uslov, a netačno inače.

■ **Primer 5.7** Izdvojiti ime i prezime studenta koji ima položen neki ispit.

```
range of d is dosije  
range of i is ispit  
d.ime, d.prezime
```

```
where exists i (d.indeks=i.indeks and ocena > 5)
```

■ **Primer 5.8** Izdvojiti ime i prezime studenta koji je položio sve predmete.

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where forall p ( exists i (p.id_predmeta=i.id_predmeta
                        and d.indeks=i.indeks
                        and ocena > 5))
```

Promenljive koje se navode u listi atributa se nazivaju *slobodne promenljive* i njihov domen važenja je ceo relacioni izraz (na primer promenljiva d u prethodnom primeru). Promenljive koje su vezane kvantifikatorom se nazivaju *vezane promenljive* i njihov domen važenja je samo uslovni izraz naveden uz kvantifikator (na primer promenljive p i i u prethodnom primeru). U implementaciji nije potrebno podržati oba kvantifikatora jer se kvantifikator forall može definisati preko operatora exists, jer važi

```
forall X (uslov) = not exists X ( not uslov)
```

gde je X promenljiva. Te bi rešenje prethodnog primera bilo i

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where not exists p ( not exists i (
                        p.id_predmeta=i.id_predmeta and
                        d.indeks=i.indeks and
                        ocena > 5))
```

U relacionom računu postoji i operator implikacije sa sledećom sintaksom

```
if uslov1 then uslov2
```

koji izračunava netačno jedino za vrednosti promenljivih za koje važi uslov1, ali ne važi uslov2.

■ **Primer 5.9** Izdvojiti ime i prezime studenta koji je položio sve predmete od 6 espb.

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where forall p ( if p.bodovi=6 then
                  exists i ( p.id_predmeta=i.id_predmeta
                             and d.indeks=i.indeks
                             and ocena > 5))
```

■

Ovaj zadatak bi mogao da se reši i bez operatora implikacije jer se uslovni izraz

```
if uslov1 then uslov2
```

može zapisati kao

```
not uslov1 or uslov2
```

Rešenje prethodnog primera bi bilo i

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where forall p ( not p.bodovi=6 or
                  exists i ( p.id_predmeta=i.id_predmeta
                             and d.indeks=i.indeks
                             and ocena > 5))
```

Preciznije, neka je iz prethodnog primera

- *uslov1*: p.bodovi=6
- *uslov2*: exists i (p.id_predmeta=i.id_predmeta and d.indeks=i.indeks and ocena > 5)

tada uslovni izraz zadatka možemo napisati na jedan od sledećih načina:

1. forall p (if uslov1 then uslov2)
2. forall p (not uslov1 or uslov2)
3. not exists p (not (not uslov1 or uslov2))
4. not exists p (uslov1 and not uslov2)

Zapis uslovnog izraza pod brojem 4 je dobijen primenom De Morganovog zakona nad brojem 3. Korišćenjem zapisa uslovnog izraza pod brojem 4, rešenje zadatka bi bilo i

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where not exists p ( p.bodovi=6 or
                    not exists i ( p.id_predmeta=i.id_predmeta
                                and d.indeks=i.indeks
                                and ocena > 5))
```

Uslovni izraz u relacionom izrazu računa može biti definisan i na sledeći način:

```
uslovni-izraz :=
poređenje |
uslovni-izraz AND uslovni-izraz |
uslovni-izraz OR uslovni-izraz |
NOT uslovni-izraz |
IF uslovni-izraz THEN uslovni-izraz |
EXISTS promenljiva (uslovni-izraz) |
FORALL promenljiva (uslovni-izraz)
```

Ovde je prikazana sintaksa zadavanja izraza relacionog računa n-torki koja je bliska sintaksi upitnih jezika zasnovanih na relacionom računu. Pored nje postoje i druge sintakse.

5.2 Relacioni račun domena

U relacionom računu domena promenljiva se deklariše nad domenom atributa (domenska promenljiva) umesto nad relacijom. Relacioni račun domena u okviru uslovnog izraza podržava i oblik poređenja koji se naziva uslov pripadnosti. Uslov pripadnosti ima oblik

$R (lista\ parova)$

gde je R ime relacije, a svaki par je oblika Ax , gde je A ime atributa relacije R , a x je domenska promenljiva nad domenom atributa A ili konstanta iz domena atributa A . Uslov pripadnosti je tačan ako postoji torka u relaciji R takva da je za svaki par Ax iz liste parova poređenje $A = x$ tačno za tu torku.

Umesto liste atributa u relacionom računu domena se navodi lista domenskih promenljivih.

Neka je za naredne primere promenljiva

- **imex** definisana nad domenom atributa ime u relaciji dosije;
- **indeksx** nad domenom atributa indeks u relaciji dosije;
- **idx** nad domenom atributa id_predmeta u relaciji predmet.

■ **Primer 5.10** Naći imena studenata iz Beograda.

```
imex
where dosije( ime imex, mesto_rodjenja 'Beograd')
```

■

■ **Primer 5.11** Pronaći imena i indekse studenata koji su polagali Analizu 1.

```
imex, indeksx
where exists idx ( predmet(id_predmeta idx, naziv 'Analiza 1') and
                  ispit(id_predmeta idx, indeks indeksx) and
                  dosije(indeks indeksx, ime imex))
```

■

5.3 Relaciona algebra ili relacioni račun

Edgar Codd je u svom radu dao **algoritam redukcije** kojim je pokazao da je relaciona algebra moćna bar koliko i relacioni račun. Pokazao je da se proizvoljni relacioni izraz računa može prevesti na semantički ekvivalentan relacioni izraz algebre [1]. Za upitni jezik se kaže da je *relaciono kompletan* ako je moćan bar koliko i relacioni račun, odnosno ako se bilo koja relacija koja se dobija kao rezultat relacionog izraza na relacionom računu može dobiti i izrazom na tom upitnom jeziku. Kako je Codd svojim algoritmom redukcije pokazao da je relaciona algebra relaciono kompletna, onda se može reći da je upitni jezik relaciono kompletan ako je moćan bar koliko i relaciona algebra. Relaciona kompletnost može da se posmatra kao osnovna mera izražajnosti jezika za baze podataka. Neki upitni jezici su više zasnovani na algebri, a neki na računu. SQL ima osobine i relacione algebre i relacionog računa.

6. Integritet i sigurnost

Pojam *integritet* se u kontekstu relacionih baza podataka odnosi na preciznost, punovažnost i korektnost podataka u bazi. Održavanje integriteta podataka je od najveće važnosti za RSUBP, zbog čega se u sistemu definišu pravila (tzv. ograničenja integriteta) koja se primenjuju na podatke.

Intuitivno, ograničenje integriteta je logički izraz pridružen bazi podataka za koji se zahteva da njegovo izračunavanje uvek daje vrednost **tačno**. Definisana ograničenja se proveravaju pri pravljenju objekata u bazi ili menjanju njihovog sadržaja.

U relacionoj bazi podataka uvek mora da važi **zlatno pravilo** čija prva verzija glasi: *Nijednoj operaciji ažuriranja nije dozvoljeno da ostavi bilo koju relaciju u stanju koje narušava bilo koje od ograničenja te relacije* [1]. Proširena, opštija verzija zlatnog pravila je: *Nijednoj operaciji ažuriranja nije dozvoljeno da ostavi bilo koju bazu podataka u stanju u kome se neko od ograničenja baze podataka izračunava kao netačno*. Posledica primene zlatnog pravila je da pre bilo kakvog stvarnog ažuriranja proverava važenje ograničenja.

6.1 Klasifikacija ograničenja integriteta

Jedna klasifikacija ograničenja integriteta je **prema tipu**. Prema ovoj klasifikaciji, ograničenje može biti:

- ograničenje baze podataka;
- ograničenje relacije;
- ograničenje atributa;
- ograničenje tipa.

Ograničenje tipa je definicija skupova vrednosti koji čine određeni tip. Npr. za tip težina

se može definisati da su moguće vrednosti realni brojevi između 0 i 250.

Ograničenje atributa je ograničenje na skup dozvoljenih vrednosti datog atributa date relacije i predstavljaju deo definicije atributa.

Ograničenje relacije predstavlja ograničenje na vrednosti pojedinačne relacije koje se proverava pri ažuriranju te relacije. Primer ograničenja relacije bi bio:

```
constraint rel1
  if not ( is_empty ( predmet ) ) then
    count ( predmet
      where sifra= sifra ('R270')) > 0
  end if
```

koje može da se izrazi na sledeći način: ako uopšte postoji neki predmet tada bar jedan od njih mora da ima šifru R270.

Ograničenja baze podataka su ograničenja koja se odnose na vrednosti koje je dozvoljeno čuvati u bazi i odnosi se na dve ili više različitih relacija. Primer ograničenja baze podataka bi bio:

```
constraint baza1
  forall dosije d forall ispit i
    is_empty (( d join i )
      where i.indeks > 20150000
      and i.indeks = d.indeks
      and godina_roka=godina_roka(2015))
```

koje može da se izrazi na sledeći način: nijedan student upisan na studije 2015. godine ne može da polaže uspit u 2015. godini.

Posebnu grupu ograničenja relacije ili baze podatka predstavljaju ograničenja prelaza pomoću kojih se zadaju mogući prelazi iz jedne u drugu vrednost. Na primer, ako baza podataka sadrži podatke o osobama tada su važeća sledeća ograničenja prelaza:

- nije dozvoljeno venčanje već venčanih osoba;
- dozvoljeno je venčati se sa razvedenom osobom;
- osobe koje više nisu žive ne mogu da primaju platu (penziju, ...);
- ...

6.2 Ključevi

Kandidat za ključ relacije $R(X_1, X_2, \dots, X_n)$ predstavlja podskup atributa X te relacije, ako važi:

- pravilo jedinstvenosti: ne postoje dve torke u relaciji R koje imaju iste vrednosti za X ,

- **pravilo minimalnosti:** ne postoji pravi podskup skupa atributa X koji zadovoljava pravilo jedinstvenosti.

Svaka relacija ima bar jednog kandidata za ključ - skup svih atributa ili neki njegov pravi podskup.

Postoji nekoliko vrsta ključeva relacije:

- *primarni ključ* koji predstavlja jedan od kandidata za ključ;
- *alternativni ključevi* su svi kandidati za ključ sem primarnog ključa;
- *spoljašnji (strani) ključ* je skup atributa jedne relacije R_2 čije vrednosti treba da odgovaraju vrednostima nekog kandidata za ključ neke relacije R_1 ;
- *superključ* je nadskup kandidata za ključ koji poseduje jedinstvenost, ali ne i minimalnost.

Primeri ključeva u relacijama male studentske baze podataka mstud su:

- u relaciji dosije primarni ključ je atribut indeks;
- u relaciji ispitni_rok primarni ključ je skup atributa (godina_roka, oznaka_roka);
- u relaciji ispit
 - primarni ključ je skup atributa (indeks, id_predmeta, godina_roka, oznaka_roka);
 - strani ključ na relaciju ispitni_rok je skup atributa (godina_roka, oznaka_roka), kojim je zadato ograničenje da ispit može da se polaže samo u ispitnim rokovima o kojima postoje podaci u tabeli ispitni_rok;
 - strani ključ na relaciju dosije je atribut indeks, kojim je zadato ograničenje da ispit može da polaže samo student o kome postoje podaci u relaciji dosije.

Strani ključevi obezbeđuju referencijalni integritet. Osnovna ideja očuvanja referencijalnog integriteta je da sve vrednosti u tabelama treba da budu usaglašene. Na primer, student ne može da polaže ispit u ispitnom roku o kome nema podataka u tabeli ispitni_rok. Relacija koja sadrži primarni ključ (kandidat za ključ) se naziva *roditelj relacija*, a relacija koja sadrži spoljašnji ključ koji referiše na roditelj relaciju se naziva *dete relacija*. Pravilo referencijalnog integriteta je *baza podataka ne sme da sadrži neuparene vrednosti spoljašnjih ključeva*. Neupareni spoljašnji ključevi bi bile vrednosti koji bi se pojavile u atributima koje čine spoljašnji ključ u dete relaciji, a ne postoje u ključu roditelj relacije (na koju referiše strani ključ).

Roditelj relacija i dete relacija ne moraju da budu različite relacije. Npr, u relaciji *zaposleni* veza između zaposlenog i nadređenog bi mogla da se opiše pomoću stranog ključa. Takođe, moguće je definisati **referencijalni ciklus**

$$T_n \rightarrow T_{n-1} \rightarrow T_{n-2} \rightarrow \dots \rightarrow T_1 \rightarrow T_n$$

u okviru koga relacija T_n referiše na relaciju T_{n-1} , relacija T_{n-1} referiše na relaciju T_{n-2} , ..., relacija T_2 referiše na relaciju T_1 , a relacija T_1 referiše na relaciju T_n .

U upitnom jeziku SQL, pri definiciji tabele mogu se definisati

- alternativni ključevi pomoću opcije `unique`;
- primarni ključ pomoću opcije `primary key`;

- spoljašnji ključevi pomoću opcije `foreign key`;
- ograničenja pomoću opcije `check` (uslovni-izraz).

6.3 Pojam sigurnosti podataka

(neovlasćenih)

Pojam sigurnosti podataka se odnosi na zaštitu podataka od neautorizovanih korisnika, odnosno zaštitu protiv neautorizovanog pristupa, promene ili uništenja. Sa druge strane, pojam integriteta bi mogao da se opiše kao zaštita podataka protiv autorizovanih korisnika, odnosno obezbeđivanje ispravnosti i korektnosti podataka.

Postoje i izvesne sličnosti između sigurnosti i integriteta. Sistem mora da bude svestan izvesnih ograničenja koje korisnici ne smeju da prekrše. Ograničenja moraju da budu zadata (od strane DBA) u nekom jeziku i evidentirana u sistemskom katalogu (rečniku podataka). SUBP mora da vrši nadzor nad operacijama korisnika.

Mogu se osiguravati različiti elementi, pa tako jedinica podataka na koja se osigurava može biti:

- baza podataka
- relacija
- pojedinačna torka
- vrednost atributa
- alias
- šema
- indeks
- ...

6.4 Ostali aspekti sigurnosti

Kompletan sistem treba da bude zaštićen. Ne pretpostavljati da je sistem zaštite savršen. Potrebno je voditi evidenciju o prijavljivanju na bazu i redovno imati uvid u sva dešavanja na bazi.

7. OLAP

7.1 Uvod

OLAP (eng. online analytical processing) se može definisati kao "interaktivni proces formiranja, upravljanja, analiziranja i prikaza podataka". To je tehnologija koja koristi multidimenzionalne strukture za omogućavanje brzog pristupa podacima za analizu.

Omogućava poslovnim korisnicima brz, konzistentan, interaktivan pristup širokoj paleti mogućih pogleda na podatke. On neobrađene podatke transformiše u realnoj dimenziji onako kako ih vidi i razume korisnik.

Poslovni korisnici uglavnom na podatke gledaju kroz različite scenarije, kojih može biti više u procesu analize, tako da je uobičajeno da se organizuju po nečemu, na primer, po godini, regionu, gradu i drugo.

OLAP organizuje podatke preko hiperkocke, tzv. CUBE, što je skup podataka koji su organizovani i sumirani u multidimenzionu strukturu definisanu kao skup dimenzija i mera.

Dimenzija je organizovana hijerarhijski i odgovara kategorijama koje opisuje podatke u hiperkocki. Dimenzija obično označava skup sličnih članova. Na primer, u vremenskoj dimenziji se može računati po godini, po mesecu, po danu. U geografskoj dimenziji se može računati po državi, regionu, gradu, i drugo.

Mera je vrednosti u hiperkocki i uglavnom se koristi numerički. Mere su centralne vrednosti koje su agregirane.

7.2 Arhitektura

Tradicionalni RDBMS nisu primarno namenjeni da efikasno podrže OLAP upite i baze podataka koje se mere gigabajtima ili terabajtima.

Postoje specijalizovani SQL serveri u kojima postoji bolja podrška OLAP upita nad relaciono orjentisanom multidimenzionom strukturom podataka. Podržana su proširenja SQL-a novim tipovima funkcija kao što su ROLLUP, CUBE, WINDOW. Unapređeni su koncepti fizičke organizacije podataka, odnosno postoje novi tipovi indeksnih struktura i fizičko particionisanje. SQL optimizator je takođe unapređen.

Podaci za OLAP se uobičajeno čuvaju u OLAP bazama podataka. To su baze podataka koje su posebno dizajnirane da podrže upite za različite analize.

ROLAP (Relational OLAP Engine) je međusloj između pozadinske relacione baze podataka i OLAP klijenta. Transformiše i optimizuje SQL OLAP upite za izvršenje nad pozadinskom bazom podataka.

MOLAP (Multidimensional OLAP Engine) je direktna podrška multidimenzionalnih struktura tipa hiperkočke. Podaci u hiperkocki su keširani i nezavisni od pozadinske baze podataka. OLAP upiti se direktno izvršavaju nad hiperkockom.

HOLAP (Hybrid OLAP Engine) kombinuje dobra svojstva ROLAP i MOLAP servera. ROLAP omogućava efikasnu podršku rada sa velikim količinama podataka, MOLAP omogućava efikasne upite nad izvodima podataka manjeg obima, struktuiranim u obliku hiperkočke.

Proces analize zahteva određenu agregaciju podataka, obično na različite načine i prema različitim grupisanjima.

8. Teorija zavisnosti

8.1 Funkcionalne zavisnosti

Neka je R relacija i neka su X i Y proizvoljni podskupovi atributa iz R . Tada Y funkcionalno zavisi od X , u oznaci $X \rightarrow Y$ ako i samo ako je svakoj važećoj vrednosti torke X u R pridružena tačno jedna vrednost Y iz R .

Koristi se još i termin X funkcionalno određuje Y . Y funkcionalno zavisi od X ako i samo ako postoji funkcija f takva da važi $f(X) = Y$.

Torka $A1, A2, \dots, An$ funkcionalno određuje torku $B1, B2, \dots, Bm$, ukoliko važi ako su dve torke od R identične na svim atributima $A1, A2, \dots, An$ tada moraju da imaju iste vrednosti i u ostalim atributima $B1, B2, \dots, Bm$. Dve torke su identične ako su im identične vrednosti respektivnih komponenti.

To se označava kao $A1, A2, \dots, An \rightarrow B1, B2, \dots, Bm$.

U relaciji DOSIJE, indeks funkcionalno određuje ime, prezime, godinu rođenja i mesto rođenja. Odnosno, jednoj vrednosti indeksa je pridruženo tačno jedno ime, tačno jedno prezime, jedna godina rođenja i jedno mesto rođenja.

To se zapisuje na sledeći način:

- $\{indeks\} \rightarrow \{ime\}$
- $\{indeks\} \rightarrow \{prezime\}$
- $\{indeks\} \rightarrow \{god_rod_jen_ja\}$
- $\{indeks\} \rightarrow \{mesto_rod_jen_ja\}$

Neke funkcionalne zavisnosti koje važe u relaciji PREDMET su:

- $\{id_predmeta\} \rightarrow \{sifra\}$
- $\{id_predmeta\} \rightarrow \{naziv\}$

- $\{id_predmeta\} \rightarrow \{bodovi\}$

Neke funkcionalne zavisnosti koje važe u relaciji ISPIT su:

- $\{indeks, id_predmeta, godina_roka, oznaka_roka\} \rightarrow \{ocena\}$
- $\{indeks, id_predmeta, godina_roka, oznaka_roka\} \rightarrow \{datum_ispita\}$

Sada možemo redefinisati ključeve relacije u terminima funkcionalnih zavisnosti (FZ).

Podskup atributa X iz R relacije R je kandidat za ključ relacije R ako važi:

- za svaki podskup atributa Y iz R , X funkcionalno određuje Y , odnosno $X \rightarrow Y$
- ne postoji pravi podskup Z od X takav da Z funkcionalno određuje sve ostale attribute relacije R , odnosno da važi $Z \rightarrow R/Z$

Nadključ relacije R je skup atributa koji uključuje kao podskup bar jedan kandidat za ključ relacije R .

Svaka funkcionalna zavisnost predstavlja ograničenje integriteta. Funkcionalna zavisnost ne zavisi od trenutne vrednosti relacije i zbog toga u relaciji iz tabele 8.1 $oznaka_roka \rightarrow naziv$ i $naziv \rightarrow oznaka_roka$ nisu funkcionalne zavisnosti, iako bi moglo tako da izgleda na osnovu podataka koji se trenutno pojavljuju u tabeli.

oznaka_roka	godina_roka	naziv
jan1	2023	Januar 1 2023
feb	2023	Februar 1 2023

Tabela 8.1: Ilustracija trenutnih vrednosti u tabeli

8.1.1 Trivijalne i netrivialne funkcionalne zavisnosti

Trivijalna je funkcionalna zavisnost koja ne može da ne bude zadovoljena za bilo koji skup vrednosti u relaciji. Na primer, Ako je Y podskup od X , tada je $X \rightarrow Y$ trivijalna funkcionalna zavisnost.

Neke trivijalne funkcionalne zavisnosti su:

- $\{indeks\} \rightarrow \{indeks\}$
- $\{indeks, id_predmeta\} \rightarrow \{id_predmeta\}$
- $\{godina_roka, oznaka_roka\} \rightarrow \{godina_roka, oznaka_roka\}$

Funkcionalna zavisnost koja nije trivijalna je netrivialna.

8.1.2 Pravila izvođenja

Neka su A , B , i C proizvoljni podskupovi atributa relacije R . Tada važe Armstrongove aksiome:

- **Refleksivnost:** $B \subseteq A \implies A \rightarrow B$
- **Proširenje:** $A \rightarrow B \implies AC \rightarrow BC$

- **Tranzitivnost:** $A \rightarrow B \wedge B \rightarrow C \implies A \rightarrow C$

Iz prethodnih aksioma mogu da se izvedu i dodatna pravila:

- **Samo-određenje:** $A \rightarrow A$
- **Dekompozicija:** $A \rightarrow BC \implies A \rightarrow B \wedge A \rightarrow C$
- **Unija:** $A \rightarrow B \wedge A \rightarrow C \implies A \rightarrow BC$
- **Kompozicija:** $A \rightarrow B \wedge C \rightarrow D \implies AC \rightarrow BD$
- **Opšta teorema unifikacije:** $A \rightarrow B \wedge C \rightarrow D \implies A \cup (C - B) \rightarrow BD$

8.1.3 Skupovi funkcionalnih zavisnosti

Dva skupa funkcionalnih zavisnosti S i T nad relacijom R su **ekvivalentna** ako skup instanci relacije R koji zadovoljava S je jednak skupu instanci relacije R koji zadovoljava T [2].

U opštem slučaju, skup S funkcionalnih zavisnosti se izvodi iz skupa T funkcionalnih zavisnosti ako svaka instanca relacije koja zadovoljava sve funkcionalne zavisnosti iz T zadovoljava i sve funkcionalne zavisnosti iz S .

Posledica toga je da su dva skupa funkcionalnih zavisnosti S i T ekvivalentna ako i samo ako se svaka funkcionalna zavisnost u S izvodi iz funkcionalnih zavisnosti u T i svaka funkcionalna zavisnost u T izvodi iz funkcionalnih zavisnosti u S .

■ **Primer 8.1** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$ i skup funkcionalnih zavisnosti:

- $A \rightarrow BC$
- $BC \rightarrow DE$
- $AEF \rightarrow G$

Pokazati da važi funkcionalna zavisnost $ACF \rightarrow DG$.

Rešenje:

1. $A \rightarrow BC$ (dato)
2. $AC \rightarrow BC$ (proširenje C nad 1)
3. $BC \rightarrow DE$ (dato)
4. $AC \rightarrow DE$ (tranzitivnost 2 i 3)
5. $ACF \rightarrow DEF$ (proširenje F nad 4)
6. $ACF \rightarrow ACDEF$ (proširenje AC nad 5)
7. $ACF \rightarrow AEF$ (dekompozicija 6)
8. $AEF \rightarrow G$ (dato)
9. $ACF \rightarrow G$ (tranzitivnost 7 i 8)
10. $ACF \rightarrow D$ (dekompozicija 5)
11. $ACF \rightarrow DG$ (unija 9 i 10)

■

■ **Primer 8.2** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$ i skup funkcionalnih zavisnosti:

- $A \rightarrow BC$

- $B \rightarrow E$
- $CD \rightarrow EF$

Pokazati da važi funkcionalna zavisnost $AD \rightarrow F$.

Rešenje:

1. $A \rightarrow BC$ (dato)
2. $A \rightarrow C$ (dekompozicija 1)
3. $AD \rightarrow CD$ (proširenje 2)
4. $CD \rightarrow EF$ (dato)
5. $AD \rightarrow EF$ (tranzitivnost 3 i 4)
6. $AD \rightarrow F$ (dekompozicija 5)

■

■ **Primer 8.3** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$ i skup funkcionalnih zavisnosti:

- $A \rightarrow B$
- $A \rightarrow C$
- $CD \rightarrow E$
- $CD \rightarrow F$
- $B \rightarrow E$

Pokazati da važe funkcionalne zavisnosti:

- a) $A \rightarrow E$
- b) $AD \rightarrow F$
- c) $CD \rightarrow EF$

Rešenje:

a)

1. $A \rightarrow B$ (dato)
2. $B \rightarrow E$ (dato)
3. $A \rightarrow E$ (tranzitivnost 1 i 2)

b)

1. $A \rightarrow C$ (dato)
2. $AD \rightarrow CD$ (proširenje D nad 1)
3. $CD \rightarrow F$ (dato)
4. $AD \rightarrow F$ (tranzitivnost 2 i 3)

c)

1. $CD \rightarrow E$ (dato)
2. $CD \rightarrow F$ (dato)
3. $CD \rightarrow EF$ (unija 1 i 2)

■

■ **Primer 8.4** Neka je data relacija $R = \{A, B, C, D, E, F, G, H\}$ i skup funkcionalnih zavi-

snosti F :

- $A \rightarrow B$
- $CH \rightarrow A$
- $B \rightarrow E$
- $BD \rightarrow C$
- $EG \rightarrow H$
- $DE \rightarrow F$

Ispitati da li sledeće funkcionalne zavisnosti mogu da se izvedu iz F :

- a) $BFG \rightarrow AE$
- b) $ACG \rightarrow DH$
- c) $CEG \rightarrow AB$

Rešenje:

a)

Ispituje se za $BFG \rightarrow AE$.

Posmatrajmo FZ koje na desnoj strani imaju neke od atributa BFG i one attribute koji se mogu izvesti iz njih:

1. $B \rightarrow E$ (dato)
2. $EG \rightarrow H$ (dato)

Atributi sa desne strane su jedini atributi koji se mogu izvesti iz BFG (uključujući attribute B , F i G). Odnosno, A se ne može izvesti iz BFG , pa ni AE . FZ $BFG \rightarrow AE$ ne može da se izvede iz skupa F .

b)

Ispituje se za $ACG \rightarrow DH$.

Posmatrajmo FZ koje na desnoj strani imaju neke od atributa ACG i one attribute koji mogu da se izvedu iz njih:

1. $A \rightarrow B$ (dato)
2. $B \rightarrow E$ (dato)
3. $EG \rightarrow H$ (dato)

H se može izvesti, ali D se ne može izvesti iz ovih atributa (a zapravo ni iz jednog osim iz D). Dakle, FZ $ACG \rightarrow DH$ ne može da se izvede iz ovog skupa.

c)

Ispituje se za $CEG \rightarrow AB$.

Posmatrajmo FZ koje na desnoj strani imaju neke od atributa CEG i one attribute koji mogu da se izvedu iz njih:

1. $EG \rightarrow H$ (dato)
2. $CEG \rightarrow CH$ (proširenje C na 1)
3. $CH \rightarrow A$ (dato)

4. $CEG \rightarrow A$ (tranzitivnost 2 i 3)
5. $A \rightarrow B$ (dato)
6. $CEG \rightarrow B$ (tranzitivnost 4 i 5)
7. $CEG \rightarrow AB$ (unija 5 i 6)

Dakle, FZ $CEG \rightarrow AB$ može da se izvede iz skupa F . ■

■ **Primer 8.5** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$. Pokazati da su sledeća dva skupa funkcionalnih zavisnosti:

X :

- $B \rightarrow CD$
- $AD \rightarrow E$
- $B \rightarrow A$

i Y :

- $B \rightarrow CDE$
- $B \rightarrow ABC$
- $AD \rightarrow E$

ekvivalentna.

Rešenje:

Da bi se dokazalo da su skupovi FZ X i Y ekvivalentni, potrebno je da se dokaže da svaka FZ iz skupa X može da se izvede iz FZ iz skupa Y . I obrnuto, da svaka FZ iz skupa Y može da se izvede iz FZ iz skupa X .

Dokažimo prvo da X može da se izvede iz Y .

a)

Dokazuje se za $B \rightarrow CD$.

1. $B \rightarrow CDE$ (dato)
2. $B \rightarrow CD$ (dekompozicija 1)

b)

Dokazuje se za $AD \rightarrow E$.

1. $AD \rightarrow E$ (dato)

c)

Dokazuje se za $B \rightarrow A$.

1. $B \rightarrow ABC$ (dato)
2. $B \rightarrow A$ (dekompozicija 1)

Pokazali smo da svaka FZ iz X može da se izvede iz FZ iz Y .

Dokažimo sada da Y može da se izvede iz X .

a)

Dokazuje se za $B \rightarrow CDE$.

1. $B \rightarrow A$ (dato)
2. $BD \rightarrow AD$ (proširenje D nad 1)
3. $B \rightarrow CD$ (dato)
4. $B \rightarrow D$ (dekompozicija 3)
5. $B \rightarrow BD$ (proširenje B nad 4)
6. $B \rightarrow AD$ (tranzitivnost 5 i 2)
7. $AD \rightarrow E$ (dato)
8. $B \rightarrow E$ (tranzitivnost 6 i 7)
9. $B \rightarrow CDE$ (unija 3 i 8)

b)

Dokazuje se za $B \rightarrow ABC$.

1. $B \rightarrow A$ (dato)
2. $B \rightarrow CD$ (dato)
3. $B \rightarrow C$ (dekompozicija 2)
4. $B \rightarrow AC$ (unija 1 i 3)
5. $B \rightarrow ABC$ (proširenje B nad 4)

c)

Dokazuje se za $AD \rightarrow E$.

1. $AD \rightarrow E$ (dato)

Dokazali smo da Y može da se izvede iz X i da X može da se izvede iz Y , tako da X i Y jesu ekvivalentni. ■

8.1.4 Zatvorenje skupa funkcionalnih zavisnosti

Neka je S skup funkcionalnih zavisnosti nad relacijom R . Skup svih funkcionalnih zavisnosti koje mogu da se izvedu iz skupa S naziva se zatvorenje od S i označava sa S^+ .

Posledica je da su dva skupa funkcionalnih zavisnosti S i T nad relacijom R ekvivalentna ako važi da su im jednaka zatvorenja, odnosno da je $S^+ = T^+$.

Zatvorenje S^+ skupa FZ S može da se odredi primenom pravila - Armstrongovim aksiomama, kojima se nove FZ izvedu iz postojećih.

Algoritam za računanje zatvorenja skupa funkcionalnih zavisnosti

Algoritam za računanje zatvorenja T skupa funkcionalnih zavisnosti F sastoji se od sledećih koraka:

- Inicijalno je $T = F$.
- Za svaku funkcionalnu zavisnost t iz T primeniti refleksivnost i proširenja na t . Dodati dobijene funkcionalne zavisnosti u T .
- Za svaki par funkcionalnih zavisnosti (t_1, t_2) iz T , ako t_1 i t_2 mogu da se kombinuju

- pomoću tranzitivnosti tada dodati dobijenu funkcionalnu zavisnost u T .
- Postupak ponavljati sve dok se T menja.
- Ukoliko nema više promena na T , T je zatvorenje skupa funkcionalnih zavisnosti F , odnosno $T = F^+$.

■ **Primer 8.6** Neka su A, B, C, D, E i F atributi relacije R , i neka je F skup funkcionalnih zavisnosti dat sa:

- $A \rightarrow BC$
- $B \rightarrow E$
- $CD \rightarrow EF$

Ispitati da li je funkcionalna zavisnost $AD \rightarrow F$ u F^+ .

Rešenje:

- $A \rightarrow BC$ (dato)
- $A \rightarrow C$ (dekompozicija 1)
- $AD \rightarrow CD$ (proširenje D na 2)
- $CD \rightarrow EF$ (dato)
- $AD \rightarrow EF$ (tranzitivnost 3 i 4)
- $AD \rightarrow F$ (dekompozicija 5)

Dakle, FZ $AD \rightarrow F$ jeste u F^+ . ■

■ **Primer 8.7** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti F :

- $AB \rightarrow C$
- $CD \rightarrow D$
- $A \rightarrow E$

Ispitati da li sledeće funkcionalne zavisnosti pripadaju F^+ :

- $AB \rightarrow D$
- $AC \rightarrow D$
- $A \rightarrow C$
- $A \rightarrow B$
- $BE \rightarrow D$

Rešenje:

a)

Ispituje se $AB \rightarrow D$.

D se može izvesti samo D , a pošto sa leve strane ove FZ nema D , onda ova FZ ne pripada F^+ .

b)

Ispituje se $AC \rightarrow D$.

Isti je odgovor kao i pod a).

c)

Ispituje se $A \rightarrow C$.

Iz A može da se izvede samo E , tako da iz A ne može da se izvede C . Dakle, FZ $A \rightarrow C$ nije u F^+ .

d)

Ispituje se $A \rightarrow B$.

B se ne nalazi sa desne strane nijedne FZ, tako da se ne može izvesti ni iz čega drugog osim iz B , pa ne važi ni ova FZ.

e)

Ispituje se $BE \rightarrow D$.

Isti odgovor kao i pod a) i b).

Ni jedna od ovih FZ se ne može izvesti iz F , pa ni jedna ne pripada zatvorenju skupa F , odnosno F^+ . ■

■ **Primer 8.8** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti F :

- $A \rightarrow BC$
- $CD \rightarrow E$
- $B \rightarrow D$
- $E \rightarrow A$

Odrediti zatvorenje F^+ skupa F .

Rešenje:

1. $A \rightarrow BC$ (dato)
2. $A \rightarrow B$ (dekompozicija 1)
3. $B \rightarrow D$ (dato)
4. $A \rightarrow D$ (tranzitivnost 1 i 3)
5. $A \rightarrow BCD$ (unija 1 i 4)
6. $A \rightarrow CD$ (dekompozicija 5)
7. $CD \rightarrow E$ (dato)
8. $A \rightarrow E$ (tranzitivnost 6 i 7)
9. $A \rightarrow A$ (samo-određenje)
10. $A \rightarrow ABCDE$ (unija 1, 4, 8 i 9)
11. $E \rightarrow A$ (dato)
12. $E \rightarrow ABCDE$ (tranzitivnost 11 i 10)
13. $CD \rightarrow ABCDE$ (tranzitivnost 7 i 12)
14. $BC \rightarrow CD$ (proširenje C iz 3)
15. $BC \rightarrow ABCDE$ (tranzitivnost 14 i 13)

Iz A , E , CD i BC se mogu izvesti svi atributi relacije R , što je pokazano u 10, 12, 13, 15, pa se mogu izvesti i svi podskupovi relacije R primenjujući dekompoziciju. Odnosno, mogu se napraviti projekcije izvedenih relacija.

Takođe, FZ 10, 12, 13, 15 se ne mogu redukovati sa leve strane, ali se sa leve strane

mogu dodavati sve kombinacije preostalih atributa primenjujući pravilo refleksivnosti.

■ **Primer 8.9** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti F :

- $A \rightarrow B$
- $CD \rightarrow E$
- $B \rightarrow C$

Odrediti zatvorenje F^+ skupa F .

Rešenje:

1. $A \rightarrow B$ (dato)
2. $B \rightarrow C$ (dato)
3. $A \rightarrow C$ (tranzitivnost 1 i 2)
4. $A \rightarrow BC$ (unija 1 i 3)
5. $AD \rightarrow ABCD$ (proširenje AD nad 4)
6. $AD \rightarrow CD$ (dekompozicija 5)
7. $CD \rightarrow E$ (dato)
8. $AD \rightarrow E$ (tranzitivnost 6 i 7)
9. $AD \rightarrow ABCDE$ (unija 5 i 8)

Iz AD se mogu izvesti svi atributi relacije R , što je pokazano u 9, pa se mogu izvesti i svi podskupovi relacije R primenjujući dekompoziciju. Odnosno, mogu se napraviti projekcije izvedenih relacija.

Takođe, FZ se ne mogu redukovati sa leve strane, ali se sa leve strane mogu dodavati sve kombinacije preostalih atributa primenjujući pravilo refleksivnosti.

■ **Napomena.** U praksi se određivanje zatvorenja skupa funkcionalnih zavisnosti relativno retko radi pošto je vrlo neefikasno. Umesto toga, često treba izračunati da li data funkcionalna zavisnost pripada zatvorenju skupa funkcionalnih zavisnosti, odnosno da li se može izvesti iz datog skupa funkcionalnih zavisnosti.

8.1.5 Zatvorenje skupa atributa

Da bi odredili da li je K nadključ treba odrediti skup atributa relacije R koji funkcionalno zavisi od K . Ako je skup atributa K nadključ tada K funkcionalno određuje sve attribute u R . K je nadključ ako i samo ako je K^+ od K u odnosu da dati skup funkcionalnih zavisnosti jednako skupu svih atributa relacije R .

Neka je $A = \{A_1, A_2, \dots, A_n\}$ skup atributa relacije R i S skup funkcionalnih zavisnosti nad R . Zatvorenje skupa atributa A u odnosu na skup funkcionalnih zavisnosti S je skup atributa B takvih da svaka relacija koja zadovoljava sve funkcionalne zavisnosti u skupu S zadovoljava i funkcionalnu zavisnost $A_1, A_2, \dots, A_n \rightarrow B$.

Zatvorenje skupa atributa A se označava sa A^+ . Uvek važi da su svi pojedinačni atributi iz A u A^+ .

Algoritam za određivanje zatvorenja skupa atributa

Neka je $A = \{A_1, A_2, \dots, A_n\}$ skup atributa i S skup funkcionalnih zavisnosti. Algoritam za određivanje zatvorenja A^+ skupa atributa A je:

- Izvršiti dekompoziciju svih funkcionalnih zavisnosti tako da imaju samo jedan atribut na desnoj strani.
- Neka je X skup atributa koji predstavlja zatvorenje. Inicijalno $X = \{A_1, A_2, \dots, A_n\}$.
- Tražiti funkcionalne zavisnosti oblika $B_1, B_2, \dots, B_m \rightarrow C$ takve da svako B_i pripada X i da C ne pripada X . Dodati C u X . Ponavljati pretragu sve dok ima promena skupa X .
- Ukoliko ne postoji atribut koji bi mogao da se doda skupu X tada je X zatvorenje skupa atributa A , odnosno $X = \{A_1, A_2, \dots, A_n\}^+$.

■ **Primer 8.10** Neka relacija sadrži attribute A, B, C, D, E i F . Neka važe sledeće funkcionalne zavisnosti:

- $AB \rightarrow C$
- $BC \rightarrow AD$
- $D \rightarrow E$
- $CF \rightarrow B$

Šta je zatvorenje skupa atributa $\{A, B\}$?

Rešenje:

Prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut:

1. $AB \rightarrow C$
2. $BC \rightarrow A$
3. $BC \rightarrow D$
4. $D \rightarrow E$
5. $CF \rightarrow B$

Inicijalno $X = \{A, B\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X :

- Na osnovu 1 se dodaje C , pa je $X = \{A, B, C\}$.
- Na osnovu 3 se dodaje D , pa je $X = \{A, B, C, D\}$.
- Na osnovu 4 se dodaje E , pa je $X = \{A, B, C, D, E\}$.

Ovde se staje, pošto ne postoji više ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{A, B, C, D, E\} = \{A, B\}^+$. ■

■ **Primer 8.11** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti:

- $A \rightarrow B$
- $B \rightarrow C$
- $CD \rightarrow E$

Odrediti zatvorenje (skupa) atributa $\{A\}$?

Rešenje:

Prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut:

1. $A \rightarrow B$
2. $B \rightarrow C$
3. $CD \rightarrow E$

Inicijalno $X = \{A\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X :

- Na osnovu 1 se dodaje B , pa je $X = \{A, B\}$.
- Na osnovu 2 se dodaje C , pa je $X = \{A, B, C\}$.

Ovde se staje, pošto ne postoji više ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{A, B, C\} = \{A\}^+$. ■

■ **Primer 8.12** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti F :

- $A \rightarrow B$
- $B \rightarrow C$
- $CD \rightarrow E$

Da li funkcionalna zavisnost $A \rightarrow E$ pripada zatvorenju F^+ ?

Rešenje:

Može se izračunati zatvorenje atributa A . Ukoliko to zatvorenje sadrži atribut E , tada i samo tada će važiti i ova FZ.

Skup FZ isti kao i u prethodnom zadatku, a u njemu smo već izračunali da je $\{A\}^+ = \{A, B, C\}$.

Pošto E ne pripada zatvorenju atributa A onda neće važiti ni FZ $A \rightarrow E$. ■

■ **Primer 8.13** Neka je data relacija $R = \{A, B, C, D, E, F\}$ i skup funkcionalnih zavisnosti F :

- $AB \rightarrow C$
- $C \rightarrow A$
- $BC \rightarrow D$
- $ACD \rightarrow B$
- $BE \rightarrow C$
- $EG \rightarrow H$

- $DE \rightarrow F$

Ispitati da li sledeće funkcionalne zavisnosti mogu da se izvedu iz F :

- a) $BFG \rightarrow AE$
- b) $ACG \rightarrow DH$
- c) $CEG \rightarrow AB$

Rešenje:

Potrebno je izračunati zatvorenja atributa sa leve strane i proveriti da li se u njima nalaze atributi sa desne strane. Ukoliko se nalaze, onda tražene FZ mogu da se izvedu, u suprotnom ne mogu da se izvedu.

Prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut:

1. $AB \rightarrow C$
2. $C \rightarrow A$
3. $BC \rightarrow D$
4. $ACD \rightarrow B$
5. $BE \rightarrow C$
6. $EG \rightarrow H$
7. $DE \rightarrow F$

a)

Ispituje se $BFG \rightarrow AE$.

Računa se zatvorenje atributa $\{B, F, G\}$.

Inicijalno $X = \{B, F, G\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X .

Ovde se staje, pošto ne postoji ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{B, F, G\} = \{B, F, G\}^+$. Kako se ni A ni E ne nalaze u ovom skupu, to ova FZ ne može da se izvede iz FZ iz F .

b)

Ispituje se $ACG \rightarrow DH$.

Računa se zatvorenje atributa $\{A, C, G\}$.

Dekompozicije smo izračunali u prethodnom delu, tako da možemo da krenemo sa sledećim korakom.

Inicijalno $X = \{A, C, G\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X .

Ovde se staje, pošto ne postoji ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{A, C, G\} = \{A, C, G\}^+$. Kako se ni D ni H ne nalaze u ovom skupu, to ni ova FZ ne može da se izvede iz FZ iz F .

c)

Ispituje se $CEG \rightarrow AB$.

Računa se zatvorenje atributa $\{C, E, G\}$.

Dekompozicije smo izračunali u prethodnom delu, tako da možemo da krenemo sa sledećim korakom.

Inicijalno $X = \{C, E, G\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X :

- Na osnovu 2 se dodaje A , pa je $X = \{A, C, E, G\}$.
- Na osnovu 6 se dodaje H , pa je $X = \{A, C, E, G, H\}$.

Ovde se staje, pošto ne postoji više ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{A, C, E, G, H\} = \{C, E, G\}^+$. Kako se B ne nalazi u ovom skupu, to ni ova FZ ne može da se izvede iz FZ iz F .

■

■ **Primer 8.14** Neka je data relacija $R = \{A, B, C, D, E, F\}$ i skup funkcionalnih zavisnosti F :

- $AB \rightarrow C$
- $AD \rightarrow E$
- $B \rightarrow D$
- $AF \rightarrow B$
- $B \rightarrow E$

Odrediti zatvorenje (skupa) atributa:

- a) $\{A\}$
- b) $\{A, B\}$
- c) $\{A, F\}$

relacije R .

Rešenje:

Prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut (u ovom slučaju to su iste FZ pošto svaka sa desne strane ima samo po jedan atribut):

1. $AB \rightarrow C$
2. $AD \rightarrow E$
3. $B \rightarrow D$
4. $AF \rightarrow B$
5. $B \rightarrow E$

a)

Izračunava se $\{A\}^+$.Samo iz A se ništa više ne može izvesti, tako da je $\{A\}^+ = \{A\}$.

b)

Izračunava se $\{A, B\}^+$.

Inicijalno $X = \{A, B\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X :

- Na osnovu 1 se dodaje C , pa je $X = \{A, B, C\}$.
- Na osnovu 3 se dodaje D , pa je $X = \{A, B, C, D\}$.
- Na osnovu 5 se dodaje E , pa je $X = \{A, B, C, D, E\}$.

Ovde se staje, pošto ne postoji više ni jedna FZ tako da su atributi sa leve strane kompletno u skupu X , a atribut sa desne strane da nije u skupu X , tj ništa se više ne može dodati.

Stoga, $X = \{A, B, C, D, E\} = \{A, B\}^+$.

c)

Izračunava se $\{A, F\}^+$.

Inicijalno $X = \{A, F\}$. Traže se FZ takve da su svi atributi sa leve strane u skupu X , a atribut sa desne strane da nije u skupu X :

- Na osnovu 4 se dodaje B , pa je $X = \{A, B, F\}$.
- Na osnovu 3 se dodaje D , pa je $X = \{A, B, D, F\}$.
- Na osnovu 5 se dodaje E , pa je $X = \{A, B, D, E, F\}$.
- Na osnovu 1 se dodaje C , pa je $X = \{A, B, C, D, E, F\}$.

Ovde se staje, pošto su svi atributi dodati, odnosno $X = R = \{AF\}^+$.

■

8.1.6 Izračunavanje kandidata za ključ preko zatvorenja skupa atributa

Skup atributa K relacije R je kandidat za ključ ukoliko važe oba sledeća uslova:

- **Jedinstvenost** - skup K jedinstveno određuje sve attribute relacije R
- **Minimalnost** - iz skupa K ne može da se ukloni ni jedan atribut tako da za tako dobijen skup važi uslov jedinstvenosti

Kandidati za ključ se mogu izračunati preko zatvorenja atributa relacije R . Neka je A podskup atributa relacije R . Ukoliko je zatvorenje od A jednako celoj relaciji R , tada važi uslov jedinstvenosti. Ukoliko ni za jedan pravi podskup atributa A ne važi da je zatvorenje tog podskupa jednako relaciji R , onda za skup A važi i uslov minimalnosti, odnosno skup A je kandidat za ključ.

■ **Primer 8.15** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$ i skup funkcionalnih zavisnosti:

- $A \rightarrow BC$
- $B \rightarrow D$
- $CD \rightarrow E$
- $E \rightarrow A$

Odrediti zatvorenja svih podskupova atributa relacije R .

Navesti sve kandidate za ključ relacije R .

Rešenje:

Potrebno je prvo za svaki podskup od R izračunati zatvorenje tog podskupa. Ukoliko je zatvorenje nekog podskupa jednako celom skupu R , tada je zadovoljeno pravilo jedinstvenosti. Ukoliko za takav podskup važi i pravilo minimalnosti, onda je to kandidat za ključ. Odnosno, kandidati za ključ su svi podskupovi skupa R čija su zatvorenja jednaka celom skupu R i iz kojih se ne može ukloniti ni jedan atribut tako da je zatvorenje takvog skupa jednako celom skupu R .

Zatvorenja jednočlanih podskupova:

$$\begin{aligned}\{A\}^+ &= \{A, B, C, D, E\} \\ \{B\}^+ &= \{B, D\} \\ \{C\}^+ &= \{C\} \\ \{D\}^+ &= \{D\} \\ \{E\}^+ &= \{A, B, C, D, E\}\end{aligned}$$

Zatvorenja dvočlanih podskupova:

$$\begin{aligned}\{A, B\}^+ &= \{A, B, C, D, E\} \\ \{A, C\}^+ &= \{A, B, C, D, E\} \\ \{A, D\}^+ &= \{A, B, C, D, E\} \\ \{A, E\}^+ &= \{A, B, C, D, E\} \\ \{B, C\}^+ &= \{A, B, C, D, E\} \\ \{B, D\}^+ &= \{B, D\} \\ \{B, E\}^+ &= \{A, B, C, D, E\} \\ \{C, D\}^+ &= \{A, B, C, D, E\} \\ \{C, E\}^+ &= \{A, B, C, D, E\} \\ \{D, E\}^+ &= \{A, B, C, D, E\}\end{aligned}$$

Zatvorenja tročlanih podskupova:

$$\begin{aligned}\{A, B, C\}^+ &= \{A, B, C, D, E\} \\ \{A, B, D\}^+ &= \{A, B, C, D, E\} \\ \{A, B, E\}^+ &= \{A, B, C, D, E\} \\ \{A, C, D\}^+ &= \{A, B, C, D, E\} \\ \{A, C, E\}^+ &= \{A, B, C, D, E\} \\ \{A, D, E\}^+ &= \{A, B, C, D, E\} \\ \{B, C, D\}^+ &= \{A, B, C, D, E\}\end{aligned}$$

$$\begin{aligned}\{B, C, E\}^+ &= \{A, B, C, D, E\} \\ \{B, D, E\}^+ &= \{A, B, C, D, E\} \\ \{C, D, E\}^+ &= \{A, B, C, D, E\}\end{aligned}$$

Zatvorenja četvoročlanih podskupova:

$$\begin{aligned}\{A, B, C, D\}^+ &= \{A, B, C, D, E\} \\ \{A, B, C, E\}^+ &= \{A, B, C, D, E\} \\ \{A, B, D, E\}^+ &= \{A, B, C, D, E\} \\ \{A, C, D, E\}^+ &= \{A, B, C, D, E\} \\ \{B, C, C, E\}^+ &= \{A, B, C, D, E\}\end{aligned}$$

Zatvorenje petočlanog skupa:

$$\{A, B, C, D, E\}^+ = \{A, B, C, D, E\}$$

Podskupovi $\{A\}$, $\{E\}$, $\{B, C\}$ i $\{C, D\}$ su kandidati za ključ. Nadskupovi ovih skupova nisu kandidati za ključ, jer za njih neće važiti pravilo minimalnosti. ■

■ **Primer 8.16** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup funkcionalnih zavisnosti:

- $AB \rightarrow C$
- $CD \rightarrow E$
- $DE \rightarrow B$

Da li su AB i ABD kandidati za ključ relacije R ? Odgovor obrazložiti.

Rešenje:

Izračunajmo zatvorenja ovih skupova atributa. Ukoliko je zatvorenje jednako celom R i ukoliko je minimalno onda to jeste kandidat za ključ.

Za računanje zatvorenja skupa atributa prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut (u ovom slučaju to su iste FZ pošto svaka sa desne strane ima samo po jedan atribut):

1. $AB \rightarrow C$
2. $CD \rightarrow E$
3. $DE \rightarrow B$

Tražimo zatvorenje skupa atributa $\{A, B\}$.

Inicijalno, zatvorenje je $X = \{A, B\}$. Na dalje dodajemo desnu stranu FZ ukoliko se leva strana nalazi u skupu X :

- Na osnovu 1 dodaje se C , pa je $X = \{A, B, C\}$.

Nema se više šta dodati, pa pošto $X \neq R$, ne važi pravilo jedinstvenosti, odnosno AB nije kandidat za ključ.

Tražimo zatvorenje skupa atributa $\{A, B, D\}$.

Inicijalno, zatvorenje je $X = \{A, B, D\}$. Na dalje dodajemo desnu stranu FZ ukoliko se leva strana nalazi u skupu X :

- Na osnovu 1 dodaje se C , pa je $X = \{A, B, C, D\}$.
- Na osnovu 2 dodaje se E , pa je $X = \{A, B, C, D, E\}$.

Nema se više šta dodati pošto je $X = R$, pa važi pravilo jedinstvenosti. Ne može se ni jedan atribut izbaciti tako da i dalje važi pravilo jedinstvenosti:

$$\{A, B\}^+ = \{A, B, C\}$$

$$\{A, D\}^+ = \{A, D\}$$

$$\{B, D\}^+ = \{B, D\}$$

Dakle, važi i pravilo minimalnosti, pa ABD jeste kandidat za ključ. ■

■ **Primer 8.17** Neka je data relacija $R = \{A, B, C, D, E, F, G\}$ i skup funkcionalnih zavisnosti:

- $AB \rightarrow C$
- $CD \rightarrow E$
- $EF \rightarrow G$
- $FG \rightarrow E$
- $DE \rightarrow C$
- $BC \rightarrow A$

Da li su:

- BDF
- $ACDF$
- $ABDFG$
- $BDFG$

kandidati za ključ relacije R ? Odgovor obrazložiti.

Rešenje:

Prvo se izračunavaju dekompozicije tako da svaka FZ ima sa desne strane samo po jedan atribut (u ovom slučaju to su iste FZ pošto svaka sa desne strane ima samo po jedan atribut):

1. $AB \rightarrow C$
2. $CD \rightarrow E$
3. $EF \rightarrow G$
4. $FG \rightarrow E$
5. $DE \rightarrow C$
6. $BC \rightarrow A$

a)

Ispituje se BDF .

Iz $X = \{B, F, D\}$ ne može da se izvede više ni jedan atributa, odnosno $\{B, F, D\}^+ = \{B, F, D\}$, tako da ovo nije kandidat za ključ.

b)

Ispituje se $ACDF$.

Inicijalno je $X = \{A, C, D, F\}$. Iz $\{A, C, D, F\}$ se može izvesti E na osnovu 1, pa je $X = \{A, C, D, E, F\}$. B se nikako ne može izvesti osim iz B , pošto nije sa desne strane ni jedne FZ, tako da $ACDF$ nije kandidat za ključ.

c)

Ispituje se $ABDFG$.

Iz $X = \{A, B, D, F, G\}$ se može izvesti C na osnovu 1. Može se izvesti i E iz 2. Tako da zatvorenje od $\{A, B, D, F, G\}$ jeste ceo skup R . Uslov jedinstvenosti je ispunjen. Međutim, da bi bio kandidat za ključ, mora i uslov minimalnosti da bude ispunjen.

Ako posmatramo skup $Y = \{B, D, F, G\}$ kao pod d), dobićemo da se iz njega može izvesti:

- E na osnovu 4, $Y = \{B, D, E, F, G\}$.
- C na osnovu 5, $Y = \{B, C, D, E, F, G\}$.
- A na osnovu 6, $Y = \{A, B, C, D, E, F, G\}$.

Time smo dobili da je zatvorenje skupa $\{B, D, F, G\}$ ceo skup R . Dodatno, ni jedan atribut se ne može ukloniti tako da i dalje zatvorenje bude ceo skup R :

$$\begin{aligned}\{B, D, F\}^+ &= \{B, D, F\} \\ \{B, D, G\}^+ &= \{B, D, G\} \\ \{B, F, G\}^+ &= \{B, F, E, G\} \\ \{D, F, G\}^+ &= \{C, D, E, F, G\}\end{aligned}$$

Dakle, $BDFG$ jeste kandidat za ključ relacije R .

Samim tim, $ABDFG$ nije kandidat za ključ relacije R jer je nadskup jednog kandidata za ključ relacije R . ■

Napomena. Mogu se zaključiti neka pravila koja važe:

- Ako se atribut ne javlja na *desnoj* strani niti jedne funkcionalne zavisnosti tada on *mora* da bude deo ključa
- Od preostalih atributa, ako se atribut ne javlja na *levoj* strani niti jedne funkcionalne zavisnosti tada on *nije* deo ključa

■ **Primer 8.18** Neka je data relacija $R = \{A, B, C, D, E, F, G, H\}$ i skup funkcionalnih zavisnosti:

- $CD \rightarrow A$
- $EC \rightarrow H$
- $GHB \rightarrow AB$
- $C \rightarrow D$
- $EG \rightarrow A$
- $H \rightarrow B$
- $BE \rightarrow CD$

- $EC \rightarrow B$

Naći sve kandidate za ključ relacije R .

Rešenje:

Računa se dekompozicija svih FZ tako da sa desne strane postoji samo po jedan atribut:

1. $CD \rightarrow A$
2. $EC \rightarrow H$
3. $GHB \rightarrow A$
4. $GHB \rightarrow B$
5. $C \rightarrow D$
6. $EG \rightarrow A$
7. $H \rightarrow B$
8. $BE \rightarrow C$
9. $BE \rightarrow D$
10. $EC \rightarrow B$

Primetimo da sa desne strane nisu atributi E , F i G , što znači da oni sigurno moraju da budu u sastavu primarnog ključa, jer se nikako drugačije ne mogu izvesti.

Primetimo da od preostalih atributa, sa leve strane nije atribut A , tako da on ne ulazi u sastav kandidata za ključ. Atribut F se takođe ne nalazi sa leve strane nijedne FZ, ali je on već pokriven prvim uslovom tako da će biti u sastavu kandidata za ključ.

Izračunava se zatvorenje skupa $\{E, F, G\}$. Iz ovog skupa se može izvesti A na osnovu 6. Ovaj skup se više ne može proširiti. Odnosno, $\{E, F, G\}^+ = \{E, F, G, A\}$.

Ovaj skup nije kandidat za ključ, tako da ga proširujemo sa po jednim atributom. Izračunavaju se zatvorenja dopuštenih četvoročlanih podskupova (A nije dopušteni atribut):

$$\{B, E, F, G\}^+ =$$

- $\{A, B, E, F, G\}$ (na osnovu 6)
- $\{A, B, C, E, F, G\}$ (na osnovu 8)
- $\{A, B, C, D, E, F, G\}$ (na osnovu 5)
- $\{A, B, C, D, E, F, G, H\}$ (na osnovu 2)

$$\{C, E, F, G\}^+ =$$

- $\{A, C, E, F, G\}$ (na osnovu 6)
- $\{A, C, D, E, F, G\}$ (na osnovu 5)
- $\{A, C, D, E, F, G, H\}$ (na osnovu 2)
- $\{A, B, C, D, E, F, G, H\}$ (na osnovu 7)

$$\{D, E, F, G\}^+ =$$

- $\{A, D, E, F, G\}$ (na osnovu 6)

$$\{E, F, G, H\}^+ =$$

- $\{A, E, F, G, H\}$ (na osnovu 6)

- $\{A, B, E, F, G, H\}$ (na osnovu 7)
- $\{A, B, C, E, F, G, H\}$ (na osnovu 8)
- $\{A, B, C, D, E, F, G, H\}$ (na osnovu 5)

Četvoročlani kandidati za ključ su skupovi $\{B, E, F, G\}$, $\{C, E, F, G\}$ i $\{E, F, G, H\}$.

Svi nadskupovi ovih skupova nisu kandidati za ključ, tako da njih ne treba proširivati atributima u cilju potrage za kandidatima za ključ.

Kako od skupa $\{D, E, F, G\}$ nismo došli do cele relacije R , to nastavljamo potragu i u petočlanim skupovima koji sadrže skup $\{D, E, F, G\}$. Međutim, nemamo čime proširiti ovaj skup, a da ne obuhvatimo neki od kandidata za ključ koji smo već našli - svaki od skupova $\{B, D, E, F, G\}$, $\{C, D, E, F, G\}$ i $\{D, E, F, G, H\}$ sadrži po neki kandidat za ključ. Stoga, potraga za ključevima ovde staje.

Konačno, svi kandidati za ključ su $BEFG$, $CEFG$ i $EFGH$. ■

8.1.7 Pokrivač skupa funkcionalnih zavisnosti

Ako su S_1 i S_2 dva skupa funkcionalnih zavisnosti i ako je svaka funkcionalna zavisnost iz S_1 uključena u S_2 , odnosno ako je S_1^+ podskup od S_2^+ tada je S_2 pokrivač za S_1 .

Ako je S_1 pokrivač od S_2 i S_2 pokrivač od S_1 tada su S_1 i S_2 ekvivalentni.

8.1.8 Nereducibilni skup funkcionalnih zavisnosti

Funkcionalna zavisnost $X \rightarrow Y$ u skupu S funkcionalnih zavisnosti je levo nereducibilna ako iz X ne može da se ukloni ni jedan atribut bez promene zatvorenja S^+ .

Skup funkcionalnih zavisnosti S je nereducibilan ako:

- Desna strana svake funkcionalne zavisnosti u S sadrži tačno jedan atribut
- Leva strana svake funkcionalne zavisnosti je levo nereducibilna
- Ni jedna funkcionalna zavisnost ne može da se ukloni iz S bez promene S^+

Skup funkcionalnih zavisnosti koji zadovoljava prethodna pravila naziva se **minimalan** ili **kanonički**.

Algoritam za određivanje nereducibilnog skupa funkcionalnih zavisnosti

Neka je S skup funkcionalnih zavisnosti. Tada se nereducibilan skup funkcionalnih zavisnosti može dobiti na sledeći način:

- Koristeći pravilo dekompozicije prepisati sve funkcionalne zavisnosti iz S tako da desna strana sadrži tačno jedan atribut.
- Eliminirati sve redundantne funkcionalne zavisnosti iz skupa funkcionalnih zavisnosti dobijenih prethodnim postupkom, odnosno one funkcionalne zavisnosti koje se mogu izvesti iz preostalih funkcionalnih zavisnosti.
- Redukovati leve strane funkcionalnih zavisnosti gde je to moguće.

■ **Primer 8.19** Neka relacija R ima attribute A, B, C, D i skup funkcionalnih zavisnosti:

- $A \rightarrow BC$
- $B \rightarrow C$
- $A \rightarrow B$
- $AB \rightarrow C$
- $AC \rightarrow D$

Naći nereducibilni skup funkcionalnih zavisnosti ekvivalentan datom skupu.

Rešenje:

Prvo se dekompozicijom dobijaju FZ takve da sa desne strane bude samo po jedan atribut:

1. $A \rightarrow B$
2. $A \rightarrow C$
3. $B \rightarrow C$
4. $A \rightarrow B$
5. $AB \rightarrow C$
6. $AC \rightarrow D$

Zatim se uklanjaju redundantne FZ.

1)

Prvo to je FZ4 $A \rightarrow B$ zato što je duplirana.

Dobija se skup FZ:

1. $A \rightarrow B$
2. $A \rightarrow C$
3. $B \rightarrow C$
4. $AB \rightarrow C$
5. $AC \rightarrow D$

Dakle, FZ4 se može ukloniti.

2)

Kako se iz FZ1 $A \rightarrow B$ i FZ3 $B \rightarrow C$ može izvesti FZ2 $A \rightarrow C$ pravilom tranzitivnosti, to se i FZ2 može ukloniti.

Dobija se skup FZ:

1. $A \rightarrow B$
2. $B \rightarrow C$
3. $AB \rightarrow C$
4. $AC \rightarrow D$

Stoga, FZ1 se takođe može ukloniti.

3)

FZ3 $AB \rightarrow C$ se može dobiti na sledeći način:

1. $A \rightarrow B$ (dato)
2. $B \rightarrow C$ (dato)
3. $AB \rightarrow B$ (refleksivnost)
4. $AB \rightarrow C$ (tranzitivnost 3 i 2)

tako da se i ona može ukloniti.

Dobija se skup FZ:

1. $A \rightarrow B$
2. $B \rightarrow C$
3. $AC \rightarrow D$

Proverava se da li se neke FZ mogu redukovati sa leve strane.

4)

FZ3 $AC \rightarrow D$ se može redukovati na sledeći način:

1. $A \rightarrow B$ (dato)
2. $B \rightarrow C$ (dato)
3. $A \rightarrow C$ (tranzitivnost 1 i 2)
4. $A \rightarrow AC$ (proširenje A nad 3)
5. $AC \rightarrow D$ (dato)
6. $A \rightarrow D$ (tranzitivnost 4 i 5)

Tako da se ona zamenjuje sa FZ $A \rightarrow D$.

Dobija se skup FZ:

1. $A \rightarrow B$
2. $B \rightarrow C$
3. $A \rightarrow D$

Iz ovog skupa se ne može ukloniti nijedna FZ, niti se leve strane mogu redukovati.

Stoga, dobijen je nereducibilni skup FZ:

- $A \rightarrow B$
- $B \rightarrow C$
- $A \rightarrow D$

koji je ekvivalentan početnom skupu FZ.

■

■ **Primer 8.20** Neka je data relacija $R = \{A, B, C, D, E, F\}$ i skup F funkcionalnih zavisnosti:

- $AB \rightarrow C$
- $C \rightarrow A$
- $BC \rightarrow D$
- $ACD \rightarrow B$
- $BE \rightarrow C$
- $CE \rightarrow FA$
- $CF \rightarrow BD$
- $D \rightarrow EF$

Odrediti nereducibilni pokrivač ovog skupa F .

Rešenje:

Dekompozicijom je prvo potrebno napraviti FZ takve da sa desne strane postoji samo jedan atribut:

1. $AB \rightarrow C$
2. $C \rightarrow A$
3. $BC \rightarrow D$
4. $ACD \rightarrow B$
5. $BE \rightarrow C$
6. $CE \rightarrow A$
7. $CE \rightarrow F$
8. $CF \rightarrow B$
9. $CF \rightarrow D$
10. $D \rightarrow E$
11. $D \rightarrow F$

Zatim je potrebno ukloniti FZ koje mogu da se dobiju iz drugih.

1)

FZ6 $CE \rightarrow A$ se može dobiti iz FZ2 $C \rightarrow A$:

1. $C \rightarrow A$ (dato)
2. $CE \rightarrow AE$ (proširenje E nad 1)
3. $CE \rightarrow A$ (dekompozicija 2)

Tako da se ona može ukloniti.

2)

FZ9 $CF \rightarrow D$ se može dobiti iz FZ8 $CF \rightarrow B$ i FZ3 $BC \rightarrow D$:

1. $CF \rightarrow B$ (dato)
2. $CF \rightarrow BC$ (proširenje C nad 1)
3. $BC \rightarrow D$ (dato)

4. $CF \rightarrow D$ (tranzitivnost 2 i 3)

Pa se i ona može ukloniti.

3)

FZ4 $ACD \rightarrow B$ se može dobiti preko FZ11 $D \rightarrow F$, FZ8 $CF \rightarrow B$:

1. $D \rightarrow F$ (dato)
2. $ACD \rightarrow ACF$ (proširenje AC nad 1)
3. $CF \rightarrow B$ (dato)
4. $ACF \rightarrow AB$ (proširenje A nad 3)
5. $ACD \rightarrow AB$ (tranzitivnost 2 i 4)
6. $ACD \rightarrow B$ (dekompozicija 5)

Stoga, i ona se može ukloniti.

Kako dalje redukcije nisu moguće dobija se sledeći nereducibilan skup FZ:

- $AB \rightarrow C$
- $C \rightarrow A$
- $BC \rightarrow D$
- $BE \rightarrow C$
- $CE \rightarrow F$
- $CF \rightarrow B$
- $D \rightarrow E$
- $D \rightarrow F$

Ovaj zadatak se mogao uraditi i na drugi način.

1)

FZ6 $CE \rightarrow A$ može da se izvede iz FZ2 $C \rightarrow A$:

1. $C \rightarrow A$ (dato)
2. $CE \rightarrow AE$ (proširenje E nad 1)
3. $CE \rightarrow A$ (dekompozicija 2)

Pa se ona može ukloniti.

2)

FZ8 $CF \rightarrow B$ može da se dobije preko FZ2 $C \rightarrow A$, FZ4 $ACD \rightarrow B$ i FZ9 $CF \rightarrow D$:

1. $C \rightarrow A$ (dato)
2. $CF \rightarrow D$ (dato)
3. $CF \rightarrow AD$ (kompozicija 1 i 2)
4. $CF \rightarrow ACD$ (proširenje C nad 3)
5. $ACD \rightarrow B$ (dato)

6. $CF \rightarrow B$ (tranzitivnost 4 i 5)

Pa se i ona može ukloniti.

3)

FZ4 $ACD \rightarrow B$ može da se zameni sa $CD \rightarrow B$ pomoću FZ2 $C \rightarrow A$:

1. $C \rightarrow A$ (dato)
2. $CD \rightarrow ACD$ (proširenje CD nad 1)
3. $ACD \rightarrow B$ (dato)
4. $CD \rightarrow B$ (tranzitivnost 2 i 3)

Pa se A uklanja iz ove FZ.

Kako dalje redukcije nisu moguće dobija se sledeći nereducibilan skup FZ:

- $AB \rightarrow C$
- $C \rightarrow A$
- $BC \rightarrow D$
- $CD \rightarrow B$
- $BE \rightarrow C$
- $CE \rightarrow F$
- $CF \rightarrow D$
- $D \rightarrow E$
- $D \rightarrow F$

Svaki od ovih (različitih) skupova je nereducibilan pokrivač originalnog skupa funkcionalnih zavisnosti.

■

■ **Primer 8.21** Neka je data relacija $R = \{A, B, C, D, E, F\}$ i skup F funkcionalnih zavisnosti:

- $AB \rightarrow D$
- $B \rightarrow C$
- $AE \rightarrow B$
- $A \rightarrow D$
- $D \rightarrow EF$

Odrediti nereducibilni skup funkcionalnih zavisnosti relacije R .

Rešenje:

Dekompozicijom se dobijaju FZ koje sa desne strane imaju samo po jedan atribut:

1. $AB \rightarrow D$
2. $B \rightarrow C$
3. $AE \rightarrow B$
4. $A \rightarrow D$
5. $D \rightarrow E$

6. $D \rightarrow F$

Traže se FZ koje mogu da se izvedu iz drugih ili da se redukuju sa leve strane.

1)

FZ1 $AB \rightarrow D$ može da se dobije preko FZ4 $A \rightarrow D$:

1. $A \rightarrow D$ (dato)
2. $AB \rightarrow D$ (refleksivnost 1)

Tako da se ona može ukloniti.

2)

Iz FZ3 $AE \rightarrow B$ može da se ukoni E sa leve strane preko FZ4 $A \rightarrow D$ i FZ5 $D \rightarrow E$:

1. $A \rightarrow D$ (dato)
2. $D \rightarrow E$ (dato)
3. $A \rightarrow E$ (tranzitivnost 1 i 2)
4. $A \rightarrow A$ (samo određenje)
5. $A \rightarrow AE$ (unija 3 i 4)
6. $AE \rightarrow B$ (dato)
7. $A \rightarrow B$ (tranzitivnost 5 i 6)

Pa se E uklanja iz ove FZ.

Kako dalje redukcije nisu moguće dobija se sledeći nereducibilan skup FZ:

- $B \rightarrow C$
- $A \rightarrow B$
- $A \rightarrow D$
- $D \rightarrow E$
- $D \rightarrow F$

■

■ **Primer 8.22** Neka je data relacija $R = \{A, B, C, D, E\}$ i skup F funkcionalnih zavisnosti:

- $A \rightarrow B$
- $AB \rightarrow D$
- $C \rightarrow AD$
- $C \rightarrow E$

Odrediti nereducibilni skup funkcionalnih zavisnosti relacije R .

Rešenje:

Dekompozicijom se dobijaju FZ koje sa desne strane imaju samo po jedan atribut:

1. $A \rightarrow B$
2. $AB \rightarrow D$

3. $C \rightarrow A$
4. $C \rightarrow D$
5. $C \rightarrow E$

Traže se FZ koje mogu da se izvedu iz drugih ili da se redukuju sa leve strane.

1)

Iz FZ2 može da se ukloni B sa leve strane preko FZ1 $A \rightarrow B$:

1. $A \rightarrow B$ (dato)
2. $A \rightarrow AB$ (proširenje A nad 1)
3. $AB \rightarrow D$ (dato)
4. $A \rightarrow D$ (tranzitivnost 2 i 3)

Tako da se B uklanja iz ove FZ.

2)

FZ4 $C \rightarrow D$ može da se ukloni uz pomoć tranzitivnosti FZ2 $A \rightarrow D$ i FZ3 $C \rightarrow A$:

1. $C \rightarrow A$ (dato)
2. $A \rightarrow D$ (dato)
3. $C \rightarrow D$ (tranzitivnost 1 i 2)

Pa se ona uklanja.

Kako dalje redukcije nisu moguće dobija se sledeći nereducibilan skup FZ:

- $A \rightarrow B$
- $A \rightarrow D$
- $C \rightarrow A$
- $C \rightarrow E$

■

8.1.9 Projekcija funkcionalnih zavisnosti

Neka se relacija R sa skupom funkcionalnih zavisnosti S projektuje na relaciju $R_1 = p\{R\}$. Postavlja se pitanje koje funkcionalne zavisnosti su važeće u R_1 .

Određuje se projekcija od S koja sadrži sve funkcionalne zavisnosti takve da:

- mogu da se izvedu iz S
- uključuju jedino attribute iz R_1

Algoritam određivanja projekcije skupa funkcionalnih zavisnosti

Neka je R_1 projekcija relacije R , tada se projekcija T skupa funkcionalnih zavisnosti S relacije R određuje na sledeći način:

- Neka je T skup funkcionalnih zavisnosti u R_1 . Inicijalno je T prazan skup.
- Za svaki podskup X atributa iz R_1 odrediti zatvorenje X^+ u odnosu na skup S . Atributi koji su u R ali ne i u R_1 mogu da se koriste u izračunavanju X^+ . Dodati u T sve netrivialne funkcionalne zavisnosti $X \rightarrow A$ takve da A pripada zatvorenju X^+ i A pripada R_1 .
- Minimalan skup T se određuje na sledeći način:
 - Ako postoji t iz T koja može da se izvede iz drugih u T , ukloniti t
 - Neka je $Y \rightarrow B$ iz T sa najmanje dva atributa i neka je Z dobijeno iz Y uklanjanjem jednog od atributa. Ako $Z \rightarrow B$ može da se izvede iz funkcionalnih zavisnosti u T (uključujući $Y \rightarrow B$), tada se $Y \rightarrow B$ zamenjuje sa $Z \rightarrow B$.
 - Ponoviti prethodne korake sve dok ima promena u T .

■ **Primer 8.23** Neka $R\{A, B, C, D\}$ sadrži funkcionalne zavisnosti:

- $A \rightarrow B$
- $B \rightarrow C$
- $C \rightarrow D$

Neka je $R_1(A, C, D) = p(R)$. Odrediti skup funkcionalnih zavisnosti relacije R_1 .

Rešenje:

Inicijalno je skup funkcionalnih zavisnosti T relacije R_1 prazan.

Određuju se zatvorenja podskupova X iz R_1 , odnosno sledeća zatvorenja:

$$\{A\}^+ = \{A, B, C, D\},$$

$$\{C\}^+ = \{C, D\},$$

$$\{D\}^+ = \{D\},$$

$$\{A, C\}^+ = \{A, B, C, D\},$$

$$\{A, D\}^+ = \{A, B, C, D\},$$

$$\{C, D\}^+ = \{C, D\}.$$

Dodaju se u T sve netrivialne funkcionalne zavisnosti $X \rightarrow A$ takve da A pripada zatvorenju X^+ i A pripada R_1 .

Dakle, u T se dodaju FZ:

- $A \rightarrow C$
- $A \rightarrow D$
- $C \rightarrow D$
- $AC \rightarrow D$
- $AD \rightarrow C$

Sada treba izračunati nereducibilan skup FZ koji je ekvivalentan skupu T .

1)

FZ4 $AC \rightarrow D$ može da se dobije kompozicijom iz FZ2 $A \rightarrow D$ i FZ3 $C \rightarrow D$:

1. $A \rightarrow D$ (dato)
2. $C \rightarrow D$ (dato)
3. $AC \rightarrow D$ (kompozicija 1 i 2)

Tako da ona može da se ukloni.

2)

FZ5 $AD \rightarrow C$ može da se dobije iz FZ1 $A \rightarrow C$:

1. $A \rightarrow C$ (dato)
2. $AD \rightarrow C$ (refleksivnost 1)

Tako da i ona može da se ukloni.

3)

FZ2 $A \rightarrow D$ može da dobije pravilom tranzitivnosti preko FZ1 $A \rightarrow C$ i FZ3 $C \rightarrow D$.

1. $A \rightarrow C$ (dato)
2. $C \rightarrow D$ (dato)
3. $A \rightarrow D$ (tranzitivnost 1 i 2)

Pa se može ukloniti.

Kako se više ni jedna FZ ne može ukloniti niti redukovati, nereducibilan skup FZ koji je ekvivalentan skupu T je:

- $A \rightarrow C$
- $C \rightarrow D$

Dobijeni skup je projekcija skupa funkcionalnih zavisnosti relacije R koji važi na projekciji $R_1(A, C, D)$ skupa R .

■

8.2 Višeznačne zavisnosti

Funkcionalne zavisnosti su specijalni slučaj opštijih logičkih veza među atributima relacije, koje se zovu višeznačne zavisnosti (VZ). Opštost ovog tipa logičke veze ogleda se u sledećem: kod funkcionalne zavisnosti skupa atributa Y od skupa atributa X ($X \rightarrow Y$), vrednost skupa atributa X određuje jednu vrednost skupa atributa Y , dok kod višeznačne zavisnosti Y od X (u oznaci $X \twoheadrightarrow Y$), vrednost skupa atributa X određuje skup vrednosti skupa atributa Y [6].

Neka su X, Y i Z neprazni skupovi atributa relacije R , pri čemu je $Z = R \setminus XY$ i neka su x, z proizvoljni elementi projekcija relacije R na skupove atributa X, Z , redom.

Neka je $Y_{xz} = \{y | y \in R[Y] \text{ i } xyz \in R\}$. Y_{xz} je skup svih elemenata projekcije relacije R na

skupu atributa Y koji su u relaciji R sa x, z , tj. Y_{xz} je projekcija na skup atributa Y restrikcije relacije R po uslovu $X = x$ i $Z = z$.

U relaciji R važi višeznačna zavisnost skupa Y od skupa X , u oznaci $X \twoheadrightarrow Y$, ako važi jednakost $Y_{xz} = Y_{xz'}$ za svako $x \in R[X]$ i $z, z' \in R[Z]$ za koje su $Y_{xz}, Y_{xz'}$ oba neprazna.

U relaciji R postoji višeznačna zavisnost $X \twoheadrightarrow Y$ ako pripadnost n -torki $xyz, xy'z' \in R$, za $x \in R[X]$, $y, y' \in R[Y]$ i $z, z' \in R[Z]$ povlači i pripadnost n -torki $xyz', xy'z \in R$.

Takođe, višeznačna zavisnost $X \twoheadrightarrow Y$ važi u R ako je skup Y_{xz} jednoznačno određen vrednošću x , tj. zavisi samo od x a ne i od z .

Ako u relaciji R važi višeznačna zavisnost $X \twoheadrightarrow Y$, gde su X, Y neprazni podskupovi skupa atributa od R , onda u relaciji R važi i višeznačna zavisnost $X \twoheadrightarrow Z$, gde je $Z = R \setminus XY$ neprazan skup.

Dekompozicija: Neka je R relacija i neka su X, Y, Z neprazni podskupovi skupa atributa od R , pri čemu je $Z = R \setminus XY$. Za relaciju R važi jednakost $R = R[XY] * R[XZ]$ ako i samo ako važi višeznačna zavisnost $X \twoheadrightarrow Y$.

■ **Primer 8.24** Neka su dati atributi *predmet*, *nastavnik* i *udzbenik* relacije *Program*.

Neka važi pravilo da svaki predmet može da predaje skup nastavnika i da za svaki predmet postoji skup udžbenika koji se može koristiti na tom predmetu nezavisno od toga koji nastavnik ga predaje, kao u tabeli 8.2.

predmet	nastavnik	udzbenik
p1	n1	u1
p1	n1	u2
p1	n2	u1
p1	n2	u2
p3	n3	u3
p4	n3	u4
p5	n5	u5
p6	n6	u5

Tabela 8.2: Relacija *Program* - sadrži informacije o vezi između predmeta, nastavnika i udžbenika

Da li u relaciji *Program* postoje višeznačne zavisnosti i ako postoje, koje?

Rešenje:

U relaciji *Program* postoje VZ $predmet \twoheadrightarrow nastavnik$ i $predmet \twoheadrightarrow udzbenik$.

Relacija *Program* se može dekomponovati na relacije $(predmet, nastavnik)$ i $(predmet, udzbenik)$, kao u tabelama 8.3, tako da se spajanjem novodobijene dve relacije dobije natrag relacija *Program*.

■

Za postojanje višeznačne zavisnosti potrebno je da postoje bar tri atributa. Ukoliko

predmet	nastavnik
p1	n1
p1	n2
p3	n3
p4	n3
p5	n5
p6	n6

predmet	udžbenik
p1	u1
p1	u2
p3	u3
p4	u4
p5	u5
p6	u5

Tabela 8.3: Dekompozicija tabele *Program*

bi postojala samo dva atributa, na primer *predmet* i *nastavnik*, ne bi postojala višeznačna zavisnost. Takođe, ukoliko bi neki nastavnik dodao neki udžbenik za neki predmet nezavisno od drugih nastavnika, takođe ne bi postojala višeznačna zavisnost.

■ **Primer 8.25** Neka se u relaciji *Program* doda da nastavnik *n1* koristi udžbenik *u3* za predmet *p1*, kao u tabeli 8.4.

predmet	nastavnik	udžbenik
p1	n1	u1
p1	n1	u2
p1	n1	u3
p1	n2	u1
p1	n2	u2
p3	n3	u3
p4	n3	u4
p5	n5	u5
p6	n6	u5

Tabela 8.4: Nema višeznačnih zavisnosti

Ukoliko se za nastavnika *n2* ne doda da koristi taj udžbenik za predmet *p1*, tada više ne postoje višeznačne zavisnosti, jer postojanje udžbenika za predmet više nije nezavisno od nastavnika na predmetu.

■

8.3 Zavisnosti spajanja

Opštiji oblik zavisnosti među atributima relacije, u odnosu na višeznačne zavisnosti, predstavlja takozvanu zavisnost spajanja. Definicija ovog oblika logičke veze zasniva se na uopštenju svojstva dekompozicije funkcionalnih odnosno višeznačnih zavisnosti u više projekcija bez gubljenja informacija. Postoje primeri relacija koje se ne mogu binarno dekomponovati bez gubljenja informacija, ali se mogu konačno dekomponovati [6].

U relaciji *R* postoji zavisnost spajanja ako i samo ako relacija *R* može da se dobije kao rezultat iz prirodnog spajanja više njenih projekcija.

Dekompozicija: Neka je R relacija i neka su X_1, X_2, \dots, X_m neprazni podskupovi skupa atributa od R , pri čemu je $R = X_1 \cup X_2 \cup \dots \cup X_m$. Za relaciju R važi jednakost $R = R[X_1] * R[X_2] * \dots * R[X_m]$ ako i samo ako u relaciji R postoji zavisnost spajanja, u oznaci $*\{X_1, X_2, \dots, X_m\}$.

Specijalno, u relaciji R sa tri neprazna podskupa atributa X, Y, Z , pri čemu je $Z = R \setminus XY$ važi zavisnost spajanja $*\{XY, XZ\}$ ako i samo ako u relaciji R važi višeznačna zavisnost $X \rightarrow \rightarrow Y$ (samim tim i će važiti i $\forall Z X \rightarrow \rightarrow Z$).

■ **Primer 8.26** Neka je data relacija *Nastava* sa atributima *predmet*, *nastavnik* i *udzbenik*, i neka su date informacije koje predmete predaju koji nastavnici i pri tome koju literaturu koriste, kao u tabeli 8.5.

predmet	nastavnik	udzbenik
p1	n1	u1
p1	n2	u2
p3	n3	u3
p4	n3	u4
p5	n5	u5
p6	n6	u5

Tabela 8.5: Relacija *Nastava* - ima informacije o tome koje predmete predaju koji nastavnici i pri tome koju literaturu koriste

Da li postoji zavisnost spajanja u relaciji *Nastava*?

Rešenje:

Relacija *Nastava* se može dekomponovati na tri relacije: $(predmet, nastavnik)$, $(predmet, udzbenik)$ i $(nastavnik, udzbenik)$ bez gubitka informacija, kao u tabelama 8.6, tako da se spajanjem te tri relacije može dobiti ponovo relacija *Nastava*. To znači da postoji zavisnost spajanja u relaciji *Nastava*, u oznaci $*\{(predmet, nastavnik), (predmet, udzbenik), (nastavnik, udzbenik)\}$.

■

predmet	nastavnik
p1	n1
p1	n2
p3	n3
p4	n3
p5	n5
p6	n6

nastavnik	udzbenik
n1	u1
n2	u2
n3	u3
n3	u4
n5	u5
n6	u5

predmet	udzbenik
p1	u1
p1	u2
p3	u3
p4	u4
p5	u5
p6	u5

Tabela 8.6: Dekompozicija tabele *Nastava*

9. Normalizacija

Projektovanje baze podataka se može podeliti na logičko projektovanje i fizičko projektovanje. Logičko projektovanje se odnosi na modeliranje relacija, atributa, tipova, uslova ograničenja. Modeliranje relacija u logičkom projektovanju se može izvršiti:

- **Normalizacijom** - dekomponovanjem velike relacije u više malih relacija u cilju izbegavanja ponavljanja podataka i pojave različitih vrsti anomalija u podacima, o kojima će biti reči u nastavku.
- **Semantičkim modeliranjem** - upotrebom modela objekata i veza u cilju formiranja malih relacija u kojima su atributi grupisani u prirodne celine.

Ova dva načina su različiti pristupi modeliranju relacija, ali je efekat isti, odnosno biće dobijene relacije koje su oslobođene ponavljanja i anomalija u podacima.

Normalizacija je restrukturiranje modela podataka redukovanjem relacija tako da budu u najjednostavnijoj formi. Ona je ključni korak u logičkom projektovanju baze podataka. Svrha normalizacije je izbegavanje redundantnosti i pojedinih anomalija. U procesu normalizacije operator projekcije se primenjuje više puta na datu relaciju na takav način da se spajanjem projekcija može doći do početne relacije. Stoga, proces normalizacije je reverzibilan i čuva informacije, odnosno uvek je moguće da se tekući rezultat preslika unatrag do početnog stanja.

9.1 Redudantnost i anomalije

Redudantnost - ponavljaju se podaci nepotrebno, na primer, u tabeli 9.1 kontakt prodavca, adresa kupca i sekcija se ponavljaju, a nije neophodno.

Zbog ponavljanja javljaju se neke anomalije, kao što je anomalija ažuriranja.

Anomalija ažuriranja - ukoliko postoji ponavljanje istog podatka, kada je potrebno promeniti taj podatak treba ga promeniti svuda gde se pojavljuje. Na primer, u tabeli 9.1 naziv prodavca se mora izmeniti na više mesta ukoliko se prodavac nalazi u više redova.

Anomalija unošenja - ne može se uneti neka informacija ukoliko ne postoji neka druga informacija sa kojom suštinski nije vezana, ali u tabeli jeste. Na primer, u tabeli 9.1 da bi se unele informacije o artiklu, taj artikal neko prvo mora kupiti.

Anomalija brisanja - ukoliko se izbriše neka informacija onda može sa sobom povući brisanje i još neke informacije sa kojom je vezana, ali suštinski ta druga informacija je nezavisna od prve. Na primer, u tabeli 9.1 ako se obriše svaka kupovina nekog artikla, brišu se i informacije o tom artiklu.

Ime kup.	Adresa kup.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
Marko	Vuka K. 109	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0231, 0234	Sveska, Olovka	skola	Super123	super@gmail.com	100, 20
Mitar	Vuka K. 109	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
Mitar	Jovanova 100	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Veljko	Jagiceva 10	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.1: Ilustracija redundantnosti i anomalija

Zašto se redundantnost pojavljuje? Atributi koji nemaju neke prirodne veze su grupisani u jednu relaciju R . Ovaj problem se često rešava dekompozicijom. Dekompozicija podrazumeva zamenu relacije R sa nekoliko manjih relacija R_1, R_2, \dots, R_n .

9.2 Vrste funkcionalnih zavisnosti

Funkcionalna zavisnost $X \rightarrow Y$ relacije R , gde su X i Y podskupovi skupa atributa relacije R , je:

- **superključna** ili potpuna ukoliko je X ceo ključ relacije R
- **trivijalna** ukoliko je Y podskup od X
- **parcijalna** ukoliko je X pravi podskup ključa relacije R
- **tranzitivna** ukoliko postoji podskup Z skupa atributa relacije R različit od X i Y takav da važi $X \rightarrow Z$ i $Z \rightarrow Y$.

9.3 Kandidati za ključ, ključni i neključni atributi

■ **Primer 9.1** Ako je dat svih atributa relacije R , odrediti kandidate za ključ date relacije R u kojoj važe FZ is skupa S :

$R = \{\text{id_kupca, ime_kupca, adresa_kupca, sifra_artikla, ime_artikla, cena_artikla, sekcija, prodavac, kontakt_prodavca}\}.$

Skup funkcionalnih zavisnosti S je:

- $\text{id_kupca} \rightarrow \text{ime_kupca}, \text{adresa_kupca}$
- $\text{sifra_artikla} \rightarrow \text{ime_artikla}, \text{sekcija}, \text{cena_artikla}$
- $\text{ime_artikla} \rightarrow \text{sekcija}$
- $\text{prodavac} \rightarrow \text{kontakt_prodavca}$
- $\text{kontakt_prodavca} \rightarrow \text{prodavac}$

Rešenje:

Nijedan skup atributa kardinalnosti jedan, niti dva, nije kandidat za ključ. Kandidati za ključ su $\{\text{id_kupca}, \text{kontakt_prodavca}, \text{sifra_artikla}\}$ i $\{\text{id_kupca}, \text{prodavac}, \text{sifra_artikla}\}$. Bilo koji od ova dva skupa se mogu izabrati za primarni ključ relacije R, može da bude, na primer, prvi skup. ■

Ključni atribut je atribut koji se nalazi u sastavu bar nekog kandidata za ključ.

Neključni atribut je atribut koji se ne nalazi u sastavu niti jednog kandidata za ključ.

9.4 Normalne forme

Relacija je normalizovana ukoliko zadovoljava uslove koji su navedeni odgovarajućim normalnim formama (prvom, drugom, i ostalim normalnim formama). **Smatra se da je relacija dovoljno normalizovana ukoliko ispunjava uslove Bojs-Kodove normalne forme.**

9.4.1 Prva normalna forma

Definicija: Relacija je u prvoj normalnoj formi ako:

- Svako polje sadrži tačno jednu vrednost (NULL je vrednost).
- Sve vrednosti atributa su istog tipa.
- Redovi su jedinstveni.

Relacija prikazana tabelom 9.1 nije u 1NF, jer neka polja imaju više vrednosti, što se može bolje uočiti u tabeli 9.2.

Ime kup.	Adresa kup.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
Marko	Vuka K. 109	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0231, 0234	Sveska, Olovka	skola	Super123	super@gmail.com	100, 20
Mitar	Vuka K. 109	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
Mitar	Jovanova 100	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Veljko	Jagiceva 10	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.2: Višestruke vrednosti

To se može rešiti tako što se višestruke vrednosti za attribute razdvajaju u posebne vrste, kao u tabeli 9.3.

U prethodnom primeru mogu se zapaziti sledeće karakteristike:

- Kupac ima tačno jednu adresu.

Ime kup.	Adresa kup.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
Marko	Vuka K. 109	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0231	Sveska	skola	Super123	super@gmail.com	100
Mitar	Vuka K. 109	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
Mitar	Jovanova 100	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0234	Olovka	skola	Super123	super@gmail.com	20
Veljko	Jagiceva 10	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.3: Uklanjanje višestrukih vrednosti

- Šifra artikla određuje ime artikla i sekciju.
- Kontakt prodavca zavisi od naziva prodavca.
- Ime artikla nije jedinstveno.
- Ime kupca nije jedinstveno, potrebno je dodati šifru kupca.
- Različiti kupci mogu biti na istoj adresi.

Ime kup.	Adresa kup.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
Marko	Vuka K. 109	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0231	Sveska	skola	Super123	super@gmail.com	100
Mitar	Vuka K. 109	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
Mitar	Jovanova 100	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
Petar	Simina 20	0234	Olovka	skola	Super123	super@gmail.com	20
Veljko	Jagiceva 10	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.4: Izbor jedinstvenog identifikatora

Neophodno je da se doda *id* kupca, pošto ime kupca nije dobar identifikator za kupca, što je bolje istaknuto u tabeli 9.4, gde dva kupca imaju isto ime, ali različite adrese. Ovakav identifikator se naziva **surogat ključ** i njegova svrha je da zameni prirodni nedostajući ključ, kao u tabeli 9.5. Ako imamo prirodni ključ, onda nije potreban surogat ključ. Primer prirodnog ključa u ovoj tabeli je kontakt prodavca.

Id K.	Ime kup.	Adresa kup.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
m_1	Marko	Vuka K. 109	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
p_1	Petar	Simina 20	0231	Sveska	skola	Super123	super@gmail.com	100
m_2	Mitar	Vuka K. 109	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
m_3	Mitar	Jovanova 100	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
p_1	Petar	Simina 20	0234	Olovka	skola	Super123	super@gmail.com	20
v_1	Veljko	Jagiceva 10	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.5: Dodavanje surogat ključa

Skup funkcionalnih zavisnosti sada je:

- $id_kupca \rightarrow ime_kupca, adresa_kupca$
- $sifra_artikla \rightarrow ime_artikla, sekcija, cena_artikla$
- $ime_artikla \rightarrow sekcija$
- $prodavac \rightarrow kontakt_prodavca$
- $kontakt_prodavca \rightarrow prodavac$

9.4.2 Druga normalna forma

Definicija: Relacija je u drugoj normalnoj formi (2NF) ako je u prvoj normalnoj formi i ako nijedan neključni atribut nije funkcionalno zavisian od pravog podskupa kandidata ključa (ne postoje funkcionalne zavisnosti neključnih atributa od pravog podskupa ključa, tzv. parcijalne FZ).

Dekompozicija: Za svaku FZ $X \rightarrow Y$ koja narušava 2NF se izvršava dekompozicija na sledeći način: zamenjuje se relacija R dvema relacijama R_1 i R_2 , gde je $R_1 = X \cup Y$, dok je $R_2 = R \setminus Y$. FZ ostaje očuvana u relaciji R_1 . R postaje $R_1 \times R_2$.

■ **Primer 9.2** Neka je data relacija R iz prethodnog primera sa skupom atributa {id_kupca, ime_kupca, adresa_kupca, sifra_artikla, ime_artikla, cena_artikla, sekcija, prodavac, kontakt_prodavca}.

Skup funkcionalnih zavisnosti S je:

- (FZ1) $\text{id_kupca} \rightarrow \text{ime_kupca}, \text{adresa_kupca}$
- (FZ2) $\text{sifra_artikla} \rightarrow \text{ime_artikla}, \text{sekcija}, \text{cena_artikla}$
- (FZ3) $\text{ime_artikla} \rightarrow \text{sekcija}$
- (FZ4) $\text{prodavac} \rightarrow \text{kontakt_prodavca}$
- (FZ5) $\text{kontakt_prodavca} \rightarrow \text{prodavac}$

Dovesti relaciju R do 2NF.

Rešenje:

Iz prethodnog primera imamo da su kandidati za ključ:

- {id_kupca, kontakt_prodavca, sifra_artikla}
- {id_kupca, prodavac, sifra_artikla}

Funkcionalne zavisnosti koje narušavaju 2NF su FZ1 i FZ2. Potrebno je osloboditi se redom funkcionalnih zavisnosti koje narušavaju 2NF.

1)

Oslobađanje od FZ1.

Dekompozicijom se dobijaju relacije:

- **R_1** = {id_kupca, ime_kupca, adresa_kupca}
FZ1 je očuvana u relaciji R_1
- **R_2** = {id_kupca, sifra_artikla, ime_artikla, cena_artikla, sekcija, prodavac, kontakt_prodavca}
FZ2-5 su očuvane u relaciji R_2

Relacija R se može dobiti Dekartovim proizvodom ove dve relacije $R = R_1 \times R_2$, kao u tabelama 9.6 i 9.7.

2)

Id K.	Ime kup.	Adresa kup.
m_1	Marko	Vuka K. 109
p_1	Petar	Simina 20
m_2	Mitar	Vuka K. 109
m_3	Mitar	Jovanova 100
p_1	Petar	Simina 20
v_1	Veljko	Jagiceva 10

Tabela 9.6: Relacija R1

Id K.	Sifra art.	Ime artikla	Sekcija	Prodavac	Kontakt prodavca	Cena art.
m_1	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
p_1	0231	Sveska	skola	Super123	super@gmail.com	100
m_2	1111	bilijar	zabava	Aca99	aca@hotmail.com	9000
m_3	1234	Hari Potter	knjige	Super123	super@gmail.com	1000
p_1	0234	Olovka	skola	Super123	super@gmail.com	20
v_1	1112	bilijar	zabava	abc123	abc@abc.rs	2000

Tabela 9.7: Relacija R2

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- **R3** = {sifra_artikla, ime_artikla, sekcija, cena_artikla}
FZ2 i FZ3 su očuvane u relaciji R3
- **R4** = {id_kupca, sifra_artikla, prodavac, kontakt_prodavca}
FZ4 i FZ5 su očuvane u relaciji R4

Relacija R2 se može dobiti Dekartovim proizvodom ove dve relacije $R2 = R3 \times R4$, kao u tabelama 9.8 i 9.9.

Relacija R je dekomponovana na relacije R1, R3, R4, koje su u 2NF, pri čemu je $R = R1 \times R3 \times R4$ i sve FZ su očuvane.

Sifra art.	Ime artikla	Sekcija	Cena art.
1234	Hari Potter	knjige	1000
0231	Sveska	skola	100
1111	bilijar	zabava	9000
1234	Hari Potter	knjige	1000
0234	Olovka	skola	20
1112	bilijar	zabava	2000

Tabela 9.8: Relacija R3

■

9.4.3 Treća normalna forma

Definicija: Relacija je u trećoj normalnoj formi (3NF) ako je u drugoj normalnoj formi i ako nijedan neključni atribut nije funkcionalno zavisian od nekog drugog (jednog ili više)

Id K.	Sifra art.	Prodavac	Kontakt prodavca
m_1	1234	Super123	super@gmail.com
p_1	0231	Super123	super@gmail.com
m_2	1111	Aca99	aca@hotmail.com
m_3	1234	Super123	super@gmail.com
p_1	0234	Super123	super@gmail.com
v_1	1112	abc123	abc@abc.rs

Tabela 9.9: Relacija R4

neključnog atributa (ne postoje netrivialne funkcionalne zavisnosti neključnih atributa od skupa atributa koji nije ceo ključ, tzv. tranzitivne FZ neključnih atributa).

Dekompozicija: Za svaku FZ $X \rightarrow Y$ koja narušava 3NF izvršava se dekompozicija na sledeći način: zamenjuje se relacija R dvema relacijama $R1$ i $R2$, gde je $R1 = X \cup Y$, dok je $R2 = R \setminus Y$. FZ ostaje očuvana u relaciji $R1$. R postaje $R1 \times R2$.

■ **Primer 9.3** Neka je data relacija R iz prethodnog primera koja je dekomponovana na $R1$, $R3$ i $R4$. Dovedi relaciju R do 3NF.

Rešenje:

Kako je relacija već u 2NF, funkcionalna zavisnost koja narušava 3NF je FZ3.

1)

Oslobađanje od FZ3.

Dekompozicijom se dobijaju relacije:

- **R5** = {ime_artikla, sekcija}
FZ3 je očuvana u relaciji R5
- **R6** = {sifra_artikla, ime_artikla, cena_artikla}
FZ2 je očuvana u relaciji R6

Relacija $R3$ se može dobiti Dekartovim proizvodom ove dve relacije $R3 = R5 \times R6$, kao u tabelama 9.10 i 9.11.

Relacija R je dekomponovana na relacije $R1$, $R4$, $R5$, $R6$, koje su u 3NF, pri čemu je $R = R1 \times R4 \times R5 \times R6$ i sve FZ su očuvane.

Ime artikla	Sekcija
Hari Potter	knjige
Sveska	skola
bilijar	zabava
Hari Potter	knjige
Olovka	skola
bilijar	zabava

Tabela 9.10: Relacija R5

Sifra art.	Ime artikla	Cena art.
1234	Hari Potter	1000
0231	Sveska	100
1111	bilijar	9000
1234	Hari Potter	1000
0234	Olovka	20
1112	bilijar	2000

Tabela 9.11: Relacija R6

9.4.4 Bojs-Kodova normalna forma

Definicija: Relacija je u Bojs-Kodovoj normalnoj formi (BCNF) ako je u trećoj normalnoj formi i ako nijedan ključni atribut nije funkcionalno zavisian od nekog skupa atributa koji nije ceo ključ (ne postoje bilo kakve netrivialne funkcionalne zavisnosti koje ne potiču od celog ključa, tzv. bilo kakve tranzitivne FZ).

Dekompozicija: Za svaku FZ $X \rightarrow Y$ koja narušava BCNF izvršava se dekompozicija na sledeći način: zamenjuje se relacija R dvema relacijama $R1$ i $R2$, gde je $R1 = X \cup Y$, dok je $R2 = R \setminus Y$. FZ ostaje očuvana u relaciji $R1$. R postaje $R1 \times R2$.

■ **Primer 9.4** Neka je data relacija R iz prethodnog primera koja je dekomponovana na $R1$, $R4$, $R5$ i $R6$. Dovedi relaciju R do BCNF.

Rešenje:

Kako je relacija već u 3NF, funkcionalna zavisnost koja narušava BCNF je FZ4.

1)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- **R7** = {prodavac, kontakt_prodavca}
FZ4 i FZ5 ostaju očuvane u relaciji R7
- **R8** = {id_kupca, sifra_artikla, prodavac}

Relacija $R4$ se može dobiti Dekartovim proizvodom ove dve relacije $R4 = R7 \times R8$, kao u tabelama 9.12 i 9.13.

Relacija R je dekomponovana na relacije $R1$, $R4$, $R5$, $R7$, $R8$, koje su u BCNF, pri čemu je $R = R1 \times R4 \times R5 \times R7 \times R8$ i sve FZ su očuvane.

■

9.4.5 Dodatni primeri

■ **Primer 9.5** Dati su relacija $Autor = \{sifra_autora, sifra_naslova, ime_autora, redni_broj_autora\}$ i skup funkcionalnih zavisnosti F :

Prodavac	Kontakt prodavca
Super123	super@gmail.com
Super123	super@gmail.com
Aca99	aca@hotmail.com
Super123	super@gmail.com
Super123	super@gmail.com
abc123	abc@abc.rs

Tabela 9.12: Relacija R7

Id K.	Sifra art.	Prodavac
m_1	1234	Super123
p_1	0231	Super123
m_2	1111	Aca99
m_3	1234	Super123
p_1	0234	Super123
v_1	1112	abc123

Tabela 9.13: Relacija R8

- (FZ1) sifra_autora, sifra_naslova \rightarrow ime_autora, redni_broj_autora
- (FZ2) sifra_autora \rightarrow ime_autora

Dovesti relaciju do 2NF.

Rešenje:

Ključ ove relacije je {sifra_autora, sifra_naslova}.

FZ koja narušava 2NF je FZ2. Potrebno je osloboditi se FZ2.

1)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- Autor1 = {sifra_autora, ime_autora}
FZ2 ostaje očuvana u relaciji Autor1
- Autor2 = {sifra_autora, sifra_naslova, redni_broj_autora}
FZ1 ostaje očuvana u relaciji Autor2

Relacija Autor je dekomponovana na relacije Autor1, Autor2 koje su u 2NF, pri čemu je $\text{Autor} = \text{Autor1} \times \text{Autor2}$ i sve FZ su očuvane. ■

■ **Primer 9.6** Dati su relacija Naslov = {sifra_naslova, sifra_autora, redni_broj_autora, naziv_naslova, ime_autora, sifra_oblasti, naziv_oblasti} i skup funkcionalnih zavisnosti F :

- (FZ1) sifra_naslova, sifra_autora \rightarrow redni_broj_autora, naziv_naslova, ime_autora, sifra_oblasti, naziv_oblasti
- (FZ2) sifra_naslova \rightarrow naziv_naslova, sifra_oblasti
- (FZ3) sifra_autora \rightarrow ime_autora

- (FZ4) $\text{sifra_oblasti} \rightarrow \text{naziv_oblasti}$

Dovesti relaciju do 3NF.

Rešenje:

Ključ ove relacije je $\{\text{sifra_autora}, \text{sifra_naslova}\}$.

FZ koje narušavaju 3NF su FZ2, FZ3 i FZ4. FZ2 i FZ3 su parcijalne pa one narušavaju i 2NF, dok FZ4 je tranzitivna neključnih atributa i ona narušava samo 3NF.

Može se oslobađati proizvoljnim redom, a može se i prvo relacija dovesti u 2NF, pa u 3NF. Ovde će biti prikazano postupno rešenje. Dovodi se prvo relacija u 2NF oslobađanjem od FZ2 i FZ3.

1)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- $\text{Naslov1} = \{\text{sifra_naslova}, \text{naziv_naslova}, \text{sifra_oblasti}, \text{naziv_oblasti}\}$
FZ2 ostaje očuvana u relaciji Naslov1, FZ4 se premešta u relaciju Naslov1, zajedno sa atributima sa desne strane, jer bi inače bila izgubljena informacija o njoj
- $\text{Naslov2} = \{\text{sifra_naslova}, \text{sifra_autora}, \text{redni_broj_autora}, \text{ime_autora}\}$
FZ1 i FZ3 ostaju očuvane u relaciji Naslov2

Relacija Naslov je dekomponovana na relacije Naslov1, Naslov2, pri čemu je $\text{Naslov} = \text{Naslov1} \times \text{Naslov2}$ i sve FZ su očuvane.

2)

Oslobađanje od FZ3.

Dekompozicijom se dobijaju relacije:

- $\text{Naslov3} = \{\text{sifra_autora}, \text{ime_autora}\}$
FZ3 ostaje očuvana u relaciji Naslov3
- $\text{Naslov4} = \{\text{sifra_naslova}, \text{sifra_autora}, \text{redni_broj_autora}\}$
FZ1 ostaje očuvana u relaciji Naslov4

Relacija Naslov2 je dekomponovana na relacije Naslov3, Naslov4, pri čemu je $\text{Naslov2} = \text{Naslov3} \times \text{Naslov4}$ i sve FZ su očuvane.

Relacija Naslov je dekomponovana na relacije Naslov1, Naslov3, Naslov4, koje su u 2NF, pri čemu je $\text{Naslov} = \text{Naslov1} \times \text{Naslov3} \times \text{Naslov4}$ i sve FZ su očuvane.

Sada je relacija u 2NF. Potrebno je dovesti je u 3NF oslobađanjem od FZ4.

3)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- Naslov5 = {sifra_oblasti, naziv_oblasti}
FZ4 ostaje očuvana u relaciji Naslov5
- Naslov6 = {sifra_naslova, naziv_naslova, sifra_oblasti}
FZ2 ostaje očuvana u relaciji Naslov6

Relacija Naslov1 je dekomponovana na relacije Naslov5, Naslov6 koje su u 3NF, pri čemu je Naslov1 = Naslov5 x Naslov6 i sve FZ su očuvane.

Početna relacija je dekomponovana na Naslov3, Naslov4, Naslov5, Naslov6 koje su sve u 3NF, Naslov = Naslov3 x Naslov4 x Naslov5 x Naslov6 i sve FZ su očuvane.

■

■ **Primer 9.7** Neka je data relacija Pozajmica = {sifra_naslova, sifra_clana, datum_pozajmice, na_koliko_dana, sifra_knjige, naziv_naslova, sifra_oblasti, naziv_oblasti} i skup funkcionalnih zavisnosti F :

- (FZ1) sifra_naslova, sifra_clana, datum_pozajmice \rightarrow na_koliko_dana
- (FZ2) sifra_naslova, sifra_clana, datum_pozajmice \rightarrow sifra_knjige
- (FZ3) sifra_naslova, sifra_clana, datum_pozajmice \rightarrow naziv_naslova
- (FZ4) sifra_naslova, sifra_clana, datum_pozajmice \rightarrow sifra_oblasti
- (FZ5) sifra_naslova, sifra_clana, datum_pozajmice \rightarrow naziv_oblasti
- (FZ6) sifra_knjige \rightarrow sifra_naslova
- (FZ7) sifra_naslova \rightarrow sifra_oblasti
- (FZ8) sifra_naslova, sifra_clana \rightarrow sifra_naslova
- (FZ9) sifra_oblasti \rightarrow naziv_oblasti

Odrediti ključ relacije Pozajmica, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Pozajmica tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Pozajmica je {sifra_naslova, sifra_clana, datum_pozajmice}.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) - (FZ5) su superključne
- (FZ6) tranzitivna ključnih atributa
- (FZ7) parcijalna
- (FZ8) trivijalna
- (FZ9) tranzitivna neključnih atributa

FZ1-5 ne narušavaju normalne forme jer one sa leve strane imaju ceo ključ i samo elemente iz ključa tako da one treba da ostanu. FZ8 može da se ukloni pošto ne nosi nikakvu informaciju jer je trivijalna. FZ7 narušava 2NF jer je parcijalna, FZ9 narušava 3NF jer je tranzitivna neključnih atributa, dok FZ6 narušava BCNF jer je tranzitivna ključnih atributa.

1)

Oslobađanje od FZ7.

Dekompozicijom se dobijaju relacije:

- Pozajmica1 = {sifra_naslova, sifra_oblasti, naziv_oblasti}
FZ7 i FZ9 ostaju očuvane u relaciji Pozajmica1
- Pozajmica2 = {sifra_naslova, sifra_clana, datum_pozajmice, na_koliko_dana, sifra_knjige, naziv_naslova}
FZ1-5 i FZ6 ostaju očuvane u relaciji Pozajmica2

Relacija Pozajmica je dekomponovana na relacije Pozajmica1, Pozajmica2 koje su u 2NF, pri čemu je $\text{Pozajmica} = \text{Pozajmica1} \times \text{Pozajmica2}$ i sve FZ su očuvane.

Relacija je dovedena u 2NF. Potrebno je dovesti je u 3NF. FZ9 narušava 3NF.

2)

Oslobađanje od FZ9.

Dekompozicijom se dobijaju relacije:

- Pozajmica3 = {sifra_oblasti, naziv_oblasti}
FZ9 ostaje očuvana u relaciji Pozajmica3
- Pozajmica4 = {sifra_naslova, sifra_oblasti}
FZ7 ostaje očuvana u relaciji Pozajmica4

Relacija Pozajmica1 je dekomponovana na relacije Pozajmica3, Pozajmica4, pri čemu je $\text{Pozajmica1} = \text{Pozajmica3} \times \text{Pozajmica4}$ i sve FZ su očuvane.

Relacija Pozajmica je dekomponovana na relacije Pozajmica2, Pozajmica3, Pozajmica4, koje su u 3NF, pri čemu je $\text{Pozajmica} = \text{Pozajmica2} \times \text{Pozajmica3} \times \text{Pozajmica4}$ i sve FZ su očuvane.

Relacija je dovedena u 3NF. Potrebno je još dovesti je u BCNF. FZ6 narušava BCNF.

3)

Oslobađanje od FZ6.

Dekompozicijom se dobijaju relacije:

- Pozajmica5 = {sifra_naslova, sifra_knjige}
FZ6 ostaje očuvana u relaciji Pozajmica5
- Pozajmica6 = {sifra_knjige, sifra_clana, datum_pozajmice, na_koliko_dana, naziv_naslova}
FZ1-5 i FZ6 ostaju očuvane u relaciji Pozajmica6

Relacija Pozajmica2 je dekomponovana na relacije Pozajmica5, Pozajmica6 koje su u BCNF, pri čemu je $\text{Pozajmica2} = \text{Pozajmica5} \times \text{Pozajmica6}$ i sve FZ su očuvane.

Početna relacija je dekomponovana na Pozajmica3, Pozajmica4, Pozajmica5, Pozajmica6

ca6 koje su sve u BCNF, Pozajmica = Pozajmica3 x Pozajmica4 x Pozajmica5 x Pozajmica6 i sve FZ su očuvane.

■

■ **Primer 9.8** Neka je relacija Intervencija = {id_pacijenta, ime, prezime, ptt, naziv_mesta, adresa, datum, sifra_zahvata, naziv_zahvata, sifra_zuba, iznos_racuna}. Skup funkcionalnih zavisnosti F je:

- (FZ1) id_pacijenta \rightarrow ime, prezime, ptt, naziv_mesta, adresa
- (FZ2) ptt \rightarrow naziv_mesta
- (FZ3) id_pacijenta, datum \rightarrow sifra_zahvata, naziv_zahvata, sifra_zuba, iznos_racuna
- (FZ4) sifra_zahvata \rightarrow naziv_zahvata

Odrediti ključeve relacije Intervencija, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Intervencija tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Intervencija je {id_pacijenta, datum}.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) parcijalna
- (FZ2) tranzitivna neključnih atributa
- (FZ3) superključna
- (FZ4) tranzitivna neključnih atributa

Da bi relacija bila u 2NF potrebno je osloboditi se parcijalne FZ1. Zatim, kada je relacija već u 2NF, potrebno je osloboditi se tranzitivnih FZ neključnih atributa, odnosno FZ2 i FZ4. FZ3 je superključna i ona ne narušava uslove za normalne forme, tako da je ona jedina dozvoljena da ostane.

Dovodi se relacije prvo u 2NF.

1)

Oslobađanje od FZ1.

Dekompozicijom se dobijaju relacije:

- Intervencija1 = {id_pacijenta, ime, prezime, ptt, naziv_mesta, adresa}
FZ1 i FZ2 ostaju očuvane u relaciji Intervencija1
- Intervencija2 = {id_pacijenta, datum, sifra_zahvata, naziv_zahvata, sifra_zuba, iznos_racuna}
FZ3 i FZ4 ostaju očuvane u relaciji Intervencija2

Relacija Intervencija je dekomponovana na relacije Intervencija1, Intervencija2 koje su u 2NF, pri čemu je Intervencija = Intervencija1 x Intervencija2 i sve FZ su očuvane.

Relacija je dovedena u 2NF. Potrebno je dovesti u 3NF. FZ2 i FZ4 narušavaju 3NF.

2)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- Intervencija3 = {ptt, naziv_mesta}
FZ2 ostaje očuvana u relaciji Intervencija3
- Intervencija4 = {id_pacijenta, ime, prezime, ptt, adresa}
FZ1 ostaje očuvana u relaciji Intervencija4

Relacija Intervencija1 je dekomponovana na relacije Intervencija3, Intervencija4, pri čemu je $\text{Intervencija1} = \text{Intervencija3} \times \text{Intervencija4}$ i sve FZ su očuvane.

FZ4 još narušava 3NF.

3)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- Intervencija5 = {sifra_zahvata, naziv_zahvata}
FZ4 ostaje očuvana u relaciji Intervencija5
- Intervencija6 = {id_pacijenta, datum, sifra_zahvata, sifra_zuba, iznos_racuna}
FZ3 ostaje očuvana u relaciji Intervencija6

Relacija Intervencija2 je dekomponovana na relacije Intervencija5, Intervencija6, pri čemu je $\text{Intervencija2} = \text{Intervencija5} \times \text{Intervencija6}$ i sve FZ su očuvane.

Relacija Intervencija je dekomponovana na relacije Intervencija3, Intervencija4, Intervencija5, Intervencija6 koje su u 3NF, pri čemu je $\text{Intervencija} = \text{Intervencija3} \times \text{Intervencija4} \times \text{Intervencija5} \times \text{Intervencija6}$ i sve FZ su očuvane.

Relacija Intervencija je dovedena u 3NF. Ove relacije su i u BCNF, jer nema dodatnih FZ osim superključne FZ.

■

■ **Primer 9.9** Neka je data relacija Vrt = {sifra_zivotinje, ime, vrsta, datum_rodenja, mesto, zaduzena_osoba, komentar, proizvod, alternativni_proizvod, datum_dolaska, datum_odlaska}. Skup funkcionalnih zavisnosti F je:

- (FZ1) $\text{sifra_zivotinje} \rightarrow \text{ime, vrsta, datum_rodenja}$
- (FZ2) $\text{sifra_zivotinje, mesto, datum_dolaska} \rightarrow \text{datum_odlaska, zaduzena_osoba, komentar}$
- (FZ3) $\text{proizvod} \rightarrow \text{alternativni_proizvod}$

Odrediti ključeve relacije Vrt, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Vrt tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Vrt je {sifra_zivotinje, mesto, datum_dolaska, proizvod}.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) parcijalna
- (FZ2) parcijalna
- (FZ3) parcijalna

Da bi relacija bila u 2NF potrebno je osloboditi se parcijalnih FZ1, FZ2 i FZ3. Pošto nema drugih FZ osim njih, kada se one uklone, relacija će biti i u 3NF i u BCNF.

Dovodi se relacija u 2NF.

1)

Oslobađanje od FZ1.

Dekompozicijom se dobijaju relacije:

- Vrt1 = {sifra_zivotinje, ime, vrsta, datum_rodenja}
FZ1 ostaje očuvana u relaciji Vrt1
- Vrt2 = {sifra_zivotinje, mesto, zaduzena_osoba, komentar, proizvod, alternativni_proizvod, datum_dolaska, datum_odlaska}
FZ2 i FZ3 ostaju očuvane u relaciji Vrt2

Relacija Vrt je dekomponovana na relacije Vrt1, Vrt2, pri čemu je $Vrt = Vrt1 \times Vrt2$ i sve FZ su očuvane.

2)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- Vrt3 = {sifra_zivotinje, mesto, datum_dolaska, datum_odlaska, zaduzena_osoba, komentar}
FZ2 ostaje očuvana u relaciji Vrt3
- Vrt4 = {sifra_zivotinje, mesto, proizvod, alternativni_proizvod, datum_dolaska}
FZ3 ostaje očuvana u relaciji Vrt4

Relacija Vrt2 je dekomponovana na relacije Vrt3, Vrt4, pri čemu je $Vrt2 = Vrt3 \times Vrt4$ i sve FZ su očuvane.

3)

Oslobađanje od FZ3.

Dekompozicijom se dobijaju relacije:

- Vrt5 = {proizvod, alternativni_proizvod}
FZ2 ostaje očuvana u relaciji Vrt5
- Vrt6 = {sifra_zivotinje, mesto, proizvod, datum_dolaska}
FZ3 ostaje očuvana u relaciji Vrt6

Relacija Vrt4 je dekomponovana na relacije Vrt5, Vrt6, pri čemu je $Vrt4 = Vrt5 \times Vrt6$ i sve FZ su očuvane.

Relacija Vrt je dekomponovana na relacije Vrt1, Vrt3, Vrt5, Vrt6 koje su u 2NF, pri čemu je $Vrt = Vrt1 \times Vrt3 \times Vrt5 \times Vrt6$ i sve FZ su očuvane.

Relacija Vrt je dovedena u 2NF. Ove relacije su ujedno i u 3NF i u BCNF, jer nema dodatnih FZ osim superključne FZ. ■

■ **Primer 9.10** Neka je data relacija Banka = {broj_racuna, redni_broj_transakcije, stanje, status, sifra_klijenta, ime_klijenta, datum, iznos, vrsta_transakcije, naziv_vrste_transakcije}. Skup funkcionalnih zavisnosti F je:

- (FZ1) broj_racuna, redni_broj_transakcije \rightarrow stanje, status, sifra_klijenta, ime_klijenta, datum, iznos, vrsta_transakcije, naziv_vrste_transakcije
- (FZ2) broj_racuna \rightarrow stanje, status, sifra_klijenta, ime_klijenta
- (FZ3) sifra_klijenta \rightarrow ime_klijenta
- (FZ4) vrsta_transakcije \rightarrow naziv_vrste_transakcije

Odrediti ključeve relacije Banka, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Banka tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Banka je {broj_racuna, redni_broj_transakcije}.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) superključna
- (FZ2) parcijalna
- (FZ3) tranzitivna neključnih atributa
- (FZ4) tranzitivna neključnih atributa

Da bi relacija bila u 2NF potrebno je osloboditi se parcijalne FZ2. Zatim, kada je relacija već u 2NF, potrebno je osloboditi se tranzitivnih FZ neključnih atributa, odnosno FZ3 i FZ4. FZ1 je superključna i ona ne narušava uslove za normalne forme, tako da je ona jedina dozvoljena da ostane.

Dovodi se relacija u 2NF.

1)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- Banka1 = {broj_racuna, stanje, status, sifra_klijenta, ime_klijenta}
FZ2 i FZ3 ostaju očuvane u relaciji Banka1
- Banka2 = {broj_racuna, redni_broj_transakcije, datum, iznos, vrsta_transakcije, naziv_vrste_transakcije}
FZ1 i FZ4 ostaju očuvane u relaciji Banka2

Relacija Banka je dekomponovana na relacije Banka1, Banka2, koje su u 2NF, pri čemu je Banka = Banka1 x Banka2 i sve FZ su očuvane.

Relacija je dovedena u 2NF. Potrebno je dovesti je u 3NF. FZ3 i FZ4 narušavaju 3NF.

2)

Oslobađanje od FZ3.

Dekompozicijom se dobijaju relacije:

- Banka3 = {sifra_klijenta, ime_klijenta}
FZ3 ostaje očuvana u relaciji Banka3
- Banka4 = {broj_racuna, stanje, status, sifra_klijenta}
FZ2 ostaje očuvana u relaciji Banka4

Relacija Banka1 je dekomponovana na relacije Banka3, Banka4, pri čemu je Banka1 = Banka3 x Banka4 i sve FZ su očuvane.

3)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- Banka5 = {vrsta_transakcije, naziv_vrste_transakcije}
FZ4 ostaje očuvana u relaciji Banka5
- Banka6 = {broj_racuna, redni_broj_transakcije, datum, iznos, vrsta_transakcije}
FZ1 ostaje očuvana u relaciji Banka6

Relacija Banka2 je dekomponovana na relacije Banka5, Banka6, pri čemu je Banka2 = Banka5 x Banka6 i sve FZ su očuvane.

Relacija Banka je dekomponovana na relacije Banka3, Banka4, Banka5, Banka6 koje su u 3NF, pri čemu je Banka = Banka3 x Banka4 x Banka5 x Banka6 i sve FZ su očuvane.

Relacija Banka je dovedena u 3NF. Ove relacije su ujedno i u BCNF, jer nema dodatnih FZ osim superključne FZ.

■

■ **Primer 9.11** Neka je data relacija Pozoriste = {id_pozorista, naziv, adresa, broj_telefona,

ime_predstave, sifra_predstave, reditelj, zanr}. Skup funkcionalnih zavisnosti F je:

- (FZ1) $\text{id_pozorista} \rightarrow \text{naziv, adresa, broj_telefona}$
- (FZ2) $\text{broj_telefona} \rightarrow \text{adresa}$
- (FZ3) $\text{id_pozorista, sifra_predstave} \rightarrow \text{reditelj, zanr, ime_predstave}$
- (FZ4) $\text{ime_predstave} \rightarrow \text{sifra_predstave}$

Odrediti ključeve relacije Pozoriste, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Pozoriste tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Pozoriste je $\{\text{id_pozorista, sifra_predstave}\}$.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) parcijalna
- (FZ2) tranzitivna neključnih atributa
- (FZ3) superključna
- (FZ4) tranzitivna ključnih atributa

FZ1 narušava 2NF. FZ1 i FZ2 narušavaju 3NF. FZ1, FZ2 i FZ4 narušavaju BCNF.

Dovodi se prvo relacija u 2NF.

1)

Oslobađanje od FZ1.

Dekompozicijom se dobijaju relacije:

- $\text{Pozoriste1} = \{\text{id_pozorista, naziv, adresa, broj_telefona}\}$
FZ1 i FZ2 ostaju očuvane u relaciji Pozoriste1
- $\text{Pozoriste2} = \{\text{id_pozorista, ime_predstave, sifra_predstave, reditelj, zanr}\}$
FZ3 i FZ4 ostaju očuvane u relaciji Pozoriste2

Relacija Pozoriste je dekomponovana na relacije Pozoriste1, Pozoriste2 koje su u 2NF, pri čemu je $\text{Pozoriste} = \text{Pozoriste1} \times \text{Pozoriste2}$ i sve FZ su očuvane.

Relacija je dovedena u 2NF. Potrebno je dovesti u 3NF. FZ2 narušava 3NF.

2)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- $\text{Pozoriste3} = \{\text{adresa, broj_telefona}\}$
FZ2 ostaje očuvana u relaciji Pozoriste3
- $\text{Pozoriste4} = \{\text{id_pozorista, naziv, broj_telefona}\}$
FZ1 ostaje očuvana u relaciji Pozoriste4

Relacija Pozoriste1 je dekomponovana na relacije Pozoriste3, Pozoriste4, pri čemu je $\text{Pozoriste1} = \text{Pozoriste3} \times \text{Pozoriste4}$ i sve FZ su očuvane.

Relacija Pozoriste je dekomponovana na relacije Pozoriste2, Pozoriste3, Pozoriste4, koje su u 3NF, pri čemu je $\text{Pozoriste} = \text{Pozoriste2} \times \text{Pozoriste3} \times \text{Pozoriste4}$ i sve FZ su očuvane.

Relacija je dovedena u 3NF. Potrebno je dovesti u BCNF. FZ4 narušava BCNF.

3)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- $\text{Pozoriste5} = \{\text{ime_predstave}, \text{sifra_predstave}\}$
FZ4 ostaje očuvana u relaciji Pozoriste5
- $\text{Pozoriste6} = \{\text{id_pozorista}, \text{ime_predstave}, \text{reditelj}, \text{zanr}\}$
FZ3 ostaje očuvana u relaciji Pozoriste6

Relacija Pozoriste2 je dekomponovana na relacije Pozoriste5, Pozoriste6, pri čemu je $\text{Pozoriste2} = \text{Pozoriste5} \times \text{Pozoriste6}$ i sve FZ su očuvane.

Početna relacija je dekomponovana na Pozoriste3, Pozoriste4, Pozoriste5, Pozoriste6 koje su sve u BCNF, $\text{Pozoriste} = \text{Pozoriste3} \times \text{Pozoriste4} \times \text{Pozoriste5} \times \text{Pozoriste6}$ i sve FZ su očuvane.

■

■ **Primer 9.12** Neka je data relacija $\text{Projekti} = \{\text{id_sektora}, \text{sredstva_sektor}, \text{id_rukovodioca}, \text{id_radnika}, \text{id_projekta}, \text{id_kancelarije}, \text{broj_telefona}, \text{naziv_posla}, \text{sifra_posla}, \text{datum_primanja}, \text{iznos_primanja}, \text{sredstva_projekta}, \text{povrsina_kancelarije}\}$. Skup funkcionalnih zavisnosti F je:

- (FZ1) $\text{id_sektora} \rightarrow \text{sredstva_sektor}, \text{id_rukovodioca}$
- (FZ2) $\text{id_radnika} \rightarrow \text{id_sektora}, \text{id_projekta}, \text{id_kancelarije}, \text{broj_telefona}$
- (FZ3) $\text{id_radnika}, \text{naziv_posla}, \text{datum_primanja} \rightarrow \text{iznos_primanja}, \text{sifra_posla}$
- (FZ4) $\text{id_projekta} \rightarrow \text{sredstva_projekta}$
- (FZ5) $\text{id_kancelarije} \rightarrow \text{povrsina_kancelarije}$
- (FZ6) $\text{sifra_posla} \rightarrow \text{naziv_posla}$

Odrediti ključeve relacije Projekti, identifikovati vrste funkcionalnih zavisnosti, a zatim transformisati postupno relaciju Projekti tako da bude u 2NF, pa u 3NF i na kraju u BCNF.

Rešenje:

Ključ relacije Projekti je $\{\text{id_radnika}, \text{naziv_posla}, \text{datum_primanja}\}$.

Vrste funkcionalnih zavisnosti su sledeće:

- (FZ1) tranzitivna neključnih atributa

- (FZ2) parcijalna
- (FZ3) superključna
- (FZ4) tranzitivna neključnih atributa
- (FZ5) tranzitivna neključnih atributa
- (FZ6) tranzitivna ključnih atributa

FZ2 narušava 2NF. FZ2, FZ1, FZ4 i FZ5 narušavaju 3NF. FZ2, FZ1, FZ4, FZ5 i FZ6 narušavaju BCNF. FZ3 jedina može da ostane jer je superključna i ne utiče na normalne forme.

Dovodi se relacija u 2NF.

1)

Oslobađanje od FZ2.

Dekompozicijom se dobijaju relacije:

- Projekti1 = {id_radnika, id_sektora, sredstva_sektor, id_rukovodioca, id_projekta, sredstva_projekta, id_kancelarije, površina_kancelarije, broj_telefona}
FZ1, FZ2, FZ4 i FZ5 ostaju očuvane u relaciji Projekti1
- Projekti2 = {id_radnika, naziv_posla, sifra_posla, datum_primanja, iznos_primanja}
FZ6 ostaje očuvana u relaciji Projekti2

Relacija Projekti je dekomponovana na relacije Projekti1, Projekti2, pri čemu je Projekti = Projekti1 x Projekti2 i sve FZ su očuvane.

Relacija je dovedena u 2NF. Potrebno je dovesti je u 3NF. FZ1, FZ4 i FZ5 narušavaju 3NF.

2)

Oslobađanje od FZ1.

Dekompozicijom se dobijaju relacije:

- Projekti3 = {id_sektora, sredstva_sektor, id_rukovodioca}
FZ1 ostaje očuvana u relaciji Projekti3
- Projekti4 = {id_radnika, id_sektora, id_projekta, sredstva_projekta, id_kancelarije, površina_kancelarije, broj_telefona}
FZ2, FZ4 i FZ5 ostaju očuvane u relaciji Projekti4

Relacija Projekti1 je dekomponovana na relacije Projekti3, Projekti4, pri čemu je Projekti1 = Projekti3 x Projekti4 i sve FZ su očuvane.

3)

Oslobađanje od FZ4.

Dekompozicijom se dobijaju relacije:

- Projekti5 = {id_projekta, sredstva_projekta}
FZ4 ostaje očuvana u relaciji Projekti5
- Projekti6 = {id_radnika, id_sektora, id_projekta, id_kancelarije, površina_kancelarije, broj_telefona}
FZ2, FZ5 ostaju očuvane u relaciji Projekti6

Relacija Projekti4 je dekomponovana na relacije Projekti5, Projekti6, pri čemu je Projekti4 = Projekti5 x Projekti6 i sve FZ su očuvane.

4)

Oslobađanje od FZ5.

Dekompozicijom se dobijaju relacije:

- Projekti7 = {id_kancelarije, površina_kancelarije}
FZ5 ostaje očuvana u relaciji Projekti7
- Projekti8 = {id_radnika, id_sektora, id_projekta, id_kancelarije, broj_telefona}
FZ2 ostaje očuvana u relaciji Projekti8

Relacija Projekti6 je dekomponovana na relacije Projekti7, Projekti8, pri čemu je Projekti6 = Projekti7 x Projekti8 i sve FZ su očuvane.

Relacija Projekti je dekomponovana na relacije Projekti2, Projekti3, Projekti5, Projekti7 i Projekti8 koje su sve u 3NF, pri čemu je Projekti = Projekti2 x Projekti3 x Projekti5 x Projekti7 x Projekti8 i sve FZ su očuvane.

Relacija je dovedena u 3NF. FZ6 narušava BCNF. Potrebno je dovesti je u BCNF.

5)

Oslobađanje od FZ6.

Dekompozicijom se dobijaju relacije:

- Projekti9 = {naziv_posla, sifra_posla}
FZ6 ostaje očuvana u relaciji Projekti9
- Projekti10 = {id_radnika, sifra_posla, datum_primanja, iznos_primanja}

Relacija Projekti2 je dekomponovana na relacije Projekti9, Projekti10, pri čemu je Projekti2 = Projekti9 x Projekti10 i sve FZ su očuvane.

Relacija Projekti je dekomponovana na relacije Projekti3, Projekti5, Projekti7, Projekti8, Projekti9 i Projekti10 koje su sve u BCNF, pri čemu je Projekti = Projekti3 x Projekti5 x Projekti7 x Projekti8 x Projekti9 x Projekti10 i sve FZ su očuvane.

■

9.4.6 Četvrta normalna forma

Definicija: Relacija je u četvrtoj normalnoj formi (4NF) ako je u BCNF i ako ne postoje višeznačne zavisnosti.

Dekompozicija: Za $VZ X \rightarrow\rightarrow Y$ izvršava se dekompozicija na sledeći način: zamenjuje se relacija R dvema relacijama R_1 i R_2 takvim da je $R_1 = X \cup Y$ i $R_2 = R \setminus Y$, pri čemu važi $R = R_1 \bowtie R_2$.

Za primer dekompozicije pogledati 8.24.

9.4.7 Peta normalna forma

Definicija: Relacija je u petoj normalnoj formi ukoliko je u 4NF i nema drugih zavisnosti spajanja.

Dekompozicija: Za svaku ZS izvršava se dekompozicija na sledeći način: zamenjuje se relacija R relacijama R_1 i R_2, \dots, R_m takvim da je $R = * \{R_1, R_2, R_m\}$.

Za primer dekompozicije pogledati 8.26.

9.5 Dobre i loše strane dekompozicije

Dobre strane dekompozicije su smanjenje redundantnosti i pojedinih anomalija. Sa manjim relacijama je jednostavnije raditi jer imaju manje atributa, a atributi su grupisani po srodnosti.

Na primer, podaci o studentima su u relaciji DOSIJE, podaci o predmetima su u relaciji PREDMET, dok su podaci o ispitima, koji povezuju studenta sa predmetom, u zasebnoj relaciji ISPIT. Ako je potrebno pročitati osnovne informacije o studentu, nisu potrebne informacije i o njegovim položenih ispitima, pa nema potrebe ni pristupati svim tim podacima.

Sa druge strane, ukoliko je često potrebno posmatrati podatke iz više relacija koje treba spajati, onda se takođe često i vrši spajanje relacija, koje je pak skupa operacija.

9.6 Denormalizacija

Denormalizacija je namerno dupliranje podataka koje povećava redundantnost.

Kada je poznato radno opterećenje, odnosno količina i priroda podataka u tabelama, upiti koji se izvršavaju, važnost upita, ciljne performanse, može se razmisliti denormalizacija u cilju poboljšanja performansi. Ukoliko se neki atributi često koriste zajedno, iako su normalizacijom smešteni u različite relacije, može se razmotriti njihovo dupliranje kako bi se izbeglo često spajanje relacija.

10. Pretraživanje podataka

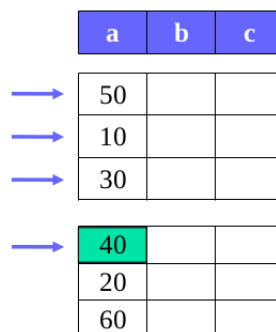
10.1 Uvod

Kako relativno brzo locirati tražene podatke?

■ **Primer 10.1** Pronaći red u relaciji T u kome atribut a ima vrednost 40.

```
select * from T where a = 40
```

Ukoliko nemamo dodatne mehanizme, linearnim pretraživanjem bi se čitao jedan po jedan slog, što znači da bi se u proseku pročitalo 50% slogova, odnosno 50% blokova na disku (slika 10.1).



	a	b	c
→	50		
→	10		
→	30		
→	40		
	20		
	60		

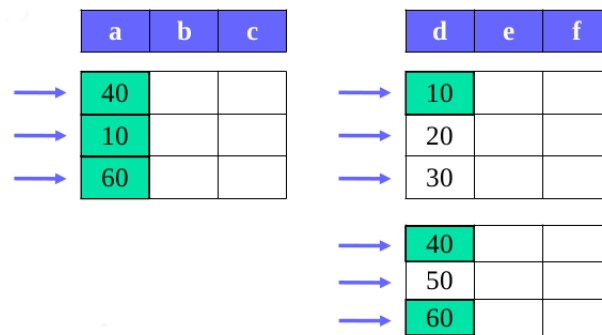
Slika 10.1: Pretraga linearne složenosti

Pristup disku je relativno skupa operacija, a svakako višestruko skuplja od pristupa lokacijama u glavnoj memoriji. Zgodnije bi bilo da se može locirati blok u kome se nalazi traženi red i da se samo taj blok učita u glavnu memoriju.

■ **Primer 10.2** Pronaći sve redove dobijene spajanjem relacija R i S tako da je atribut a iz R jednak atributu d iz S .

```
select * from R, S where R.a = S.d
```

Linearnim pretraživanjem za svaku torku iz R se traži torka u S po tačnoj vrednosti (slika 10.2. Odnosno, za svaku torku iz R se čita u proseku 50% redova iz S , što je pretraga kvadratne složenosti.



Slika 10.2: Pretraga kvadratne složenosti

Potreban je mehanizam za ubrzanje pretraživanja torke sa konkretnom vrednošću atributa.

Najvažnije merilo pri projektovanju baze podataka su performanse pri izvođenju upita koji se najčešće izvršavaju i uobičajenih operacija ažuriranja. Prvi korak pri postizanju dobrih performansi je dobar dizajn. Nakon projektovanja konceptualne sheme, što obuhvata kreiranje skupova relacija, pogleda, ograničenja, bazu podataka treba i fizički projektovati.

Fizički dizajn treba biti vođen prirodom podataka i načinom na koji će se oni koristiti. Važno je razumeti radno opterećenje (eng. workload) koji baza podataka mora podržati, a koji se sastoji od kombinacije upita i operacija ažuriranja. Treba razmisliti o korisničkim zahtevima, koliko želimo da se neki upiti brzo izvršavaju ili koliko mislimo da ćemo imati transakcija u sekundi.

Opis radnog opterećenja treba da obuhvati sledeće:

- Listu upita i njihove učestalosti u odnosu na sve upite i ažuriranja
- Listu ažuriranja i njihove učestalosti
- Cilj performansi za svaki tip upita ili ažuriranja.

Za svaki upit se mora odrediti:

- Kojim se relacijama pristupa

- Koji se atributi traže u SELECT iskazu
- Nad kojim atributima se vrši spajanje ili selekcija u WHERE iskazu i koliko su ti uslovi selektivni.

Za svako ažuriranje se mora odrediti:

- Nad kojim atributima se vrši spajanje ili selekcija u WHERE iskazu i koliko su ti uslovi selektivni
- Tip ažuriranja (INSERT, DELETE, UPDATE) i relacija koja se ažurira
- Za UPDATE komandu, polje koje treba ažurirati.

Efikasnost ažuriranja se može poboljšati dobrim izborom indeksa ali se mora razmišljati i o ažuriranju indeksa pri promeni podataka.

Važne odluke koje treba napraviti prilikom fizičkog projektovanja uključuju sledeće:

- Koje indese napraviti:
 - Koje relacije indeksirati i koje polje ili koja polja izabrati za ključ pretrage indeksa
 - Za svaki indeks odrediti da li treba biti klasterovan ili ne, da li treba biti redak ili gust.
- Da li treba praviti sledeće promene u konceptualnoj shemi u cilju poboljšanja performansi:
 - Alternativne normalizovane sheme
 - Denormalizacija
 - Vertikalno particionisanje
 - Horizontalno particionisanje
 - Pogledi
- Da li treba preformulisati upite koji se često izvršavaju tako da budu efikasniji?

Detaljnije informacije o radnom opterećenju je teško postići samo na osnovu početnog dizajna. Kasnije podešavanje sistema na osnovu konkretne upotrebe je takođe veoma značajno.

Ovo poglavlje će se baviti indeksima kao mehanizmom za poboljšanje performansi baze podataka.

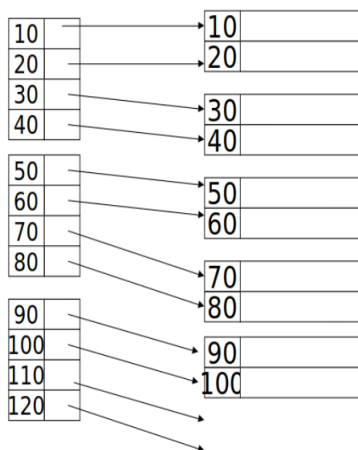
10.2 Indeksi

Indeks nad atributom *A* je struktura podataka koja omogućava jednostavno pronalaženje svih elemenata koji imaju fiksiranu vrednost atributa *A*.

Indeks se pravi nad jednim ili više atributa neke relacije. Za atribut koji će se indeksirati se bira atribut po kome se vrši pretraga redova te relacije. Pretraga se može izvoditi u cilju dobavljanja podataka, izmene podataka ili spajanja relacija.

Indeks se čuva u indeksnoj datoteci koja se sastoji od ključa pretrage i liste pokazivača na slogove sa odgovarajućim vrednostima za attribute koji su u ključu pretrage, kao na slici

10.3. Ova datoteka je uvek sortirana po ključu pretrage.



Slika 10.3: Indeksna datoteka

Da bi se našao slog sa konkretnom vrednošću atributa po kome postoji indeks, potrebno je pretražiti odgovarajuću indeksnu datoteku, koja je obično znatno manja od datoteke sa slogovima. Indeksna datoteka je sortirana po atributu pretrage, tako da je omogućeno binarno pretraživanje, pa je traženje pokazivača na odgovarajuće slogove logaritamske složenosti.

Indeksi povećavaju memorijske zahteve pošto je potrebno odvojiti dodatan prostor za smeštanje indeksne datoteke. Takođe, indeksi povećavaju i složenost izmene relacija jer kod promena sadržaja relacija potrebno je ažurirati i indekse.

10.2.1 Primarni i sekundarni indeks

Prema skupu atributa nad kojima se formira indeks, on može biti primarni ili sekundarni.

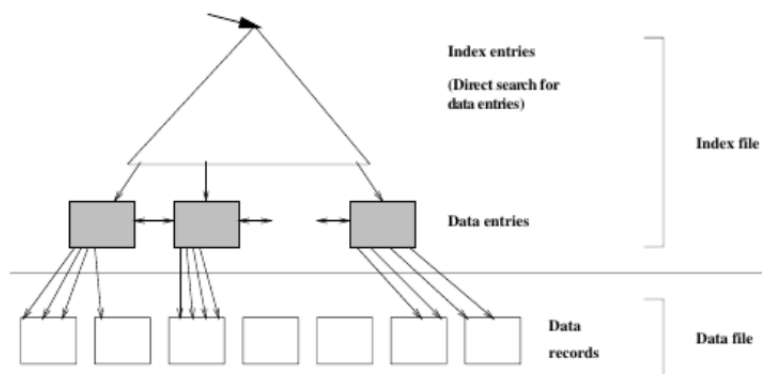
Primarni indeks je indeks nad atributima koji su primarni ključ. Primarni ključ je jedinstven, tako da su i vrednosti u indeksu jedinstvene.

Sekundarni indeks je indeks nad atributima koji nisu primarni ključ. Može forsirati jedinstvenost, ali ne mora. U ovom indeksu su dozvoljeni duplikati.

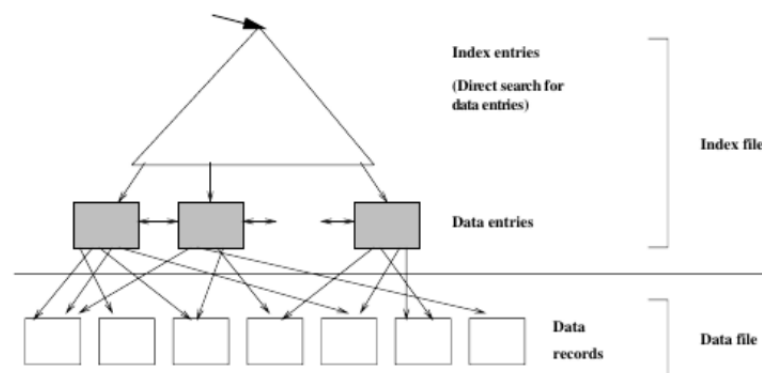
10.2.2 Klasterovan indeks

Kada je datoteka sa podacima organizovana tako da redosled slogova odgovara redosledu slogova u indeksu, kao na slici 10.4, tada se za indeks kaže da je **klasterovan**. Sa druge strane, ukoliko je redosled slogova u datoteci sa podacima takav da ne odgovara redosledu slogova u indeksu, kao na slici 10.5, tada je indeks **neklasterovan**.

Datoteka sa podacima može biti sortirana samo po jednom ključu pretrage, odnosno može imati jedan klasterovani indeks, dok može imati i više neklasterovanih indeksa.



Slika 10.4: Klasterovan indeks



Slika 10.5: Neklasterovan indeks

10.2.3 Gusti i retki indeksi

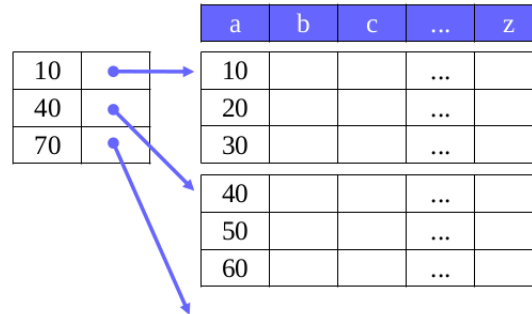
Prema upotrebi pokazivača u indeksnoj datoteci, indeksi se mogu podeliti na guste i retke.

Za indeks se kaže da je **gust** ukoliko sadrži po jedan slog za svaki slog u datoteci sa podacima, kao na slici 10.6.

		a	b	c	...	z
10	→	10			...	
20	→	20			...	
30	→	30			...	
40	→	40			...	
50	→	50			...	
60	→	60			...	

Slika 10.6: Gusti indeks

Indeks je **redak** ako sadrži po jedan slog za svaku stranicu datoteke sa podacima, kao na slici 10.7. Ne može se napraviti redak indeks koji nije klasterovan, stoga može postojati samo jedan redak indeks.



Slika 10.7: Redak indeks

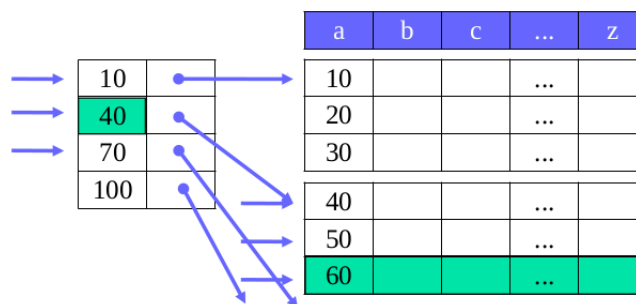
Ukoliko je potrebno naći slog sa ključem K, koraci koji se izvršavaju za gust indeks su:

- Pretražuje se indeks i nalazi ključ koji je jednak K.
- Čita se blok na koji pokazuje nađeno polje u indeksu.
- Direktno se pristupa slogu iz bloka pošto je poznata adresa na njega.

Ukoliko je potrebno naći slog sa ključem K, koraci koji se izvršavaju za redak indeks su:

- Pretražuje se indeks i nalazi najveći ključ manji ili jednak K.
- Čita se blok na koji pokazuje nađeno polje u indeksu.
- Pretražuje se blok za nalaženje sloga.

■ **Primer 10.3** Neka postoji redak indeks kao na slici 10.8.

Slika 10.8: Pretraživanje sloga sa vrednošću $a = 60$ pomoću retkog indeksa

Ukoliko se postavi sledeći upit:

```
SELECT *
FROM R
WHERE a = 60
```

traži se u indeksu najveća vrednost manja ili jednaka 60, a to je 40. Ovoj vrednosti u indeksu je pridružen pokazivač na blok podataka u kome je prvi slog 40. Sekvencijalnim pretraživanjem ovog bloka podataka će se naći i slog sa vrednošću 60 traženog atributa.

■

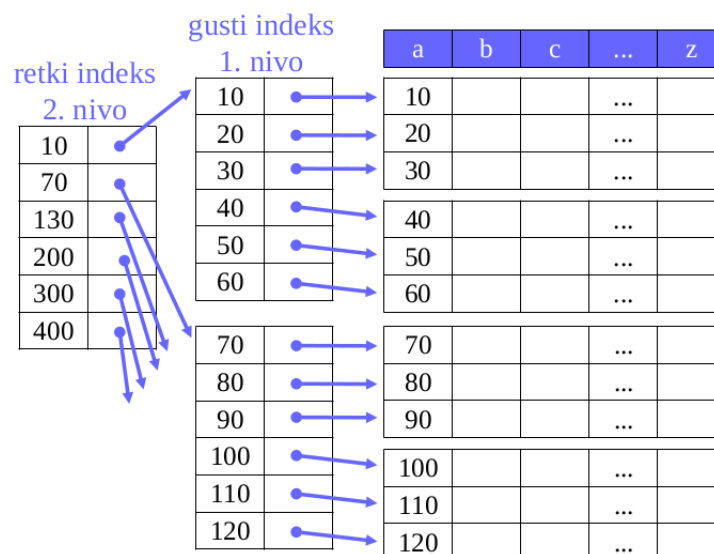
Za prebrojavanje slogova, ukoliko je dostupan gusti indeks možemo odgovoriti na upit samo pretragom indeksa i prebrojavanjem slogova u samom indeksu.

Indeks koji se koristi samo za tzv. **index-only** čitanje ne treba da bude klasterovan pošto se pomoću njega ne dobavljaju torke. Samo gusti indeksi mogu da se koriste u ove svrhe jer za svaku torku postoji i slog u indeksu.

10.2.4 Indeksi na više nivoa

Sam indeks može da se, u slučaju velikih podataka, proširi na više memorijskih blokova. Efikasnost može da se poveća upotrebom indeksa na više nivoa. Na slici 10.9 predstavljen je gusti indeks na prvom nivou i retki indeks na drugom nivou.

■ **Primer 10.4** Primer retkog indeksa nad gustim indeksom se može videti na slici 10.9.



Slika 10.9: Indeksi na više nivoa

■

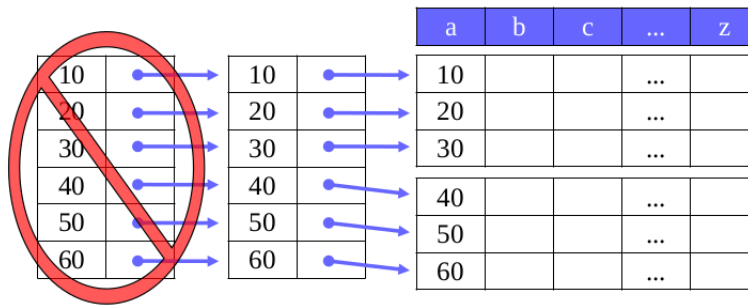
■ **Primer 10.5** Da li može da se formira gust indeks na drugom nivou, kao na slici 10.10?

Može, ali nema smisla.

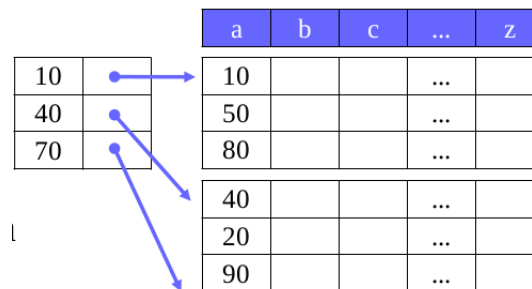
■

■ **Primer 10.6** Da li ima smisla koristiti redak indeks na nesortiranim podacima, kao na slici 10.11?

Ne, ali se može koristiti redak indeks nad gustim indeksom nad nesortiranim podacima.



Slika 10.10: Gust indeks na drugom nivou



Slika 10.11: Redak indeks nad nesortiranim podacima

■

10.2.5 Ažuriranje indeksa

Indeksna datoteka je sekvencijalna datoteka koja mora da se tretira na sličan način kao i datoteka sa sortiranim slogovima: korišćenje pomoćnog bloka (bloka prekoračenja) kod prepunjenosti tekućeg bloka, unos novih blokova, prebacivanje elemenata u odgovarajući blok.

Gust indeks pokazuje na pojedinačni slog i shodno tome modifikuje se ako je slog unet, izbrisan ili pomeren. Ne vrši se nikakva akcija u slučaju operacija nad blokovima.

Redak indeks pokazuje na blok i shodno tome može da se modifikuje ako je slog unet, izbrisan ili pomeren. Nema akcije u slučaju uključivanja bloka prekoračenja. Pokazivač mora da se unese (izbriše) ako se unosi (briše) sekvencijalni blok.

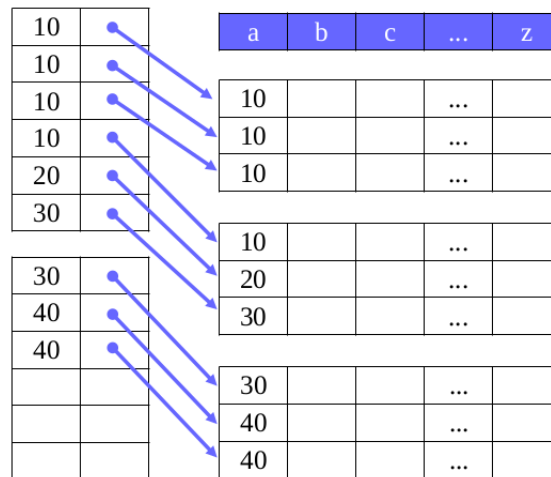
10.2.6 Pretraživanje po dupliranim vrednostima

Indeksi nad neključnim atributima mogu da sadrže duplirane vrednosti. Ako su slogovi sortirani po tim vrednostima u pitanju je klasterovani indeks.

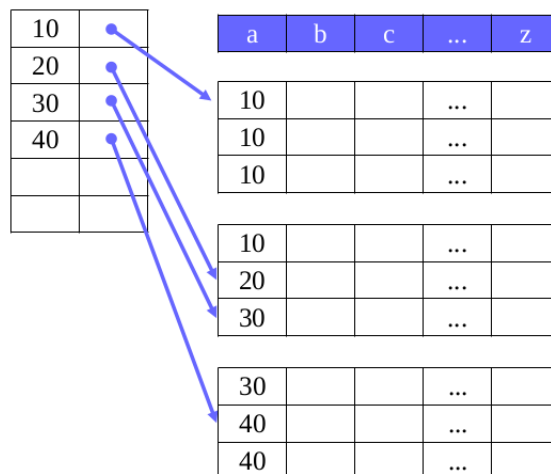
Prethodno prikazane ideje/rešenja mogu da se primene.

Više načina implementacije indeksa koji mogu da se jave:

- Gusti indeks nad jednim indeksnim poljem po slogu (pokazivač na svaki duplikat pojedinačno, slika 10.12) ili po grupi sa istim vrednostima (jedan pokazivač na prvi u grupi sa istim vrednostima, slika 10.13).
- Redak indeks, indeksno polje ima pokazivač na prvi slog u svakom bloku. Brže je pretraživanje, ali je komplikovao pronalaženje sloga, na primer, kao na slici 10.14 ako se traži slog sa ključem 30.



Slika 10.12: Duplirane vrednosti, gusti indeks za svaku vrednost

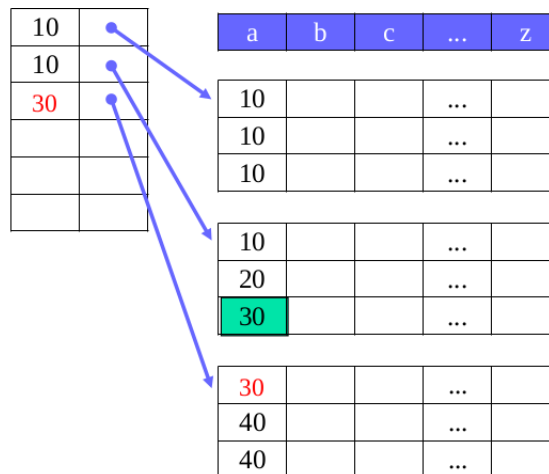


Slika 10.13: Duplirane vrednosti, gusti indeks za svaku grupu vrednosti

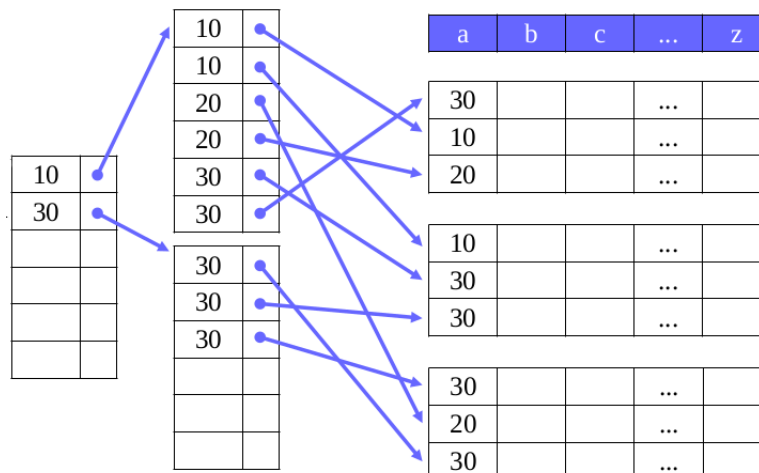
Kod sekundarnog indeksa su dozvoljene duplirane vrednosti.

■ **Primer 10.7** Primer sekundarnog indeks je dat na slici 10.15 u kome je prvi nivo gust, dok su viši nivoi retki.

Problem sa ovim rešenjem je u tome što se vrednosti u indeksu prelivaju iz jednog u drugi blok, pa tako ako se traži vrednost 30 na osnovu retkog indeksa, doći će se do bloka



Slika 10.14: Duplirane vrednosti, gusti indeks za svaku grupu vrednosti

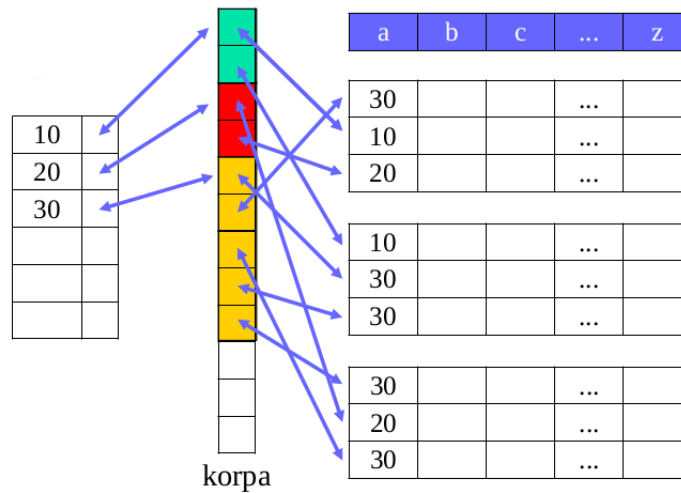


Slika 10.15: Sekundarni indeks

koji kao prvi slog ima vrednost 30, ali pretragu treba vršiti i unazad umesto samo unapred, što komplikuje situaciju. ■

Korpe (eng. buckets) su uobičajen način da se izbegne ponavljanje vrednosti.

■ **Primer 10.8** Formira se korpa datoteka, kao na slici 10.16. Vrednosti indeksa za K pokazuje na prvi element u korpi za K. Korpa datoteka se obrađuje kao i ostale sortirane datoteke. ■



Slika 10.16: Korpa datoteka

10.2.7 Uputstva pri kreiranju indeksa

Prilikom razmatranja koje indekse treba napraviti, kreće se sa listom upita, uključujući i upite koji su delovi operacija ažuriranja. Uopšteno, za dobavljanje intervala podataka se preporučuje B+ stablo, za dobavljanje konkretnih vrednosti se uobičajeno koristi heš indeks, mada se i u tim situacijama može koristiti B+ stablo. Klasterovanje će pomoći kod intervala podataka, i kod upisa konkretne vrednosti ukoliko po nekoliko zapisa ima istu vrednost za polje po kome se vrši pretraga.

Nakon sastavljanja liste željenih indeksa treba razmotriti njihov uticaj na ažuriranja u radnom opterećenju:

- Ukoliko održavanje indeksa usporava operaciju ažuriranja, razmotriti njegovo odbacivanje.
- Treba imati na umu da dodavanje indeksa može ubrzati i data ažuriranja.

10.3 B+ drveta

Često se za implementaciju indeksa koristi opšta struktura balansirano drveta (B-drveta)

- Automatski se održava odgovarajući broj nivoa.
- Održava se prostor u blokovima (obično između polovine i punog bloka).
- Drvo je balansirano – svi listovi su na istom nivou.

Postoji više različitih tipova B-drveta. U B-drvetu svi ključevi po kojima se pretražuje i odgovarajući pokazivači na podatke su predstavljeni (negde) u drvetu, dok su u B+-drvetu svi pokazivači na podatke u listovima (sortirani po vrednostima sa leva u desno).

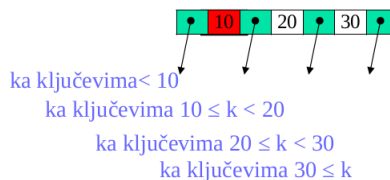
U B+ drvetu čvor ima n vrednosti (ključeva pretrage) i $n + 1$ pokazivač. U čvorovima u sredini svi pokazivači pokazuju na čvorove na nižem nivou. List sadrži n pokazivača na

podatke i 1 pokazivač na naredni čvor. U implementacijama moguće je da list sadrži $n + 2$ pokazivača: n pokazivača na podatke, pokazivač na naredni čvor i pokazivač na prethodni čvor.

Čvorovi ne mogu da budu prazni, i koriste najmanje $\text{ceil}((n + 1)/2)$ pokazivača na čvorove na narednom nivou ako su čvorovi u sredini, odnosno $\text{floor}((n + 1)/2)$ pokazivača na podatke ako su listovi.

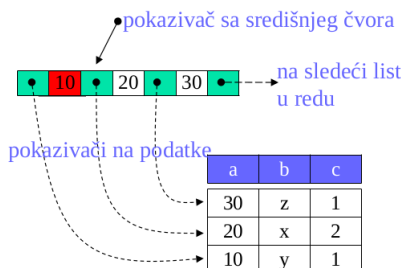
■ **Primer 10.9** Posmatrajmo drvo za $n = 3$.

Čvor u sredini za $n = 3$ prikazan je na slici 10.17. Pokazivači na levoj strani ključa pokazuju na čvor na nižem nivou sa manjom vrednošću ključa. Pokazivači na desnoj strani ključa pokazuju na čvor na nižem nivou sa istom ili većom vrednošću ključa.



Slika 10.17: Čvor u sredini za $n + 3$

List za $n = 3$ prikazan je na slici 10.18. Pokazivač na levoj strani ključa pokazuje na slog sa tom vrednošću ključa. Poslednji pokazivač pokazuje na sledeći list u redu.



Slika 10.18: List za $n = 3$

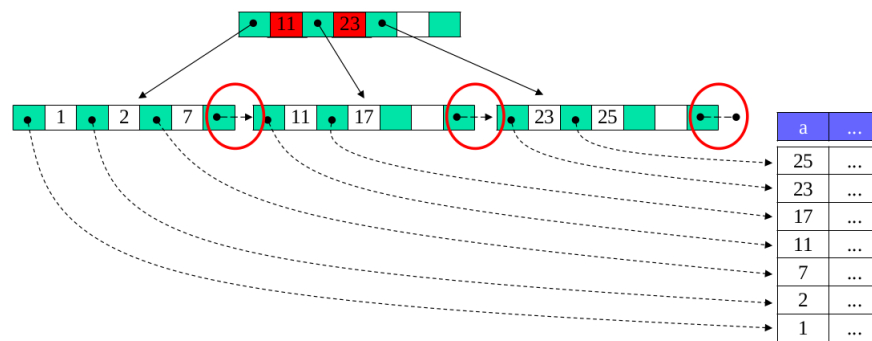
Kompletno drvo za $n = 3$ dato je na slici 10.19.

■

10.3.1 Pretraživanje podataka

Algoritam za pretragu podatka u B+ drvetu:

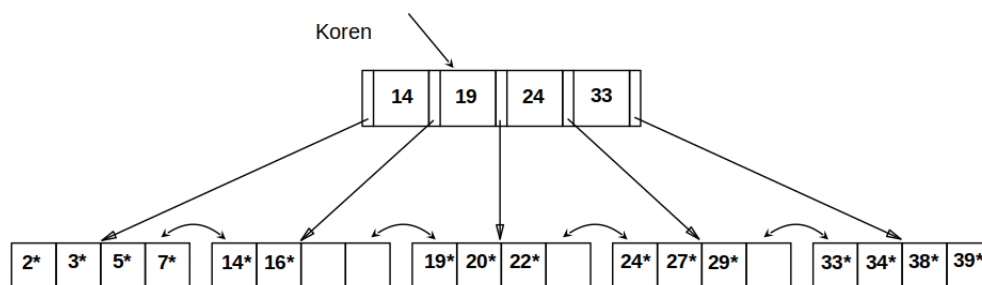
- Krenuti od korena. Naći najveću vrednost u korenu manju ili jednaku traženoj vrednosti.
- Ukoliko takva vrednost ne postoji, preći na čvor na koji pokazuje krajnji levi pokazivač.

Slika 10.19: Drvo za $n = 3$

- Ako takva vrednost postoji, preći na čvor na koji pokazuje pokazivač koji je sa desne strane nađene vrednosti.
- Pretragu rekurzivno vršiti sve dok se ne dođe do lista.
- Ukoliko se došlo do lista, pretražiti sekvencijalno list u potrazi za traženom vrednošću.
- Ukoliko vrednost postoji u identifikovanom listu, onda je nađena.
- U suprotnom, vrednost nije nađena u drvetu.

■ **Primer 10.10** Neka je dato B+ drvo kao na slici 10.20. Proveriti da li u drvetu postoje vrednosti:

- 1) 6
- 2) 15



Slika 10.20: Pretraga logaritamske složenosti

Rešenje:

1)

Traži se vrednost 6. Polazi se od korenog čvora. Nema najveće vrednosti manje ili jednake 6, tako da se pretraga nastavlja od čvora na koga pokazuje krajnji levi pokazivač, a to je list [2, 3, 5, 7]. Kako je ovo list, sekvencijalno se pretražuje, vrednost 6 se ne pronalazi.

2)

Traži se vrednost 15. Polazi se od korenog čvora. 14 je najveća vrednost manja ili jednaka 15, tako da se pretraga nastavlja od čvora na koga pokazuje pokazivač sa desne

strane vrednosti 14, a to je list [14,16]. Kako je ovo list, sekvencijalno se pretražuje, vrednost 15 se ne pronalazi.

■

10.3.2 Unošenje podataka

Algoritam za unošenje podataka u B+ drvo:

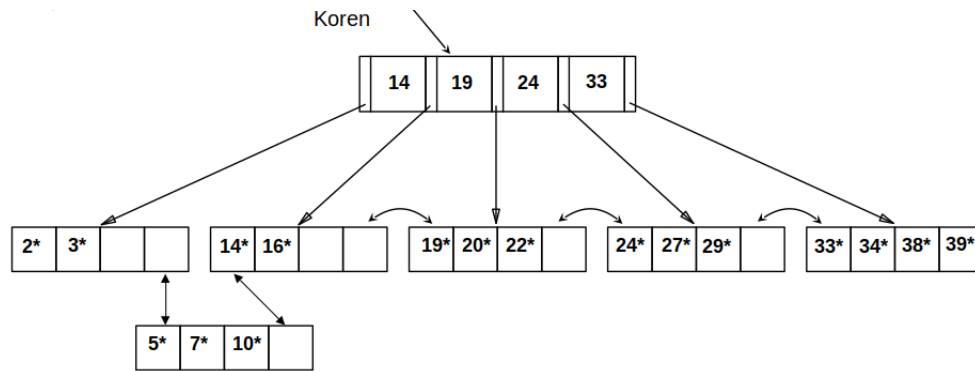
- Odrediti korektan list L .
- Upisati novu vrednost u list L .
- Ako postoji prostor unos je završen.
- Ako ne postoji prostor
 - Dodaje se novi list $L1$ i podaci iz lista L se ravnomerno raspoređuju na L i $L1$ tako da je svaki od njih pun bar do polovine kapaciteta.
 - Unosi se bočni pokazivač sa L na $L1$ i sa $L1$ na prethodni naredni od L .
 - Srednji ključ se prepisuje u čvor na nivou iznad.
 - Unosi se pokazivač na list $L1$ sa roditelj čvora lista L .
- Podela čvora na jednom nivou utiče na čvor na višem nivou ako u njega treba da se unese pokazivač na ključ. Po potrebi, rekurzivno ponoviti postupak – ako postoji prostor uneti vrednost/pokazivač, a ako ne postoji podeliti čvor.
- Ako se deli indeksni čvor, podaci se ravnomerno raspoređuju, ali se srednji ključ pomera naviše.
- Izuzetak predstavlja koreni čvor – ako treba da se u njega unese podatak a ne postoji slobodan prostor tada se koreni čvor deli na dva i formira novi koreni čvor na višem nivou sa dva dete čvora.
- Bez obzira na broj n korenom čvoru je dozvoljeno da ima samo jedan ključ i dva dete čvora.

■ **Primer 10.11** Neka je dato B+ drvo kao na slici 10.20. Uneti pokazivač na slog sa ključem 10.

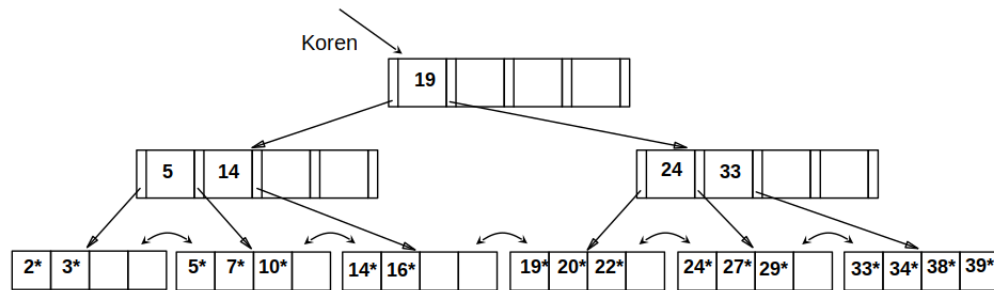
Rešenje:

- (1) Kako je $10 < 14$ unos bi trebalo da se izvrši u krajnjem levom listu.
- (2) Pošto je list put, dodaje se novi list u koji se prepisuju vrednosti [5, 7].
- (3) U novi list upisuje se vrednost 10, kao na slici 10.21.
- (4) Srednji ključ (u odnosu na oba bloka – ovde 5) se podiže u čvor naviše
- (5) Kako je koreni čvor pun, deli se na dva čvora i formira novi koreni čvor.
- (6) Podaci se ravnomerno raspoređuju (5, 14, 19, 24, 33) pri čemu se 5 i 14 upisuju u levo dete čvor, 24 i 33 u desno dete čvor, a 19 se podiže u novi koren, kao na slici 10.22.

■



Slika 10.21: Unos vrednosti u drvo



Slika 10.22: Unos vrednosti u drvo

10.3.3 Brisanje podataka

Algoritam za brisanje podataka iz B+ drveta je

- Odrediti korektan list L u kome se nalazi ključ koga treba obrisati.
- Ako posle uklanjanja ključa list ima bar polovinu elemenata, brisanje je završeno.
- U suprotnom, ako neki od susednih listova ima više od minimalnog broja ključeva i pokazivača, tada se od njega pozajmljuje par ključ/pokazivač, tako da se redosled ključeva ne menja. Pri tome se vodi računa da roditelj ključevi treba da pokazuju na novo stanje.
- Ako oba susedna lista imaju minimum parova, tada se spajaju dva susedna lista i briše jedan od njih. Broj ključeva/pokazivača u novom listu je manji od maksimuma (spojeni su listovi sa minimumom i manje od minimuma broja parova).
- Ažuriraju se ključevi roditelj čvora spojenih listova i izbriše par ključ/pokazivač u roditelj čvoru koji je pokazivao na izbrisani list.
- Ako roditelj čvor ima manje od minimalnog broja parova, postupak se rekursivno ponavlja, u suprotnom brisanje je završeno.

■ **Primer 10.12** Izbrisati ključeve 19, 20 i 22 (tim redom) iz drveta koje je rezultat prethodnog primera sa unošenjem ključa sa vrednošću 10 (slika 10.22).

Rešenje:

1)

Brisanje ključa 19 je jednostavno. Briše se vrednost 19 iz lista i u korenu se broj 19 zamenjuje brojem 20.

2)

Brisanje ključa 20 povlači pozajmicu od lista sa desne strane koji ima dovoljan broj parova ključ/pokazivač [24, 27, 29], tako da 24 prelazi u list u kome se nalazi samo 22.

Broj 20 se u korenu zamenjuje brojem 22.

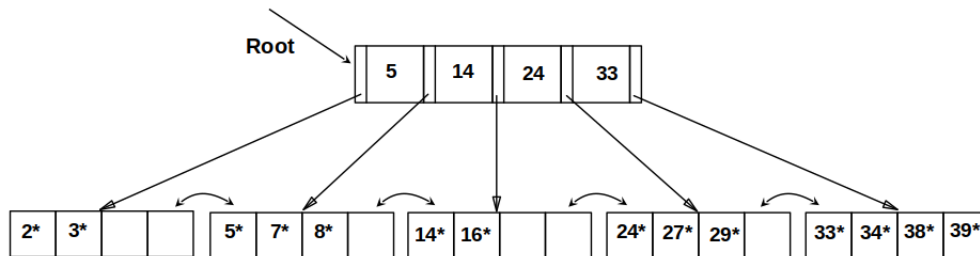
Vrši se ažuriranje ključeva u roditelj slogu – umesto 24 upisuje se prva vrednost koja je preostala u desnom listu - 27.

3)

Posle brisanja 22 ne ostaje dovoljno parova ni u jednom od susednih listova i zato se vrši spajanje lista čiji je sadržaj 24 i lista koji sadrži 27 i 29.

Ažurira se roditelj čvor; 27 se briše i kako 33 ostaje samo, a nema dovoljno parova ni u levom čvoru (sadrži 5, 14) vrši se njihovo spajanje.

Pošto koreni čvor ne može da ostane samo sa jednim dete čvorom, ponovo se primenjuje rekursivni postupak i novi koreni čvor se dobija spajanjem starog korenog čvora i jedinog dete čvora, čime se završava postupak. Dobijeno drvo je prikazano na slici 10.23.



Slika 10.23: Brisanje vrednosti iz drveta

■

10.3.4 Primene

B+-drveta mogu da budu korišćena na više načina – npr. za implementaciju indeksa nad podacima iz baze

- Listovi mogu da se ponašaju kao
 - Gusti ili retki indeksi
 - Klasterovani indeksi ili indeksi nad nesortiranim slogovima
 - Indeksi koji dopuštaju duplikate ili koji forsiraju jedinstvenost
 - ...

- Ne-list čvorovi služe za ubrzavanje pretraživanja

10.3.5 Efikasnost

Mana je što pretraživanje uvek mora da počne od korena – broj blokova koji se čitaju je jednak visini drveća plus broju pristupa slogovima.

Prednost je što je broj nivoa obično relativno mali – najčešće 3. Upiti sa intervalima se veoma brzo izvršavaju – pronađe se leva granica i zatim se vrši sekvencijalno čitanje. Ako je n dovoljno veliko, podela i spajanje će biti relativno retki. U/I operacije sa diska mogu biti smanjene ako se koreni blok stalno drži u memoriji tako da je samo mali broj U/I operacija potreban za pristup slogu.

U praksi se zbog performansi ne radi uvek podela/spajanje listova i čvorova. U slučaju velikog broja unosa/brisanja slogova potrebno je raditi eksplicitnu reorganizaciju.

11. Upravljanje transakcijama

11.1 Transakcije

Transakcija je logička jedinica posla. Može da sadrži niz operacija, ali koje bi trebalo da se izvrše atomično, odnosno ili sve da se izvrše ili nijedna da se ne izvrši. Takođe, ne smeju se naredbe izvršiti nad nekim slogovima a nad nekim ne nad kojim je planirano izvršenje, već se izvršavaju nad svim planiranim ili ni nad jednom. Deo sistema RSUBP koji se zove upravljač transakcijama obezbeđuje atomično izvršavanje operacija u transakciji. On obezbeđuje da se transakcije ne izgube, da ne budu delimično izvršene ili da ne budu izvršene više puta.

Transakcija počinje naredbom **BEGIN TRANSACTION** i završava se naredbom **COMMIT** ili **ROLLBACK**.

COMMIT signalizira uspešan kraj transakcije, govori da je baza sada u konzistentnom stanju i garantuje da će sve promene u toj transakciji biti trajne.

ROLLBACK signalizira neuspešan kraj transakcije, govori da baza može biti u ne-konzistentnom stanju, i da sve izmene koje su napravljene tom transakcijom moraju da se ponište.

■ **Primer 11.1** Na primer, neka je potrebno preneti 100 dinara sa računa 123 na račun 456. Ova operacija se sastoji iz dve radnje, skidanja sa računa 123 i dodavanje na račun 456. Te dve radnje se mogu smatrati jednom transakcijom i treba da se izvrše atomično, odnosno obe ili nijedna.

```
begin transaction;
```

```
update acc 123 {balance:= balance - $100};
```

```
if any error occurred then go to undo; end if;

update acc 456 {balance:= balance + $100};
if any error occurred then go to undo; end if;

commit;
go to finish;

undo:
    rollback;

finish:
    return;
```

■

Na početku programa ukoliko je izostavljena `BEGIN TRANSACTION`, ona se podrazumeva. Na kraju programa ukoliko je izostavljena `COMMIT` ili `ROLLBACK` naredba, `COMMIT` se podrazumeva.

U prethodnom primeru postoji provera ako se pojavila greška da se pozove `ROLLBACK`. U opštem slučaju ne može se osloniti na to da će biti eksplicitne provere, tako da postoji i **implicitni ROLLBACK** za transakcije koje ne uspeju iz bilo kog razloga da dođu do normalnog završetka, odnosno do `COMMIT`-a ili eksplicitnog `ROLLBACK`-a.

Program se sastoji od niza transakcija koje se ne preklapaju. Ugnježdene transakcije nisu podržane u većini `RSUBP`. Sistem podrazumeva da su transakcije korektne, tako da se on brine samo o tome da ispravno izvrši zadate transakcije i u tom smislu zadrži korektnost podataka u bazi. Sistem ne može da garantuje korektnost, ali može da garantuje konzistentnost u smislu zadovoljenja svih poznatih uslova ograničenja.

11.2 Oporavak

Transakcija počinje od `BEGIN TRANSACTION` ili od početka fajla, i završava se komandom `COMMIT` ili `ROLLBACK`. Istorijski, `COMMIT` je tačka u kojoj se daje do znanja da je završena jedna logička celina posla i da je baza ponovo u korektnom stanju. Komandom `ROLLBACK` se kaže da baza treba da se vrati u stanje u kom je bila, na početku transakcije, odnosno u prethodnoj `COMMIT` tački. Početak programa se u tom smislu smatra inicijalnom `COMMIT` tačkom.

U `COMMIT` tački, sva pozicioniranja (npr. kursori) se gube i svi katanci se oslobađaju (osim kod kursora sa `WITH HOLD` opcijom).

Oporavak se odnosi na oporavak same baze podataka do konzistentnog stanja nakon nekog događaja koji je prouzrokovao prekid rada. Oporavak se bazira na postojanju kopija

podataka na više mesta.

Transakcija nije samo logička jedinica posla već je i jedinica oporavka baze podataka. Pad sistema se može desiti i odmah nakon COMMIT-a, dok podaci još nisu upisani u trajnu memoriju već su na primer u baferu radne memorije. Kada dođe do pada sistema bafer radne memorije biva izgubljen. Postojanje log datoteke koja čuva sve izmene, vrste izmena, stare i nove vrednosti omogućava da se rekonstruiše stanje od prethodnog trenutka kada su izmene zaista i fizički upisane na trajnu memoriju. Ovo implicira da log datoteka mora da bude fizički zapisana u trajnoj memoriji. Dakle, procedura za restartovanje će rekonstruisati sve transakcije koje su se završile COMMIT-om od prethodnog čuvanja u trajnoj memoriji pa do pada sistema.

Najjednostavnije bi bilo da se operacije nezavršenih transakcija ne upisuju na disk, sve dok se ne završe, i onda ne bi bilo potrebe da se poništavaju izmene. Takođe, da se operacije transakcija završenim COMMIT-om upisuju COMMIT-om, i tada ne bi bio potrebe da se ponovo rekonstruišu ukoliko dođe do pada sistema.

Međutim, u praksi to nije izvodljivo. Prvo, možda bafer nije dovoljno veliki da sačuva sve izmene dok se ne završi transakcija, tako da izmene moraju da se upišu i pre završetka. Drugo, upis na disk nakon svake transakcije može da bude veoma neefikasno.

Write-ahead pravilo pri korišćenju log datoteke zahteva da se sve izmene upišu u log pre upisa u trajnu memoriju. Svi ostali logovi transakcije moraju da budu upisani u log datoteku pre COMMIT loga te transakcije. Obrada COMMIT-a ne sme da se desi pre nego što se COMMIT log upiše fizički u log datoteku.

Transakcije poseduju **ACID** osobine:

- **Atomičnost** (eng. atomicity) - sve naredne se izvršavaju ili ni jedna.
- **Konzistentnost** (eng. consistency) - baza nakon izvršenja transakcije treba da bude u konzistentnom stanju
- **Izolovanost** (eng. isolation) - transakcije nemaju uvid u međurezultate drugih transakcija u toku izvršavanja
- **Trajnost** (eng. durability) - po uspešnom završetku transakcije promene ostaju u bazi i mogu se izmeniti samo drugom transakcijom

11.2.1 Oporavak sistema

Sistem mora da ima mehanizam za oporavak ne samo od lokalnih neuspeha kao što je prekoračenje neke promenljive za jednu transakciju, već i od globalnih stanja kao što je nestanak struje, što utiče na sve transakcije koje su u opticaju.

Globalni neuspesi mogu biti:

- **Pad sistema** - na primer, nestanak struje, zove se još i laki pad (eng. soft crash)
- **Pad medija** - na primer, otkazivanje diska, zove se još i teški pad (eng. hard crash)

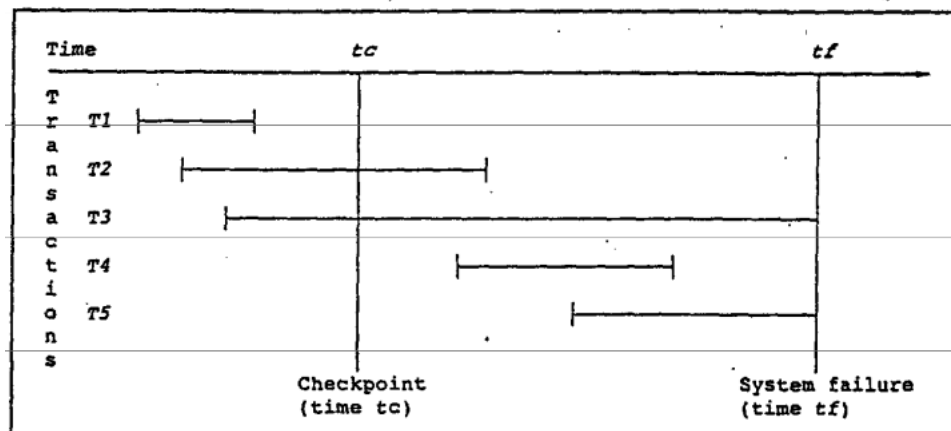
Glavna karakteristika pada sistema je u tome što se pri padu gubi sadržaj glavne memorije i bafera glavne memorije.

Postoje dve operacije koje se izvršavaju u fazi oporavka, a to su undo i redo. Operacija **redo** ponovo izvršava zadate korake. Operacija **undo** poništava izvršavanje zadatih koraka. Podsetimo, za svaki korak se u log datoteci upisuju prethodna i stara vrednosti, tako da undo zapravo vraća vrednost na prethodnu.

U određenim intervalima, uobičajeno posle određenog broja slogova upisanih u log, sistem automatski u log upisuje po jedan slog koji označava **tačku pamćenja** (eng. checkpoint). Sve operacije iz loga od prethodne tačke pamćenja do tekuće tačke pamćenja upisuje u trajnu memoriju.

Algoritam za oporavak

Postoji pet tipova transakcija u log datoteci u trenutku pada, što je prikazano na slici 11.1.



Slika 11.1: Log datoteka, pet tipova transakcija

U trenutku restartovanja sistema:

- Formiraju se dve liste transakcija **REDO** i **UNDO**.
- U listu UNDO transakcija dodaju se sve transakcije koje su bile aktivne u poslednjoj tački pamćenja (poslednjem checkpoint-u).
- Od poslednje tačke pamćenja ide se napred log datotekom i radi se sledeće:
 - Za svaki upisani BEGIN TRANSACTION log dodaje se transakcija u listu UNDO transakcija.
 - Za svaki upisani COMMIT log premešta se transakcija iz UNDO u REDO.
- Kada se stigne do kraja log datoteke, kreće se natrag ka poslednjoj tački pamćenja i za svaku transakciju iz liste UNDO izvršava se undo operacija.
- Kreće se od poslednje tačke pamćenja ka kraju log datoteke i za svaku iz liste REDO izvršava se redo operacija.

Dakle, za neuspele transakcije ili transakcije koje su prekinute padom, ne radi se ponovno izvršavanje, već se radi poništavanje efekata njihovih izvršenja.

■ **Primer 11.2** Ukoliko posmatramo log datoteku i transakcije na slici 11.1, u listu UNDO prvo se upisuju T2 i T3. Zatim se T2 završava pa prelazi u listu REDO. Zatim se ide ka kraju datoteke i dodaju se u listu UNDO T4 pa T5. T4 se završava pa prelazi u listu REDO.

Na kraju, u listi UNDO su T3 i T5, i za njih se izvršava undo, dok su T2 i T4 u listi REDO i za njih se izvršava redo. ■

Tek kada se potpuno završi aktivnost oporavka, sistem je spreman da nastavi sa radom.

ARIES

Raniji sistemi su prvo radili undo operacije, pa redo operacije.

Mnogi sistemi koriste shemu ARIES (eng. Algorithms for recovery and isolation exploiting semantics):

- **Analiza.** Napraviti REDO i UNDO liste
- **Redo.** Krenuti od pozicije u log datoteci koja je određena u fazi analize i dovesti bazu u stanje u kome je bila pre pada
- **Undo.** Poništiti efekte transakcija koje nisu uspele da se izvrše

11.2.2 Oporavak medija

Pad medija podrazumeva otkazivanje diska pri čemu su neki podaci sa diska bespovratno izgubljeni.

Oporavak podrazumeva kopiranje poslednje kopije baze iz arhive, a zatim korišćenje log datoteke da bi se rekonstruisale sve operacije transakcija koje su se uspešno završile od poslednjeg arhiviranja. Nema potrebe da se poništavaju neuspele transakcije pošto je taj sadržaj svakako izgubljen.

11.2.3 Dvofazni COMMIT

Dvofazni COMMIT je važan koncept kada transakcija treba da radi sa više nezavisnih upravljača resursa, pri čemu je svaki odgovoran za svoj skup resursa i upravlja svojim skupom logova za oporavak. Ovo se posebno odnosi na distribuirane sisteme.

Neka postoji transakcija kojom se šalje zahtev za ažuriranje na dve nezavisne lokacije. Tek kada se oba ažuriranja završe, transakcija se može smatrati uspešno završenom, što prouzrokuje COMMIT. I slično, ukoliko neko ažuriranje bude neuspešno, čitava transakcija se mora proglasiti neuspešnom, što prouzrokuje ROLLBACK.

Sistem kreira globalni COMMIT ili ROLLBACK, koji se obrađuje od strane systemske komponente zvane koordinator, čiji posao je da obezbedi da oba upravljača resursa urade COMMIT ili ROLLBACK, ispravno, čak i ako se desi pad. Koordinator prolazi kroz dve faze:

- **Priprema.** Koordinator upravljačima resursa najavljuje COMMIT ili ROLLBACK, što prouzrokuje upisivanje svih logova lokalnih resursa na trajnu memoriju. Upravljači

resursa šalju potvrdu ove akcije (poruku OK) ili informaciju da nisu uspešno izvršili akciju (poruku Not OK)

- **Commit.** Kada je koordinator primio odgovore od svih upravljača resursa, zahteva upisivanje u svoj log odluke zasnovane na odgovorima upravljača lokalnih resursa. Ukoliko su svi odgovorili OK, odluka je COMMIT. Ukoliko je bar neko odgovorio Not OK, odluka je ROLLBACK. U oba slučaja tek sada koordinator šalje konačan odgovor učesnicima i svaki učesnik mora uraditi COMMIT ili ROLLBACK, u svom lokalnu, kako je traženo.

Ukoliko sistem padne tokom ovog procesa i ukoliko se prilikom restarta pronade COMMIT odluka koordinadora, onda se nastavlja dalje da procedurom u skladu sa tom odlukom. U suprotnom, pretpostavlja se da je odluka ROLLBACK i u skladu sa time se nastavlja sa procedurom.

11.3 Konkurentnost

Termin konkurentnost označava činjenicu da SUBP dopušta većem broju transakcija pristup do istih podataka istovremeno.

Prednosti konkurentnog rada su kraće vreme odziva i maksimalna propusnost. Sa druge strane, potrebno je obezbediti da transakcije ne smetaju jedna drugoj.

11.3.1 Problemi u konkurentnom radu

Neki od problema konkurentnosti su:

- Problem nekonzistentne analize (eng. inconsistent analysis problem)
- Problem zavisnosti od nepotvrđenog čitanja (eng. uncommitted dependency problem)
- Problem izgubljenog ažuriranja (eng. lost update problem)
- Problem neuzastopnih čitanja (eng. non-repeatable reads problem)
- Problem pojavljivanja fantoma (eng. phantom reads problem)

Problem nekonzistentne analize

Neka je data situacija kao na slici 11.2. Neka postoje tri računa sa vrednostima 40, 50 i 30. Transakcija A čita vrednosti računa i sabira šta je pročitala do sata. Transakcija B prenosi iznos sa jednog računa na drugi. Transakcija A čita vrednost računa 1 i dobija 40, dosadašnja suma je 40. Zatim čita vrednost računa 2, dobija 50 i zaključuje da je dosadašnja suma 90. Transakcija B sa računa 3 prenosi 10 jedinica na račun 1 i završava se. Transakcija A zatim čita vrednost na račun 3, što je 20 i pogrešno zaključuje da je ukupna suma sa sva tri računa 110, umesto 120.

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
RETRIEVE ACC 1 : sum = 40	t1	RETRIEVE ACC 3
RETRIEVE ACC 2 : sum = 90	t2	UPDATE ACC 3 : 30 → 20
	t3	RETRIEVE ACC 1
	t4	UPDATE ACC 1 : 40 → 50
	t5	COMMIT
	t6	
	t7	
RETRIEVE ACC 3 : sum = 110, not 120	t8	

Slika 11.2: Problem nekonzistentne analize

Problem zavisnosti od nepotvrđenog čitanja

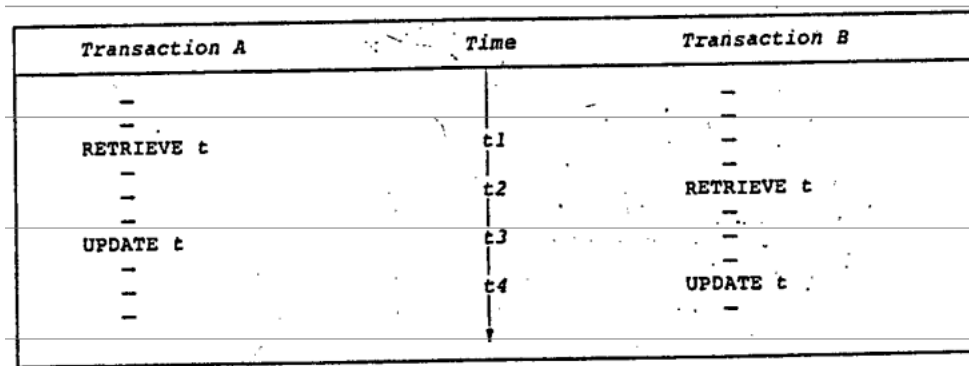
Neka je data situacija kao na slici 11.3. Transakcija *B* ažurira *t*, zatim transakcija *A* čita to ažurirano *t*, zatim transakcija *B* poništava ažuriranje, međutim transakcija *A* može dalje da radi sa vrednošću koja je poništena.

Transaction A	Time	Transaction B
RETRIEVE <i>t</i>	t1	UPDATE <i>t</i>
	t2	ROLLBACK
	t3	

Slika 11.3: Problem zavisnosti od nepotvrđenog čitanja

Problem izgubljenog ažuriranja

Neka je data situacija kao na slici 11.4. Transakcija *A* čita *t*, zatim transakcija *B* čita *t*, zatim transakcija *A* ažurira *t*, pa transakcija *B* takođe ažurira *t* ne uzimajući u obzir ažuriranje transakcije *A*. Dakle, ažuriranje transakcije *A* je izgubljeno.



Slika 11.4: Problem izgubljenog ažuriranja

Problem neuzastopnih čitanja

A čita t , a zatim B želi da izmeni t . Ukoliko B izmeni t , a zatim A pročita ponovo, dobiće drugačiju vrednost što je problem neuzastopnih čitanja.

Problem pojavljivanja fantoma

A čita neke redove iz tabele, B upisuje neke redove u istu tabelu, zatim A opet čita redove iz tabele po istom uslovu, pronalazi nove redove koji u prvom čitanju nisu bili tu, ali ih je B u međuvremenu upisala, takozvane fantomske redove.

Vrste akcija

Samo ukoliko je akcija pisanja uključena, može doći do problema u konkurentnom izvršavanju, dok ukoliko su obe akcije čitanja, problem konkurentnosti ne postoji.

Postoje četiri slučaja:

- (*čitanje, čitanje*) - Obe transakcije žele da čitaju objekat t . Ovde nema problema, pošto se ništa ne menja.
- (*čitanje, pisanje*) - A čita t , a zatim B želi da izmeni t . Ukoliko B izmeni t , može doći do problema nekonzistentne analize ili problema neuzastopnih čitanja.
- (*pisanje, čitanje*) - A menja t , a zatim B želi da čita t . U ovoj situaciji može doći do problema zavisnosti od nepotvrđenog čitanja. Ukoliko je dozvoljeno da B čita nepotvrđenu vrednost, to se još zove i **prljavo čitanje**.
- (*pisanje, pisanje*) - A menja t , a zatim B želi da menja t . U ovoj situaciji može doći do problema izgubljenog ažuriranja. Ukoliko je dozvoljeno da B menja vrednost koja može biti poništena, to se još zove i **prljavo pisanje**.

11.3.2 Zaključavanje

Prethodni problemi mogu da budu rešeni preko mehanizma koji se naziva zaključavanje ili postavljanje katanaca (eng. locking).

Neka postoje dve vrste katanaca:

- **S** - shared - deljeni, ili tzv. za čitanje. Više transakcija može imati deljeni katanac nad jednim objektom. Ovaj katanac se može dobiti ukoliko nad objektom ne postoji ni jedan katanac ili postoje samo deljeni katanci.
- **X** - exclusive - ekskluzivni, ili tzv. za pisanje. Ukoliko transakcija *A* ima ekskluzivni katanac nad objektom, ni jedna druga transakcija ne može dobiti niti deljeni niti ekskluzivni katanac, dok ga transakcija *A* ne oslobodi. Ovaj katanac se može dobiti ukoliko nad objektom ne postoji ni jedan katanac.

Matrica kompatibilnosti za navedene katance je prikazana u tabeli 11.1. U tabeli Da označava da katanci da dva tipa mogu da budu odjednom na jednom objektu, dok Ne označava da ne mogu odjednom da budu na istom objektu.

	X	S	-
X	Ne	Ne	Da
S	Ne	Da	Da
-	Da	Da	Da

Tabela 11.1: Matrica kompatibilnosti katanaca X i S

Protokol pristupa podacima, ili protokol zaključavanja, nalaže sledeće:

- Transakcija koja želi da čita objekat mora da zatraži S katanac nad tim objektom.
- Transakcija koja želi da menja objekat mora da zatraži X katanac nad objektom. Ukoliko već ima S katanac nad tim objektom, mora da zatraži promovisanje tog katanca u X katanac.
- Ukoliko traženi katanac nije dostupan odmah, transakcije se stavlja u stanje čekanja. Transakcija će biti u tom stanju dok ne bude moguće da dobije katanac, a najranije dok transakcija koja ima katanac ne oslobodi isti. Više transakcija može biti u stanju čekanja za katanac nad istim objektom, a sistem treba da obezbedi da transakcije ne čekaju zauvek. Najjednostavniji način je da se katanac dodeli po principu ko prvi zatraži prvi i dobije.
- Katanci se oslobađaju na kraju trasakcije ukoliko nisu oslobođeni pre toga.

Gornji protokol se naziva i **striktno dvofazno zaključavanje**.

11.3.3 Rešenje problema u konkurentnom radu

Prethodni protokol omogućava rešenje navedenih problema konkurentnosti. Navedimo neke od njih.

Rešenje problema nekonzistentne analize

Neka je data situacija kao na slici 11.5. Nekonzistentna analiza je izbegnuta pošto transakcija *A* ne može čitati nalog 3 sve dok transakcija *B* ne završi sa radom, ali ni

transakcija *B* ne može menjati nalog 1 sve dok transakcija *A* ne oslobodi katanac *S* nad nalogom 1. Odnosno, nastaje mrtva petlja (eng. deadlock).

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
—	—	—
RETRIEVE ACC 1 :	t1	—
(acquire S lock on ACC 1)	—	—
sum = 40	—	—
—	—	—
RETRIEVE ACC 2 :	t2	—
(acquire S lock on ACC 2)	—	—
sum = 90	—	—
—	t3	RETRIEVE ACC 3
—	—	(acquire S lock on ACC 3)
—	—	—
—	t4	UPDATE ACC 3
—	—	(acquire X lock on ACC 3)
—	—	30 → 20
—	t5	—
—	—	RETRIEVE ACC 1
—	—	(acquire S lock on ACC 1)
—	t6	—
—	—	UPDATE ACC 1
—	—	(request X lock on ACC 3)
RETRIEVE ACC 3 :	t7	wait
(request S lock on ACC 3)	—	wait
wait	—	wait
wait	—	wait

Slika 11.5: Rešenje problema nekonzistentne analize

Rešenje problema zavisnosti od nepotvrđenog čitanja

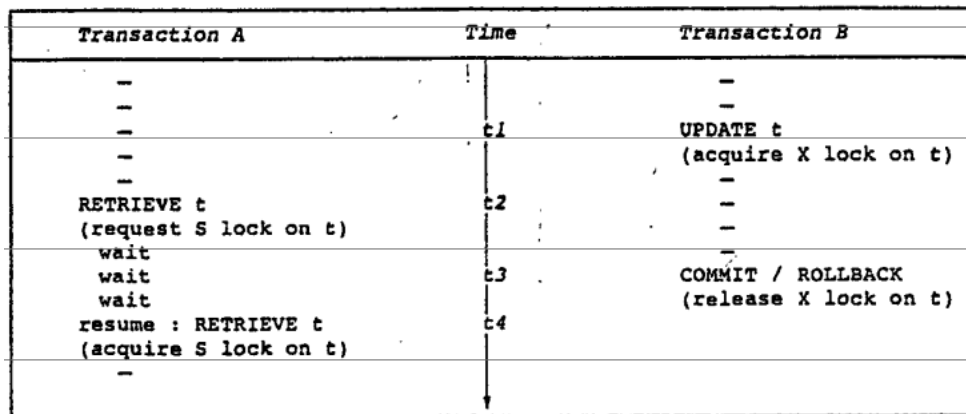
Neka je data situacija kao na slici 11.6. Transakcija *A* u trenutku *t2* ne može dobiti *S* katanac nad *t*, pa ne može ni pročitati vrednost koja nije potvrđena.

Rešenje problema izgubljenog ažuriranja

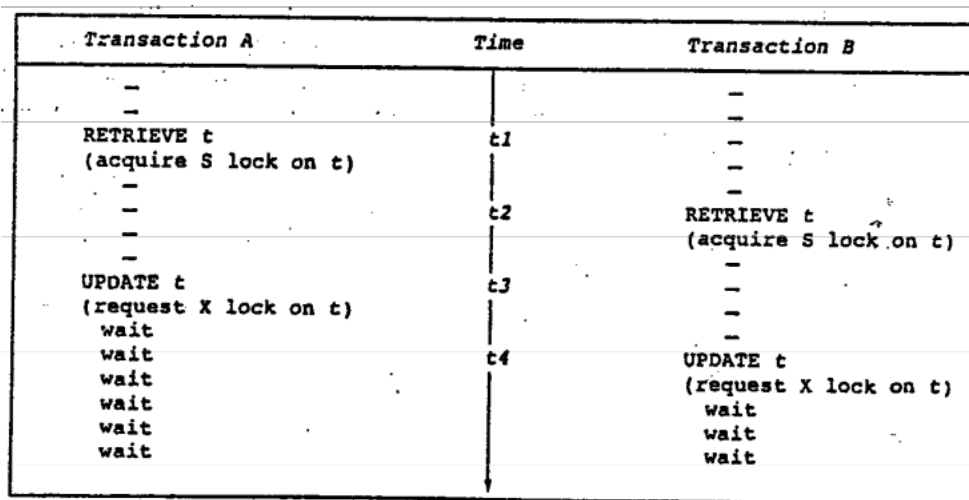
Neka je data situacija kao na slici 11.7. Transakcija *A* neće dobiti katanac *X* sve dok transakcija *B* ne oslobodi svoj katanac *S* ili posle toga katanac *X*, čime se sprečava da ažuriranje transakcije *A* bude izgubljeno. Međutim, mrtva petlja nastaje u trenutku *t4* zato što obe transakcije drže katance *S* nad *t* i traže katance *X* nad *t*, koje ne mogu dobiti dok se ne oslobode svi katanci *S* nad objektom *t*.

11.3.4 Mrtva petlja

Zaključavanje osim što rešava probleme konkurentnosti, pravi novi problem, problem mrtve petlje.



Slika 11.6: Rešenje problema zavisnosti od nepotvrđenog čitanja



Slika 11.7: Rešenje problema izgubljenog ažuriranja

Mrtva petlja je situacija u kojoj su dve ili više transakcija u stanju čekanja, gde svako od njih čeka da se oslobodi resurs koji ovi drugi imaju.

Ukoliko se pojavi mrtva petlja, poželjno je da je sistem detektuje i reši. Detektovanje mrtve petlje uključuje pronalaženje ciklusa u grafu čekanja. Razrešavanje mrtve petlje se ostvaruje tako što se neka od transakcija izabere da se prekine i izvrši poništavanje njenih akcija. Resursi koje je držala se oslobađaju i omogućava se nekim drugim transakcijama koje su čekale te resurse da nastave sa radom.

Ne koriste svi sistemi **detekciju mrtve petlje**. Neki sistemi koriste metodu **isteka vremena** (eng. timeout) tako što pretpostave da transakcija koja dugo vremena ništa nije radila je u mrtvoj petlji.

Transakcija koja je izabrana da se prekine se u nekim sistemima automatski restartuje, dok u nekim drugim se šalje izuzetak programu i program sam odlučuje da li će ponovo

pokrenuti transakciju ili ne. U svakom slučaju, problem ne treba eskalirati do krajnjeg korisnika, već ga treba obraditi na neki način.

11.3.5 Izbegavanje mrtve petlje

Mrtva petlja se može izbeći na sledeći način:

- Svaka transakcija dobija timestamp kada je počela sa radom, koje mora biti jedinstveno.
- Kada transakcija *A* zahteva katanac nad objektom koji je već zaključan od strane transakcije *B*, tada:
 - *A* čeka ukoliko je starija od *B*, u suprotom se prekida (ROLLBACK) i restartuje.
 - *A* čeka ukoliko je mlađa od *B*, u suprotnom se prekida i restartuje.
- Prilikom prekidanja i restartovanja transakcija zadržava originalni timestamp.

Bilo koja od ovih opcija će sprečiti mrtvu petlju, i svaka transakcija će sigurno uspeti da dođe do kraja eventualno. Problem sa ovim rešenjem je što ima mnogo prekida i restartovanja transakcija.

11.3.6 Serijabilnost

Serijabilnost je opšte prihvaćen kriterijum za korektnost izvršavanja skupa transakcija koje mogu da se izvršavaju isprepleteno.

Izvršenje skupa transakcija je **serijsko** kada se transakcije izvršavaju jedna za drugom bez preplitanja.

Dato izvršavanje skupa transakcija je **serijabilno** ako i samo ako je ekvivalentno nekom serijskom izvršenju skupa transakcija.

Za izvršavanje skupa transakcija se smatra da je **korektno** ukoliko je serijabilno.

Da bi izvršavanje dve konkurentne transakcije *A* i *B* bilo korektno, logički ili transakcija *A* prethodi transakciji *B*, ili je obrnuto. Odnosno, ili transakcija *A* može da ima uvid u rezultate transakcije *B*, ili transakcija *B* može da ima uvid u rezultate transakcije *A*. Ne mogu se kombinovati redosledi izvršenja transakcija nad objektima pojedinačno. Ukoliko to nije ispunjeno, raspored (eng. schedule) izvršenja nije serijabilan pa samim tim nije ni korektan.

Ukoliko sve transakcije zahtevaju dvofazno zaključavanje tada su svi mogući rasporedi izvršavanja transakcija serijabilni.

Dvofazno zaključavanje podrazumeva sledeće:

- Pre bilo koje operacije nad objektom, transakcija mora dobiti katanac nad tim objektom.
- Nakon oslobađanja katanca, transakcija ne može više bilo koje dodatne katance.

Transakcije koje se ponašaju u skladu sa ovim protokolom imaju dve faze, fazu zahteva za katancima i fazu ukidanja katanaca, odnosno oslobađanja resursa.

11.3.7 Nivoi izolacije

Objekti zaključavanja mogu biti prostor tabela, tabela, red ili konkretna vrednost.

Zbog performansi i povećanja nivoa konkurentnosti proširuje se elementarni koncept zaključavanja resursa:

- RSUBP podržava tri opšte kategorije katanaca: S, U (ažuriranje sa namerom) i X i više načina za njegovo korišćenje.
- Uvodi se koncept nivoa izolacije.
- Uvode se dodatne tehnike za povećanje nivoa konkurentnosti.

Termin nivo izolacije se koristi za opis stepena ometanja koje tekuća transakcija može da podnese pri konkurentnom izvršavanju. Realni sistemi zbog različitih razloga (npr. performanse) dopuštaju rad sa nivoom izolacije koji je manji od maksimalnog. Što je veći nivo izolacije manje su dopuštene smetnje i obratno.

Mogući nivoi izolacije u DB2 su:

- **RR** - Repeatable Read - zaključava sve slogove koji su referisani u okviru transakcije a ne samo slogove koji su rezultat upita, i to do kraja transakcije. Ponovljenim čitanjem ne može se promeniti rezultat pošto druga transakcija ne može da doda ili izmeni pročitane redove sve dok se transakcija ne završi.
- **RS** - Read Stability - zaključava slogove koji su pročitani u okviru transakcije i koji zadovoljavaju rezultat upita. Tako se rezultat upita jedne transakcije ne može promeniti drugom transakcijom sve dok se prva transakcija ne završi. Ukoliko bi se ponovo pokušalo čitanje, rezultati bi mogli da se razlikuju, jer druge transakcije mogu da upisuju ili menjaju ostale redove koji nisu bili u rezultatu prvog upita.
- **CS** - Cursor Stability (predefinisan) - zaključava se samo onaj slog kome se pristupilo dok se kursor nalazi na njemu. Katanci se drže dok se ne pristupi narednom slogu. Sve ostale slogove koje je čitala druge transakcije menjati i pre okončanja ove transakcije. I dalje transakcije ne mogu da vide rezultate izmene drugih transakcija pre završetka tih transakcija.
- **UR** - Uncommitted Read - dopušta čitanje nepotvrđenih slogova druge transakcije. Ovo je najslabiji nivo izolovanosti i dopušta drugim transakcijama da pristupe podacima koje transakcija trenutno čita kao i transakciji da pročita slogove koje su promenile transakcije koje se još nisu završile.

Katanci se drže do kraja transakcije, tj. do njene potvrde ili poništenja (osim u slučaju nivoa izolovanosti koji ne obezbeđuju u potpunosti ACID svojstva transakcija – kao što su CS i UR). Pri ovakvom pristupu dvofaznost transakcija je očita, ali je izvršenje transakcija nad istim objektima gotovo serijsko. Stoga se pribegava “usitnjavanju” transakcija, što opet povećava izgleda za uzajamno blokiranje.

Ukoliko transakcija radi sa manjim nivoom izolovanosti od maksimalnog i ukoliko se izvršava konkurentno zajedno sa drugim transakcijama, ne može se garantovati da će transformisati korektno stanje baze u drugo takođe korektno stanje baze.

Ukoliko sistem dopušta manji nivo izolovanosti, onda treba da postoji podrška da programer u aplikaciji eksplicitno može da zaključa resurs. Na primer, u DB2 postoji LOCK TABLE naredba.

11.3.8 Namera zaključavanja

Protokol zaključavanja sa namerom (eng. intent locking protocol) navodi da transakciji nije dozvoljeno da zahteva katanac nad slogom pre prvog zahteva za postavljanjem katanca uopšte.

Na taj način se konflikt između zahteva otkriva na nivou tabela, a ne na nivou slogova.

Ukoliko se postavljaju katanci nad većim objektima, na primer, tabelama, prostorima tabela ili bak čitavim bazama podataka, tada je u velikoj meri smanjena konkurentnost. Sa druge strane, ukoliko se smanji objekat postavljanja katanca, biće veliki broj katanaca i dosta proverava da li je nekome dopušteno da dobije neki katanac ili ne.

Pretpostavimo da neka transakcija T želi da dobije X katanac na nekoj relaciji R . Sistem treba da može da odgovori da li neka druga transakcija ima katanac kad bilo kojim slogom relacije R . Očigledno je nepoželjno da mora da se proveriti svaki slog da bi se dobio odgovor da li katanac može da se dobije ili ne ili da mora da se proveriti svaki postojeći katanac da bi se proverilo da li neki od njih se odnosi na relaciju R .

Uvodi se drugi protokol, protokol namere zaključavanja, prema kome ni jedna transakcija ne može da dobije katanac na slogu pre traženja tog katanca nad relacijom iz koje je taj slog. Detekcija konflikta je u ovom slučaju znatno jednostavnija zato što se radi na nivou relacija umesto na nivou slogova.

Do sada je rečeno da katanci X i S imaju smisla kako za čitave relacije, tako i za poledinačne slogove. Uvode novi katanci, takozvani katanci namere, koji takođe imaju smisla nad čitavim relacijama ali ne i nad pojedinačnim slogovima:

- **IS** - intent shared - T namerava da postavi S na pojedinačnim slogovima iz R .
- **IX** - intent exclusive - Isto kao IS, plus T će možda da ažurira individualne slogove i u tom slučaju će da traži X katanac za te slogove.
- **S** - shared - T može da toleriše konkurentno čitanje ali ne i konkurentno pisanje.
- **SIX** - shared intent exclusive - kombinuje se S i IX, T može da toleriše konkurentno čitanje ali ne i konkurentno pisanje, plus T će možda da ažurira individualne slogove i u tom slučaju će da traži X katanac za te slogove.
- **U** - Update - T može da toleriše konkurentno čitanje ali ne i konkurentno pisanje, plus T će možda da ažurira individualne slogove i u tom slučaju će da traži X katanac za te slogove.
- **X** - exclusive - T ne može da toleriše konkurentni pristup relaciji R , dok ona sama će možda a možda i ne menjati pojedinačne slogove u R .

Matrica kompatibilnosti navedenih katanaca je data u tabeli 11.2.

Transakcija dobija katance od sistema, automatski (bez eksplicitnog zahteva): kad

	IS	S	IX	SIX	U	X
IS	Da	Da	Da	Da	Da	Ne
S	Da	Da	Ne	Ne	Da	Ne
IX	Da	Ne	Da	Ne	Ne	Ne
SIX	Da	Ne	Ne	Ne	Ne	Ne
U	Da	Da	Ne	Ne	Ne	Ne
X	Ne	Ne	Ne	Ne	Ne	Ne

Tabela 11.2: Matrica kompatibilnosti načina zaključavanja

transakcija otvori kursor za čitanje, automatski dobija IS katanac nad odgovarajućom tabelom; kada uspešno obavi čitanje vrste, automatski dobija S-katanac na toj vrsti; kada otvori kursor za ažuriranje, dobija IX ili U katanac nad tabelom; kada uspešno obavi ažuriranje vrste, dobija X-katanac na vrsti, itd. Pojedinačne vrste mogu biti zaključane samo S, U ili X katancima, dok se "katanci namere" mogu postaviti samo na tabele ili prostore tabela.

11.4 Zadaci

■ **Primer 11.3** Sledeća lista prikazuje niz događaja u uzastopnim izvršavanjima DB2 transakcija T1, T2, ..., pri čemu su sve operacije sa RR nivoom izolacije kursora. A, B, ..., H, ... su slogovi, ne kursori:

1 (T1) : FETCH A	13 (T5) : ROLLBACK	25 (T9) : UPDATE H
2 (T2) : FETCH B	14 (T6) : FETCH C	26 (T6) : COMMIT
3 (T1) : FETCH C	15 (T6) : UPDATE C	27 (T11): FETCH C
4 (T4) : FETCH D	16 (T7) : FETCH G	28 (T12): FETCH D
5 (T5) : FETCH A	17 (T8) : FETCH H	29 (T12): UPDATE C
6 (T2) : FETCH E	18 (T9) : FETCH G	30 (T2) : UPDATE F
7 (T2) : UPDATE E	19 (T9) : UPDATE G	31 (T11): UPDATE C
8 (T3) : FETCH F	20 (T8) : FETCH E	32 (T12): FETCH A
9 (T2) : FETCH F	21 (T7) : COMMIT	33 (T10): UPDATE A
10 (T5) : UPDATE A	22 (T9) : FETCH H	34 (T12): UPDATE D
11 (T1) : COMMIT	23 (T3) : FETCH G	35 (T4) : FETCH G
12 (T6) : FETCH A	24 (T10): FETCH A	tn :

Da li postoji bilo kakva mrtva petlja u vremenu tn? Šemu izvođenja lepo nacrtati i obrazložiti.

Rešenje:

Raspored katanaca nad objektima tokom vremena je dat tabelom 11.3.

Značenje: **čeka na resurs**, **zahtev nije počeo da se izvršava jer transakcija čeka na resurs**.

Dakle, postoje sledeće zavisnosti (\longrightarrow interpretirati kao 'čeka na'):

12 \longrightarrow 11, 8 \longrightarrow 2, 2 \longrightarrow 3, 3 \longrightarrow 9, 4 \longrightarrow 9, 9 \longrightarrow 8.

Odakle se vidi da postoji mrtva petlja između transakcija 9 \longrightarrow 8 \longrightarrow 2 \longrightarrow 3 \longrightarrow 9, kao i dva čekanja na resurs 12 \longrightarrow 11, 4 \longrightarrow 9 koja nisu u mrtvoj petlji. ■

■ **Primer 11.4** Sledeća lista prikazuje niz događaja u uzastopnim izvršavanjima DB2 transakcija T1, T2, ..., pri čemu su sve operacije sa RR nivoom izolacije kursora. A, B, ..., H, ... su slogovi, ne kursori:

1 (T1) : FETCH A	13 (T5) : ROLLBACK	25 (T9) : FETCH A
2 (T2) : FETCH B	14 (T6) : FETCH D	26 (T9) : UPDATE H
3 (T1) : FETCH C	15 (T6) : UPDATE D	27 (T6) : COMMIT
4 (T4) : FETCH D	16 (T7) : FETCH G	28 (T11): FETCH C
5 (T5) : FETCH A	17 (T8) : FETCH H	29 (T12): FETCH D
6 (T2) : FETCH F	18 (T9) : FETCH G	30 (T12): FETCH C
7 (T2) : UPDATE F	19 (T12): FETCH A	31 (T9) : UPDATE F
8 (T3) : FETCH E	20 (T9) : UPDATE G	32 (T11): UPDATE C
9 (T2) : FETCH E	21 (T8) : FETCH E	33 (T10): UPDATE A
10 (T5) : UPDATE A	22 (T7) : COMMIT	34 (T12): UPDATE D
11 (T1) : COMMIT	23 (T9) : FETCH H	35 (T4) : FETCH G
12 (T6) : FETCH A	24 (T3) : FETCH G	tn :

Da li postoji bilo kakva mrtva petlja u vremenu tn? Šemu izvođenja lepo nacrtati i obrazložiti.

Rešenje:

Raspored katanaca nad objektima tokom vremena je dat tabelom 11.4.

Značenje: **čeka na resurs**, **zahtev nije počeo da se izvršava jer transakcija čeka na resurs**.

Dakle, postoje sledeće zavisnosti (\longrightarrow interpretirati kao 'čeka na'):

10 \longrightarrow 12, 10 \longrightarrow 9, 11 \longrightarrow 12, 12 \longrightarrow 4, 4 \longrightarrow 9, 3 \longrightarrow 9, 9 \longrightarrow 8

Nema mrtve petlje. Jedan redosled izvršavanja može da bude: 8, 9, 4, 3, 12, 10, 11. ■

t	A	B	C	D	E	F	G	H	Komentar
1	T_1f								
2		T_2f							
3			T_1f						
4				T_4f					
5	T_5f								
6					T_2f				
7					T_2U				
8						T_3f			
9						T_2f			
10	T_5U								T_5 čeka
11									T_1 COMMIT, T_5 dobija resurs
	T_5U	T_2f		T_4f	T_2U	T_3f T_2f			Trenutno stanje
12	T_6f								T_6 čeka
13									T_5 ROLLBACK, T_6 dobija resurs
	T_6f	T_2f		T_4f	T_2U	T_3f T_2f			Trenutno stanje
14			T_6f						
15			T_6U						
16							T_7f		
17								T_8f	
18							T_9f		
19							T_9U		T_9 čeka
20					T_8f				T_8 čeka
21									T_7 COMMIT, T_9 dobija resurs
	T_6f	T_2f	T_6U	T_4f	T_2U T_8f	T_3f T_2f	T_9U	T_8f	Trenutno stanje T_8 čeka
22								T_9f	T_3 čeka
23							T_3f		
24	$T_{10}f$								T_9 čeka
25								T_9U	T_6 COMMIT
26									
	$T_{10}f$	T_2f		T_4f	T_2U T_8f	T_3f T_2f	T_9U T_3f	T_8f T_9f T_9U	Trenutno stanje T_8 i T_3 čekaju T_9 čeka
27			$T_{11}f$						
28				$T_{12}f$					
29			$T_{12}U$						T_{12} čeka
30						T_2U			T_2 čeka
31			$T_{11}U$						
32	$T_{12}f$								Zahtev se ne izvršava jer $T_{12}U$ čeka C
33	$T_{10}U$								Zahtev se ne izvršava jer $T_{12}U$ čeka C
34				$T_{12}U$					T_4 čeka
35							T_4f		
	$T_{10}f$ $T_{12}f$	T_2f	$T_{11}U$ $T_{12}U$	T_4f $T_{12}f$ $T_{12}U$	T_2U T_8f	T_3f T_2f T_2U	T_9U T_3f T_4f	T_8f T_9f T_9U	Trenutno stanje T_{12} , T_8 , T_2 , T_3 , T_4 , T_9 čekaju na C, E, F, G, G, H (tim redom)

Tabela 11.3: Raspored katanaca nad objektima tokom vremena

t	A	B	C	D	E	F	G	H	Komentar
1	T_1f								
2		T_2f							
3			T_1f						
4				T_4f					
5	T_5f								
6						T_2f			
7						T_2U			
8					T_3f				
9					T_2f				
10	T_5U								T_5 čeka
11									T_1 COMMIT, T_5 dobija resurs
	T_5U	T_2f		T_4f	T_3f T_2f	T_2U			Trenutno stanje
12	T_6f								T_6 čeka
13									T_5 ROLLBACK, T_6 dobija resurs
	T_6f	T_2f		T_4f	T_3f T_2f	T_2U			Trenutno stanje
14				T_6f					T_6 čeka
15				T_6U					
16							T_7f	T_8f	
17							T_9f		
18									
19	$T_{12}f$						T_9U		T_9 čeka
20					T_8f				
21									T_7 COMMIT, T_9 dobija resurs
22									
	T_6f $T_{12}f$	T_2f		T_4f T_6f T_6U	T_3f T_2f T_8f	T_2U	T_9U	T_8f	Trenutno stanje
									T_6 čeka
23								T_9f	T_3 čeka
24							T_3f		
25	T_9f							T_9U	T_9 čeka
26									T_6 COMMIT
27									
	$T_{12}f$ T_9f	T_2f		T_4f	T_3f T_2f T_8f	T_2U	T_9U T_3f	T_8f T_9U	Trenutno stanje
									T_3, T_9 čekaju
28			$T_{11}f$						
29			$T_{12}f$	$T_{12}f$					
30									
31						T_9U			Zahtev se ne izvršava jer T_9 čeka na H
32			$T_{11}U$						T_{11} čeka
33	$T_{10}U$								T_{10} čeka
34				$T_{12}U$					T_{12} čeka
35							T_4f		T_4 čeka
	$T_{12}f$ T_9f $T_{10}U$	T_2f	$T_{11}f$ $T_{12}f$ $T_{11}U$	T_4f $T_{12}f$ $T_{12}U$	T_3f T_2f T_8f	T_2U T_9U	T_9U T_3f T_4f	T_8f T_9f T_9U	Trenutno stanje
									T_3 čeka $T_{10}, T_{11}, T_{12}, T_4, T_9$ čekaju

Tabela 11.4: Raspored katanaca nad objektima tokom vremena

12. Optimizacija

12.1 Uvod

Optimizacija upita predstavlja i izazov i mogućnost u relacionim sistemima - izazov zato što je optimizacija neophodna da bi se postigla zadovoljavajuće performanse, a mogućnost zato što je u stvari jedna od prednosti relacionih sistema to što su na dovoljnom semantičkom nivou da je optimizacija uopšte i moguća.

U nerelacionim sistemima, sa druge strane, upiti se pišu na nižem semantičkom nivou, tako da je sva optimizacija mahom u rukama korisnika koji piše upit, dok ima manje mogućnosti za automatsku optimizaciju.

Prednost automatske optimizacije se ogleda u više stvari:

- Korisnik ne mora da brine kako najbolje da formuliše upite tako da dostigne najbolje performanse.
- Dobar optimizator ima puno dostupnih informacija koje korisnik obično nema, kao što je broj različitih vrednosti svakog tipa podataka, kardinalnost relacija, broj različitih vrednosti za svaki atribut svake relacije, broj koliko puta se neka vrednost nekog tipa pojavila u svakom od atributa. Sve ove vrednosti se čuvaju u sistemskom katalogu i optimizator ih jednostavnije može koristiti da izračuna efikasnost bilo koje strategije izvršavanja upita.
- Ukoliko se statistika baze podataka promeni tokom vremena, tada će možda drugačija strategija biti bolja i možda će biti potrebna neka vrsta reorganizacije. To je jednostavnije sa automatskim optimizatorom.
- Optimizator je program koji je mnogo strpljiviji od čoveka i spremniji da ispita svaku mogućnost i svaki plan, ma koliko ih bilo.
- Optimizator je pisan od strane najiskusnijih programera iz te oblasti, i to iskustvo i znanje se može preneti ovim putem da bude dostupno svima.

Uloga optimizatora je da izabere efikasnu strategiju za izračunavanje datog relacionog izraza. Veliki udeo u tome ima proces transformacije izraza, statistika baze podataka, dekompozicija upita, način implementacije određenih operacija niskog nivoa.

■ **Primer 12.1** Neka je dat upit

```
((dosije join ispit) where id_predmeta = 2016) [ime]
```

Neka baza sadrži 100 studenata, 10000 ispita, od kojih se 50 odnosi na 2016. godinu. Neka se relacije dosije i ispit nalaze na disku, svaka relacija u po jednoj datoteci. Efikasnost neka se meri u broju U/I operacijama.

Ukoliko bi se direktno izračunavalo, spajanje relacija dosije i ispit proizvelo bi 10000 torki koje su dovoljno velike da ne mogu da stanu u glavnu memoriju i moraju da se upišu natrag na disk. Zatim, vrši se restrikcija po godini 2016, što zahteva ponovo učitavanje 10000 torki u glavnu memoriju, pri čemu se dobija rezultat od 50 torki koje sada mogu da ostanu u glavnoj memoriji. Izračunava se zatim projekcija na atribut ime.

Druga strategija bi bila da se prvo izvrši restrikcija po godini, pri čemu se dobija 50 torki koje mogu da ostanu u glavnoj memoriji. One se zatim spajaju sa dosijeom, gde se dobija 50 torki koje ponovo staju u glavnu memoriju. Nakon toga, izvršava se projekcija nad rezultatom.

Prvi pristup ima ukupno 1030000 čitanja i pisanja torki dok drugi ima samo 10100. Drugi pristup daje 100 puta brže izračunavanje. Moguće je i dalje unapređivati strategiju, ako se na primer, napravi indeks po atributu id_predmeta. ■

Problem optimizacije se može podeliti na manje ili više nezavisne potprobleme koji su prikazani u nastavku.

12.2 Faze obrade upita

Faze obrade upita su:

- Parsiranje upita i provera semantike
- Konverzija upita u kanonički oblik
- Analiza i izbor kandidata za procedure niskog nivoa
- Formiranje planova upita i izbor najjeftinijeg

12.2.1 Parsiranje upita i provera semantike

U ovoj fazi se proverava sintaksna ispravnost upita i prevodi se upit u interni zapis koji je pogodniji za obradu u računaru. Obično se koristi drvo upita ili drvo apstraktne sintakse. Interni zapis može biti, na primer, i neki od formalizama, relaciona algebra ili relacioni račun. Proverava se korektnost tipova argumenata, funkcija, korelacija, podupita i drugo.

12.2.2 Konverzija upita u kanonički oblik

Optimizator obavlja operacija za koje postoji garancija da su dobre, bez obzira na to kakvi su podaci u bazi. SQL upit je moguće zapisati na više načina. Poželjno je dovesti upit u ekvivalentan kanonički oblik koji je mnogo efikasniji. Dva upita su ekvivalentna ako i samo ako se pri njihovom izvršavanju u svim slučajevima dobija isti rezultat. Kanonički oblik bi mogao da se definiše na sledeći način.

Definicija. Neka je dat skup upita Q i pojam ekvivalencije između njih. Za podskup C skupa Q se kaže da je skup kanoničkih formi skupa Q ako i samo ako za svaki upit q iz skupa Q postoji tačno jedan ekvivalentan upit c iz skupa C . Za upit c se kaže da je kanonički oblik upita q .

Sve relevantne osobine koje važe za q , važiće i za c , pa je dovoljno raditi sa manjim skupom C kanoničkih formi, umesto sa početnim skupom Q .

12.2.3 Analiza i izbor kandidata za procedure niskog nivoa

U ovoj fazi do izražaja dolazi važnost postojanja indeksa ili drugih fizičkih putanja do podataka, distribucija vrednosti podatka, način klasterovanja i drugo.

Osnovni pristup je da se upit posmatra kao niz rutina niskog nivoa. Za svaku rutinu, optimizator može da ima više implementiranih procedura. Na primer, za operaciju restrikcije može biti više procedura u zavisnosti od toga da li se traži po tačnoj vrednosti, da li postoji indeks nad atributom nad kojim se vrši restrikcija i drugo. Svaka procedura ima formulu za izračunavanje cene, obično u terminima U/I operacija.

Rutine niskog nivoa mogu biti čitanje indeksa, dohvaćanje torke, čitanje tabele, spajanje tabela i druge.

12.2.4 Formiranje planova upita i izbor najjeftinijeg

U ovoj fazi se formiraju planovi upita. Kako planova može biti suviše mnogo, uglavnom se koriste heuristike kako bi se izdvojio razuman obim planova. Za svaki plan se izračunava cena kao zbir cena pojedinačnih rutina od kojih se sastoji. Cena se ne može precizno izračunati jer zavisi i od veličine međurezultata, što zavisi od konkretnih vrednosti u bazi. Tako da se proračunava procena vrednosti plana upita.

12.3 Transformacija izraza

Neke transformacije mogu biti korisne u fazi 2 obrade upita.

Restrikcije i projekcije

Niz restrikcija je bolje zameniti jednom restrikcijom.

`(A where p1) where p2`

je bolje zameniti sa:

`A where p1 and p2`

Niz projekcija nad istom relacijom se može zameniti poslednjom projekcijom.

`(A [col1]) [col2]`

zameniti sa:

`A [col2]`

Restrikciju projekcije je bolje zameniti projekcijom restrikcije.

`(A [col]) where p1`

zameniti sa:

`(A where p) [col]`

Distributivnost

Unarni operator f je distributivan preko binarnog operatora o ako i samo ako važi:

$$f(AoB) \equiv f(A)of(B)$$

za sve A i B .

U relacionoj algebri, restrikcija je distributivna preko unije, preseka i razlike.

`(A unija B) where p = (A where p) union (B where p)`

`(A intersect B) where p = (A where p) intersect (B where p)`

`(A difference B) where p = (A where p) difference (B where p)`

Restrikcija je distributivna preko spajanja ako i samo ako postoje najviše dve restrikcije povezane AND operatorom, koje se odnose svaka na po jedan atribut spajanja.

Projekcija je distributivna preko unije i preseka, ali ne i preko razlike.

Projekcija je distributivna preko spajanja, sve dok projekcija zadržava sve attribute spajanja.

`(A unija B) [col] = (A [col]) union (B [col])`

`(A intersect B) [col] = (A [col]) intersect (B [col])`

Komutativnost i asocijativnost

Binarni operator \circ je komutativan ako i samo ako važi

$$A \circ B \equiv B \circ A$$

za sve A i B .

Binarni operator \circ je asocijativan ako i samo ako važi:

$$(A \circ B) \circ C \equiv A \circ (B \circ C)$$

za sve A, B i C .

U relacionoj algebri, unija, presek i spajanje su komutativni i asocijativni, ali razlika i deljenje nisu ni jedno ni drugo.

Idempotencija i absorpcija

Binarni operator \circ je idempotentan ako važi:

$$(A \circ A) \equiv A$$

U relacionoj algebri, unija, presek i spajanje su idempotentni, ali razlika i deljenje nisu.

Unija i presek zadovoljavaju i pravilo absorpcije:

$$(A \text{ union } (A \text{ intersect } B)) = A$$

$$(A \text{ intersect } (A \text{ union } B)) = A$$

Aritmetički izrazi

Opštija forma distributivnosti važi za neke aritmetičke operacije. Binarni operator δ je distributivan preko binarnog operatora \circ ako i samo ako važi:

$$A \delta (B \circ C) \equiv (A \delta B) \circ (A \delta C)$$

za sve A, B i C .

Na primer, $'*'$ je distributivan preko $'+'$, pa važi:

$$A * (B + C) \equiv (A * B) + (A * C)$$

Logički izrazi

Neka su A i B dva atributa dve različite relacije, tada je umesto izraza:

$$A > B \text{ and } B > 3$$

bolje pisati

$$A > B \text{ and } B > 3 \text{ and } A > 3$$

To je moguće zato što je relacija poređenja tranzitivna. Ovo omogućava da se dodatna restrikcija izvrši i pre spajanja.

Umesto

$A > B \text{ or } (C = D \text{ and } E < F)$

može se koristiti

$(A > B \text{ or } C = D) \text{ and } (A > B \text{ or } E < F)$

zahvaljujući činjenici da je OR distributivna preko AND.

Zapravo, svaki logički izraz se može predstaviti u konjunktivnoj normalnog formi (KNF):

$C1 \text{ and } C2 \text{ and } \dots \text{ and } CN$

gde su $C1, C2, \dots, CN$ logički izrazi koji ne sadrže konjunkcije.

KNF je tačna samo ako su svi konjunktivi tačni, pa optimizator može izabrati izračunavanje od najjednostavnijeg konjunktiva ka složenijim. U paralelnoj obradi se oni čak mogu i paralelno izračunavati. Čim se za neki utvrdi da je netačan, izračunavanje ostalih može da se prekine.

Semantičke transformacije

Transformisati upit u jednostavniji na osnovu toga šta zapravo predstavljaju. Na primer, u upitu:

$(\text{dosije join ispit}) [\text{indeks}, \text{ispredmeta}, \text{ocena}]$

nije bilo ni potrebe da se ispit spaja sa dosijeom, već je bilo dovoljno koristiti upit:

$\text{ispit} [\text{indeks}, \text{ispredmeta}, \text{ocena}]$

12.4 Statistika u bazi podataka

Faze 3 i 4 u obradi upita, koje se odnose na izbor načina pristupa podacima, se oslanjaju na statistiku baze podataka koja se nalazi u katalogu.

Osnovne statistike koje se koriste u DB2 su:

- Za svaku tabelu
 - Kardinalnost

- Broj stranica koje su zauzete tabelom
- Koliko deo prostora tabela je obuhvaćen tabelom
- Za svaku kolonu
 - Broj različitih vrednosti
 - Druga najveća vrednost u koloni
 - Druga najmanja vrednost u koloni
 - Za indeksirane kolone, deset najfrekventnijih vrednosti i broj njihovih pojavljivanja
- Za svaki indeks
 - Indikator da li je indeks klasterovan
 - Broj stranica u kojima su listovi
 - Broj nivoa u indeksu

Statistika se ne ažurira u realnom vremenu, već je potrebno pozvati RUNSTATS komandu.

12.5 Implementacija operatora spajanja

12.5.1 Gruba sila

Spajanje se može izvršiti takozvanom metodom 'grube sile'. To znači da se za svaku torku iz spoljne relacije sekvencijalno čita unutrašnja relacija u pronalaze redovi po uslovu spajanja. Ovaj metod se često zove i **ugnježdeno spajanje** (eng. nested loop), što nije precizan izraz, pošto i sve ostale metode imaju ugnježdene spajanja. U ovoj metodi je poželjno da spoljna relacija spajanja bude ona koja je manja.

12.5.2 Korišćenje indeksa

Ukoliko postoji indeks nad atributom spajanja unutrašnje relacije, tada se za svaku torku spoljne relacije pregleda indeks i direktno pristupa slogu unutrašnje relacije na osnovu vrednosti iz indeksa.

12.5.3 Korišćenje hash-a

Ova metoda je slična metodi korišćenja indeksa, sa tom razlikom što se umesto u indeks gleda u hash da bi se našao odgovarajući slog u unutrašnjoj tabeli.

12.5.4 Merge metoda

Ova metoda pretpostavlja da se obe tabele fizički nalaze u jednom nizu tako da se torke koje imaju istu vrednost atributa spajanja nalaze jedna za drugom. Tada u jednom prolazu se može izvršiti spajanje ove dve tabele.

12.5.5 Hash metoda

Hash metoda takođe koristi samo jedan prolaz kroz relacije. Prvi prolaz kreira hash tabelu nad atributom spajanja unutrašnje relacije. Drugi prolaz čita spoljašnju relaciju i za svaki slog izračunava hash vrednost koju zatim traži u hash tabeli unutrašnje relacije. Velika prednost ove metode u odnosu na merge metodu je u tome što torke ovih dveju relacija ne moraju da budu fizički učešljane, niti je potrebno sortiranje.

13. Relacioni upitni jezik SQL

13.1 Predavanje 1

Relacioni upitni jezik se koristi za komunikaciju sa relacionom bazom podataka. Korišćenjem relacionog upitnog jezika zadaju se naredbe za izdvajanje željenih podataka ili njihovo menjanje.

Prvi potpuno prototipski relacionalni sistem SYSTEM R je imao upitni jezik SQUARE (Specifying QUeries As Relational Expressions). Njegov sledbenik je SEQUEL (Structured English QUery Language), koji je u odnosu na SQUARE imao manje matematičku sintaksu. Ime upitnog jezika SEQUEL je kasnije preimenovano u SQL (Structured Query Language). Prvi standard za upitne jezike je ANSI SQL/86, koji je nastao na osnovu realizovanih verzija SQL jezika u tom trenutku. Od tada je objavljeno više standarda [6]. SQL sadrži podjezike kojima pripadaju:

- jezik za definisanje (eng. data definition language (DDL)) objekata u bazi podataka, npr. tabela, pogleda, indeksa;
- jezik za manipulaciju podacima (eng. data manipulation language (DML)), tj. za unos novih, brisanje ili menjanje postojećih podataka ili izdvajanje podataka;
- jezik za kontrolu pristupa podacima (eng. data control language (DCL)).

Jezik za definisanje podataka (DDL) se koristi za pravljenje (naredba create), brisanje (naredba drop) i menjanje (naredba alter) objekata u bazi podataka. Objekat koji se pravi, menja ili briše može biti baza podataka, tabela, pogled, indeks ... U ovom delu je dat kratak uvod u naredbu za pravljenje tabele da bi se razumele osnovne komponente tabele koje su potrebne za razumevanje naredbe za pretraživanje (select). U nastavku kursa će biti detaljnije obrađene naredbe jezika za definisanje podataka. U ovom poglavlju je dat opis SQL naredbi RSUBP db2.

Naredba za pravljenje baze podataka ima sintaksu

```
create database ime-baze-podataka
```

■ Primer 13.1 Napisati naredbu za pravljenje baze podataka stud2020

```
create database stud2020
```

■

Naredba za brisanje baze podataka ima sintaksu

```
drop database ime-baze-podataka
```

■ Primer 13.2 Napisati naredbu za brisanje baze podataka stud2020

```
drop database stud2020
```



Sintaksa naredbe za pravljenje tabele je

```
create table ime-tabele  
(def-kolone [, def-kolone]*  
[, def-primarni-ključ]?  
[, def-strani-ključ]*)
```

Definicija kolone (def-kolone) je oblika

```
ime-kolone tip-podatka [not null]?
```

gde tip podatka može biti ugrađeni (npr. int, varchar) ili korisnički definisani (npr. indeks). Neki od ugrađenih tipova podataka (eng. built-in-type) su:

- za cele brojeve:
 - smallint;
 - integer ili int;
- za realne brojeve:
 - float;
 - dec(ukupan_broj_cifara, broj_cifara_za_razlomljeni_deo);
- za niske:
 - char(dužina);
 - varchar(dužina);
- za datum - date.

Za svaku kolonu se može definisati da li ona može da sadrži ili ne nedostajuće vrednosti. Podrazumevano je da kolona može sadržati nedostajuće vrednosti. Ukoliko kolona ne sme da sadrži nedostajuće vrednosti, potrebno je u njenoj definiciji navesti `not null`.

Definicija primarnog ključa je oblika

```
[constraint ime]? primary key (ime-kolone [, ime-kolone]*)
```

Primarni ključ spada u ograničenja koja mogu da se definišu nad tabelom, te se na početku definicije primarnog ključa može navesti i ime koje se dodeljuje tom ograničenju (constraint ime). Sa ime-kolone [, ime-kolone]* se navode kolone čije vrednosti jedinstveno identifikuju svaku vrstu (red, torku). Bitna napomena: Nijedna kolona koja ulazi u sastav primarnog ključa ne sme da sadrži nedostajuće vrednosti, tj. mora da budu definisana sa `not null` opcijom.

Definicija stranog ključa je oblika

```
[constraint ime]? foreign key (ime-kolone [, ime-kolone]*)  
references bazna-tabela
```

Pošto i strani ključ spada u ograničenja koja mogu da se definišu nad tabelom, i njegova definicija može početi sa navođenjem imena ograničenja. Sa ime-kolone [, ime-kolone]* se navode kolone za koje mora važiti da njihove vrednosti moraju postojati u kolonama primarnog ključa bazne tabele na koju strani ključ referiše. Bazna tabela, na koju strani ključ referiše, se navodi sa references bazna-tabela. Npr. u tabeli Ispit se definiše strani ključ za koji je bazna tabela dosije. U koloni indeks u tabeli ispit može postojati samo indeks studenta koji postoji u primarnom ključu (tj. u koloni indeks) tabele dosije. Drugim rečima, postavljeno je ograničenje da ispit može da prijavi samo student o kome postoje podaci u tabeli dosije.

13.1.1 Jezik za manipulaciju podacima

Jeziku za manipulaciju podacima (DML) pripadaju naredbe za pretraživanje podataka (naredba select), unošenje novih redova u tabelu (naredba insert), menjanje postojećih redova u tabeli (naredba update), brisanje postojećih redova u tabeli (naredba delete) i pripajanje redova (naredba merge).

Naredba za pretraživanje

Naredba za pretraživanje (select) se sastoji od klauzula (linija). Osnovni oblik ove naredbe je

```
select lista-kolona 1  
from ime-tabele    2  
where uslovi        3
```

pri čemu se

- klauzula from koristi za navođenje tabele (ili tabela) iz kojih je potrebno izdvojiti podatke;
- klauzula select koristi za projekciju, tj. izdvajanje kolona koje su od interesa i koje će se naći u tabeli koja se dobija kao rezultat izvršavanja upita;
- klauzula where koristi za restrikciju, tj. navođenje logičkog uslova koji moraju da zadovolje redovi koji će se naći u tabeli koja se dobija kao rezultat izvršavanja upita.

Pored svake klauzule je naveden i redosled izvršavanja klauzula.

Uslov koji se navodi u okviru klauzule where može da sadrži poređenje vrednosti kolona, konstanti i izraza. U uslovu se niska kao konstanta navodi pod jednostrukim navodnicima. Za poređenje mogu da se koriste relacijske operacije =, <, >, <>, <= i >=. Ukoliko je

potrebno navesti više uslova, za spajanje se koriste logičke operacije I (and) i ILI (or). Dva uslova se spajaju sa operatorom and ako je potrebno da važe oba uslova, a sa operatorom or ako je dovoljno da važi jedan od uslova. Za negiranje uslova koristi se operator not u obliku not (uslov). U rezultatu će se naći samo redovi tabele koja je navedena u okviru klauzule from i za koje je tačan uslov naveden u okviru klauzule where.

■ **Primer 13.3** Prikazati kompletan sadržaj tabele dosije.

```
select *  
from dosije
```

■

■ **Primer 13.4** Prikazati broj indeksa, ime i prezime studenta.

```
select indeks, ime, prezime  
from dosije
```

■

■ **Primer 13.5** Prikazati informacije o studentima koji su rođeni u Beogradu.

```
select *  
from dosije  
where mesto_rodjenja = 'Beograd'
```

■

■ **Primer 13.6** Prikazati informacije o studentima koji su nisu rođeni u Beogradu.

```
select *  
from dosije  
where mesto_rodjenja <> 'Beograd'
```

■

■ **Primer 13.7** Prikazati imena i prezimena studenata koji su rođeni 01.01.1994. godine u Beogradu.

```
select ime, prezime  
from dosije  
where datum_rodjenja='01.01.1994'and mesto_rodjenja='Beograd'
```

■

■ **Primer 13.8** Prikazati imena i prezimena studenata koji su rođeni 01.01.1994. godine ili su rođeni u Beogradu.

```
select ime, prezime
from   dosije
where  datum_rodjenja='01.01.1994' or mesto_rodjenja='Beograd'
```

Provera da li je vrednosti kolone (ili izraza) nedostajuća vrši se pomoću operatora `is null`. Upotrebljava se u obliku

```
izraz is null
```

Uslov za proveru da li vrednost izraza nije nedostajuća se zadaje u obliku

```
izraz is not null
```

■ **Primer 13.9** Prikazati imena i prezimena studenata za koje nije poznato mesto rođenja.

```
select ime, prezime, mesto_rodjenja
from dosije
where mesto_rodjenja is null
```

■ **Primer 13.10** Prikazati imena i prezimena studenata za koje je poznato mesto rođenja.

```
select ime, prezime, mesto_rodjenja
from dosije
where mesto_rodjenja is not null
```

13.2 Predavanje 2

13.2.1 Uklanjanje dupliranih redova iz rezultata

Tabela koja se dobija kao rezultat upita može da sadrži duplirane redove. Da bi se uklonili duplirani redovi iz rezultata, potrebno je navesti `distinct` pre navođenja liste kolona u klauzuli `select`.

■ **Primer 13.11** Prikazati jedinstvene identifikatore predmeta koji su polagani u januarskom ispitnom roku 2015. godine.

```
select distinct id_predmeta
from   ispit
where  godina_roka=2015 and oznaka_roka='jan'
```

■ **Primer 13.12** Prikazati identifikatore predmeta i ocene dobijene na ispitima iz tih predmeta u januarskom ispitnom roku 2015. godine. Upit napisati tako da u rezultatu nema ponavljanja redova.

```
select distinct id_predmeta, ocena
from   ispit
where  godina_roka=2015 and oznaka_roka='jan'
```

Primetiti da sledeće rešenje nije korektno:

```
select distinct id_predmeta, distinct ocena
from   ispit
where  godina_roka=2015 and oznaka_roka='jan'
```

13.2.2 Dodatni operatori za postavljanje uslova

Pored uobičajenih relacija za poređenje, mogu se koristiti i predikati:

- **between** za proveru da li je vrednost izraza u zadatom opsegu i koristi se u obliku

```
izraz between pocetak and kraj
```

Uslov je tačan ako je rezultat izraza u intervalu [pocetak, kraj]. Ukoliko je potrebno navesti uslov da vrednost izraza nije u zadatom opsegu, operator **between** može se koristiti sa operatorom **not** u obliku

```
izraz not between pocetak and kraj
```

- **in** za proveru da li je vrednost izraza u skupu željenih vrednosti i koristi se u obliku

```
izraz in (lista vrednosti razdvojenih zapetama)
```

Ukoliko je potrebno navesti uslov da vrednost izraza nije u navedenom skupu vrednosti, operator **in** može se koristiti sa operatorom **not** u obliku

```
izraz not in (lista vrednosti razdvojenih zapetama)
```

- like za poređenje niski, pri čemu se pravi razlika između malih i velikih slova. Upotrebljava se u obliku

```
izraz like maska
```

pri čemu se maska navodi između jednostrukih navodnika. Karakteri % i _ u okviru maske imaju posebno značenje:

- _ označava pojavljivanje bilo kog jednog karaktera;
- % označava pojavljivanje 0 ili više bilo kog karaktera.

Ukoliko se neki od ovih znakova sa specijalnim značenjem mora upotrebiti u okviru maske onda se ispred njega navodi prekidački simbol koji se definiše sa escape na sledeći način

```
izraz like maska escape prekidacki-simbol
```

- **Primer 13.13** Za studente čije su ocene na ispitima u intervalu [6,8] prikazati broj indeksa, identifikator predmeta i dobijenu ocenu.

```
select indeks, id_predmeta, ocena
from   ispit
where  ocena between 6 and 8
```

Zadatak se može rešiti i bez korišćenja operatora between:

```
select indeks, id_predmeta, ocena
from   ispit
where  ocena >= 6 and ocena <= 8
```

- **Primer 13.14** Za ispite koji nisu polagali u ispitnom roku sa oznakom *jan* i na kojima je dobijena ocena 7, 8 ili 9, prikazati broj indeksa, identifikator predmeta i dobijenu ocenu.

```
select indeks, id_predmeta, ocena
from   ispit
where  ocena in (7,8,9) and oznaka_roka <> 'jan'
```

Rešenje zadatka je i upit

```
select indeks,id_predmeta, ocena
from   ispit
where  (ocena=6 or ocena=7 or ocena=8) and oznaka_roka<>'jan'
```

■ **Primer 13.15** Prikazati ime, prezime i mesto rođenja za svakog studenata čiji naziv mesta rođenja

- se završava na *ad*

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja like '%ad'
```

- sadrži slovo *r*

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja like '%r%'
```

- kao drugo slovo sadrži *r*

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja like '_r%'
```

- ima podnisku *m%k*

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja like '%m+%k%' escape '+'
```

ili

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja like '%m\%k%' escape '\'
```

- ne sadrži slovo *o*

```
select ime, prezime, mesto_rodjenja
from   dosije
where  mesto_rodjenja not like '%o%'
```

13.2.3 Izvedene kolone

U klauzuli `select` se mogu navoditi imena kolona, a mogu se navoditi i računski izrazi. Računski izraz se posebno izvršava za svaki red u rezultatu i dobijeni rezultati računskog izraza će biti u posebnoj koloni u tabeli koja se dobija kao rezultat upita. Kolona koja sadrži

rezultat računskog izraza podrazumevano nema ime, tj. dodeljen joj je redni broj kolone u rezultatu. Koloni se može dodeliti ime tako što se iza izraza navede i ime u jednom od sledećih oblika:

- `izraz ime`
Kolona će u rezultatu imati naziv **IME**.
- `izraz AS ime`
Kolona će u rezultatu imati naziv **IME**.
- `izraz as "Ime"`
Kolona će u rezultatu imati naziv **Ime**.
- `izraz "Ime"`
Kolona će u rezultatu imati naziv **Ime**.

Na isti način se koloni koja već ima ime može dodeliti novo ime koje će biti prikazano u tabeli koja se dobija kao rezultat upita.

■ **Primer 13.16** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova.

```
select id_predmeta, sifra, bodovi* 2
from predmet
```

U rezultatu će kolona sa dvostrukim brojem bodova imati ispisan redni broj umesto imena.

■ **Primer 13.17** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova. Novodobijenu vrednost označiti kao *DVOSTRUKO*.

```
select id_predmeta, sifra, bodovi* 2 as dvostruko
from predmet
```

■ **Primer 13.18** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova. Novodobijenu vrednost označiti kao *Dvostruko*.

```
select id_predmeta, sifra, bodovi* 2 as "Dvostruko"
from predmet
```

■ **Primer 13.19** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova. Novodobijenu vrednost označiti kao *Dvostruko*. Između kolona šifra i *Dvostruko* dodati kolonu sa imenom *Opis* u okviru koje će za svaki red biti ispisan *Dvostruka vrednost broja bodova je =*.

```
select id_predmeta, sifra,  
       'Dvostruka vrednost broja bodova je =' as "Opis",  
       bodovi* 2 as "Dvostruko"  
from predmet
```

■ **Primer 13.20** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova. Novodobijenu vrednost označiti kao *Dvostruko*. Između kolona šifra i Dvostruko dodati kolonu sa imenom *Opis* u okviru koje će za svaki red biti ispisano *Dvostruka vrednost broja bodova je =*. Izdvojiti samo predmete za koje je dvostruki broj bodova veći od 15.

```
select id_predmeta, sifra,  
       'Dvostruka vrednost broja bodova je =' as "Opis",  
       bodovi* 2 as "Dvostruko"  
from predmet  
where bodovi*2>15
```

Primetiti da naredno rešenje nije korektno zbog redosleda izvršavanja klauzula. Klauzula where se izvršava pre klauzule select, te se novo ime kolone, koje se dodeljuje u klauzuli select ne može koristiti u klauzuli where.

```
select id_predmeta, sifra,  
       'Dvostruka vrednost broja bodova je =' as "Opis",  
       bodovi* 2 as "Dvostruko"  
from predmet  
where "Dvostruko" >15
```

13.2.4 Uređivanje rezultata upita

Rezultat upita može biti uređen po jednoj ili više kolona u rastućem ili opadajućem redosledu pomoću klauzule `order by` koja se navodi na kraju upita u obliku

```
order by lista-kolona
```

U klauzuli `order by` može se navesti ime ili redni broj kolone iz klauzule `select`. Rezultat se po jednoj koloni može urediti

- u rastućem poretку i tada se iza imena ili broja kolone navodi `asc`;

- u opadajućem poretku i tada se iza imena ili broja kolone navodi desc.

Ako se ne navede poredak, podrazumeva se rastući.

Ukoliko je u klauzuli `order by` navedena lista sa dve ili više kolona, tada se redovi u rezultatu uređuju prvo po vrednostima u prvoj koloni u listi prema zadatom poretku, zatim se redovi koji imaju iste vrednosti u prvoj koloni uređuju po vrednostima u drugoj koloni u listi prema zadatom poretku ...

Kada se u upitu navede klauzula `order by` redosled izvršavanja klauzula naredbe `select` je

```
select lista-kolona      3
from ime-tabele          1
where uslovi             2
order by lista-kolona    4
```

■ **Primer 13.21** Prikazati sadržaj tabele predmet uređen po broju bodova u rastućem i šifri predmeta u opadajućem redosledu.

```
select *
from predmet
order by bodovi asc, sifra desc
```

ili

```
select *
from predmet
order by bodovi, sifra desc
```

ili

```
select id_predmeta, sifra, bodovi, naziv
from predmet
order by bodovi, sifra desc
```

ili

```
select id_predmeta, sifra, bodovi, naziv
from predmet
order by 3, 2 desc
```

ili


```
select id_predmeta, sifra, bodovi, naziv
from predmet
order by 3, sifra desc
```

■ **Primer 13.22** Prikazati identifikator i šifru predmeta, kao i dvostruku vrednost broja njegovih bodova. Novodobijenu kolonu nazvati DVOSTRUKO. Rezultat urediti po dvostrukom broju bodova u rastućem i šifri predmeta u opadajućem redosledu.

```
select id_predmeta, sifra, bodovi* 2 as dvostruko
from predmet
order by 3 asc, 2 desc
```

ili

```
select id_predmeta, sifra, bodovi* 2 as dvostruko
from predmet
order by dvostruko asc, 2 desc
```

Primititi da dodeljeno ime koloni u klauzuli select može da se koristi u klauzuli order by jer se klauzula order by izvršava nakon klauzule select.

13.2.5 Proizvod tabela

Navođenjem dve (ili više) tabela u klauzuli from u obliku

```
from tabela1, tabela2
```

vrši se proizvod navedenih tabela. Ako se u listi tabela navedu dve tabele, tada se kao rezultat dobija tabela koja u zaglavlju ima sve kolone iz prve tabele i sve kolone iz druge tabele. Telo tabele koja se dobija kao rezultat sadrži sve redove iz prve tabele uparene sa svakim redom druge tabele. U klauzuli select mogu se navoditi samo imena kolona bez navođenja imena tabele ako je ime kolone jedinstveno, u sprotnom se mora navesti i ime tabele, tj. kvalifikovano ime u obliku tabela.kolona.

Proizvod tabela se može izvršiti i korišćenjem operatora cross join u obliku

```
from tabela1 cross join tabela2
```

■ **Primer 13.23** Prikazati sve moguće kombinacije studenata i predmeta koje oni mogu da odaberu.

```
select *  
from dosije, predmet
```

ili

```
select *  
from dosije cross join predmet
```

■

13.2.6 Spajanje tabela

Unutrašnje spajanje tabela

Dve tabela se mogu spojiti primenom proizvoda nad tim tabelama, a zatim restrikcije radi navođenja uslova spajanja.

■ **Primer 13.24** Prikazati podatke o svim ispitima i studentima koji su ih polagali. Izveštaj urediti po broju indeksa studenta i oznaci ispitnog roka u kome je ispit polagan.

```
select *  
from dosije, ispit  
where dosije.indeks=ispit.indeks  
order by dosije.indeks, ispit.oznaka_roka
```

Da u rezultatu ne bi bile obe kolone po kojima se vrši spajanje, potrebno je eksplicitno navesti spisak kvalifikovanih kolona u klauzuli select.

```
select dosije.indeks,dosije.ime, dosije.prezime, dosije.datum_rodjenja,  
       dosije.mesto_rodjenja, dosije.datum_upisa,  
       ispit.id_predmeta, ispit.godina_roka, ispit.oznaka_roka,  
       ispit.ocena, ispit.datum_ispita, ispit.bodovi  
from dosije, ispit  
where dosije.indeks=ispit.indeks  
order by dosije.indeks, ispit.oznaka_roka
```

ili

```
select dosije.indeks,ime, prezime, datum_rodjenja, mesto_rodjenja,  
       datum_upisa, id_predmeta, godina_roka, oznaka_roka, ocena,  
       datum_ispita, bodovi  
from dosije, ispit  
where dosije.indeks=ispit.indeks  
order by dosije.indeks, oznaka_roka
```

■

Dve tabele se mogu spojiti i korišćenjem operatora `inner join` za unutrašnje spajanje u okviru klauzule `from` u obliku

```
from tabela1 [inner]? join tabela2 on uslov-spajanja
```

Neka su date dve tabele kao na 13.1.

a	c
1	m
2	n
3	l

(a) Tabela A

b	c
4	m
5	n
6	r

(b) Tabela B

Tabela 13.1: Primer dve tabele

Rezultat unutrašnjeg spajanja ovih tabela upitom

```
select tabelaA.a, tabelaA.c, tabelaB.b
from  tabelaA join tabelaB on tabelaA.c=tabelaB.c
```

ili

```
select tabelaA.a, tabelaA.c, tabelaB.b
from  tabelaA inner join tabelaB on tabelaA.c=tabelaB.c
```

je prikazan u tabeli 13.2. Kako bi izgledao rezultat da je u klauzuli `select` navedena `*`?

a	c	b
1	m	4
2	n	5

Tabela 13.2: Rezultat primera unutrašnjeg spajanja tabela A i B

■ **Primer 13.25** Prikazati podatke o svim ispitima i studentima koji su ih polagali. Izveštaj urediti po broju indeksa studenta i oznaci ispitnog roka u kome je ispit polagan.

```
select *
from  dosije join ispit on dosije.indeks=ispit.indeks
order by dosije.indeks, ispit.oznaka_roka
```

ili

```
select *
from   dosije inner join ispit on dosije.indeks=ispit.indeks
order by dosije.indeks, ispit.oznaka_roka
```

■

Spoljašnje spajanje tabela

Rezultat unutrašnjeg spajanja su upareni redovi koji postoje u obe tabele i zadovoljavaju uslov spajanja. Ako je potrebno da se u rezultatu nađu i redovi iz jedne ili obe tabele, a koji nemaju svog para po navedenom uslovu spajanja, koristi se spoljašnje spajanje. Postoje tri vrste spoljašnjeg spajanja: levo, desno i potpuno.

Operator za levo spoljašnje spajanje `left outer join` se koristi u obliku

```
from tabela1 left [outer]? join tabela2 on uslov-spajanja
```

Tabela dobijena primenom levog spoljašnjeg spajanja se sastoji od uparenih redova obe tabele po zadatom uslovu spajanja i od redova koji se nalaze u levoj (tabela1) tabeli, a nemaju svog para u desnoj tabeli (tabela2) po zadatom uslovu spajanja. U rezultatu će za redove iz leve tabele koji nemaju svog para biti ispisane nedefinisane vrednosti u kolonama preuzetim iz desne tabele. Zaglavlje tabele koja se dobija kao rezultat sadrži sve kolone iz leve i sve kolone iz desne tabele.

Rezultat levog spoljašnjeg spajanja tabela navedenih u tabeli 13.1 izvršavanjem upita

```
select tabelaA.a, tabelaA.c, tabelaB.c, tabelaB.b
from   tabelaA left join tabelaB on tabelaA.c=tabelaB.c
```

je prikazan u tabeli 13.3.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
3	l	null	null

Tabela 13.3: Rezultat primera levo spoljašnjeg spajanja tabela A i B

Operator za desno spoljašnje spajanje `right outer join` se koristi u obliku

```
from tabela1 right [outer]? join tabela2 on uslov-spajanja
```

Tabela dobijena primenom desno spoljašnjeg spajanja se sastoji od uparenih redova obe tabele po zadatom uslovu spajanja i od redova koji se nalaze u desnoj (tabela2) tabeli, a nemaju svog para u levoj tabeli (tabela1) po zadatom uslovu spajanja. U rezultatu će za redove iz desne tabele koji nemaju svog para biti ispisane nedefinisane vrednosti u kolonama preuzetim iz leve tabele. Zaglavlje tabele koja se dobija kao rezultat sadrži sve kolone iz leve i sve kolone iz desne tabele.

Rezultat desno spoljašnjeg spajanja tabela navedenih u tabeli 13.1 izvršavanjem upita

```
select tabelaA.a, tabelaA.c, tabelaB.c, tabelaB.b
from  tabelaA right join tabelaB on tabelaA.c=tabelaB.c
```

je prikazan u tabeli 13.4.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
null	null	r	6

Tabela 13.4: Rezultat primera desno spoljašnjeg spajanja tabela A i B

Operator za potpuno spoljašnje spajanje `full outer join` se koristi u obliku

```
from tabela1 full [outer]? join tabela2 on uslov-spajanja
```

Tabela dobijena primenom potpuno spoljašnjeg spajanja se sastoji od:

- uparenih redova navedenih tabela po zadatom uslovu spajanja;
- redova koji se nalaze u levoj (tabela1) tabeli, a nemaju svog para u desnoj tabeli (tabela2) po zadatom uslovu spajanja. U rezultatu će za redove iz leve tabele koji nemaju svog para biti ispisane nedefinisane vrednosti u kolonama preuzetim iz desne tabele.
- redova koji se nalaze u desnoj (tabela2) tabeli, a nemaju svog para u levoj tabeli (tabela1) po zadatom uslovu spajanja. U rezultatu će za redove iz desne tabele koji nemaju svog para biti ispisane nedefinisane vrednosti u kolonama preuzetim iz leve tabele.

Zaglavlje tabele koja se dobija kao rezultat sadrži sve kolone iz leve i sve kolone iz desne tabele.

Rezultat potpuno spoljašnjeg spajanja tabela navedenih u tabeli 13.1 izvršavanjem upita

```
select tabelaA.a, tabelaA.c, tabelaB.c, tabelaB.b
from  tabelaA full join tabelaB on tabelaA.c=tabelaB.c
```

je prikazan u tabeli 13.5.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
3	l	null	null
null	null	r	6

Tabela 13.5: Rezultat primera potpuno spoljašnjeg spajanja tabela A i B

■ **Primer 13.26** Prikazati informacije o svim predmetima i ispitima na kojima su polagani. Izdvojiti i predmete iz kojih nisu polagani ispiti.

```
select *
from  predmet left join ispit on predmet.id_predmeta=ispit.id_predmeta
```

ili

```
select *
from  ispit right join predmet on predmet.id_predmeta=ispit.id_predmeta
```

■

13.3 Predavanje 3

13.3.1 Aliasi

Tabeli u klauzuli from može se dodeliti alias na sledeći način

```
from tabela [as] * alias
```

Korišćenje aliasa je

- neophodno ako se ista tabela koristi u upitu u više različitih konteksta;
- zgodno ako tabela ima dugačko ime.

■ **Primer 13.27** Prikazati imena parova studenata koji su rođeni u istom mestu.

```
select a.ime, b.ime, a.mesto_rodjenja
from  dosije a, dosije b
where a.mesto_rodjenja=b.mesto_rodjenja
      and a.indeks>b.indeks
```

ili

```
select a.ime, dosije.ime, a.mesto_rodjenja
from   dosije a, dosije
where  a.mesto_rodjenja=dosije.mesto_rodjenja
       and a.indeks>dosije.indeks
```

■

■ **Primer 13.28** Prikazati broj indeksa, broj bodova i nazive svih predmeta koje je polagao student čije mesto rođenja nije Beograd.

```
select a.indeks, b.bodovi, c.bodovi, naziv
from   dosije a, predmet b, ispit c
where  a.indeks=c.indeks
and    b.id_predmeta=c.id_predmeta
and    a.mesto_rodjenja<>'Beograd'
```

ili

```
select a.indeks, b.bodovi, c.bodovi, naziv
from   dosije a join ispit c on a.indeks=c.indeks
join   predmet b on b.id_predmeta=c.id_predmeta
where  a.mesto_rodjenja<>'Beograd'
```

■

13.3.2 Skupovni operatori

U SQL-u postoje skupovni operatori za uniju, presek i razliku koji se upotrebljavaju u obliku

```
upit1 skupovni-operator upit2
```

Potrebno je da tabele nad kojima se primenjuje skupovni operator budu istog tipa (imaju isti broj kolona koje su kompatibilnih tipova). Sintaksa skupovnog operatora

- za uniju je:

```
upit1 union [all]? upit2
```

- za presek je:

```
upit1 intersect [all]? upit2
```

- za razliku je:

```
upit1 except [all]? upit2
```

Operator bez upotrebe `all` se ponaša kao skupovni operator, tj. ne uzima u obzir ponovljene redove. Ako je potrebno da uzima u obzir ponovljene redove upotrebljava se operator sa `all`. Rezultat je tabela koja imena kolona preuzima iz prvog upita nad kojim se primenjuje skupovni operator. Klauzul `order by` se navodi na kraju upita u kome se koristi skupovni operator i odnosi se na konačan rezultat.

Razlika upotrebe operatora sa i bez `all` je ilustrovana primerima u tabelama 13.6 i 13.7.

X		X		X
1	union	1	=	1
2		2		2
2		3		3
3		4		4
		4		

Tabela 13.6: Primer primene operatora za uniju

X		X		X
1	union all	1	=	1
2		2		1
2		3		2
3		4		2
		4		2
				3
				3
				4
				4

Tabela 13.7: Primer primene operatora za uniju

■ **Primer 13.29** Prikazati broj bodova i nazive svih predmeta ako predmeti pripadaju grupi Analiza ili su polagani u januarском ispitnom roku 2015. godine.

```
select naziv, bodovi
from   predmet
```



```
where naziv like 'Analiza%'

union

select naziv, a.bodovi
from   predmet a, ispit b
where  a.id_predmeta=b.id_predmeta
and    godina_roka=2015
and    oznaka_roka='jan'

order by bodovi desc
```

■

■ **Primer 13.30** Prikazati broj bodova i nazive svih predmeta ako predmeti pripadaju grupi Analiza i polagani su u januarskom ispitnom roku 2015. godine.

```
select naziv, bodovi
from   predmet
where  naziv like 'Analiza%'

intersect

select naziv, a.bodovi
from   predmet a, ispit b
where  a.id_predmeta=b.id_predmeta
and    godina_roka=2015
and    oznaka_roka='jan'

order by bodovi desc
```

■

■ **Primer 13.31** Prikazati broj bodova i nazive svih predmeta ako predmeti pripadaju grupi Analiza, ali nisu polagani u januarskom ispitnom roku 2015. godine.

```
select naziv, bodovi
from   predmet
where  naziv like 'Analiza%'

except

select naziv, a.bodovi
```

```
from   predmet a, ispit b
where  a.id_predmeta=b.id_predmeta
and    godina_roka=2015
and    oznaka_roka='jan'
```

```
order by bodovi desc
```

■

■ **Primer 13.32** Prikazati brojeve indeksa, imena, prezimena i mesta rođenja svih studenata koji su rođeni posle 1.1.1991. godine. U slučaju da je mesto rođenja studenta nepoznato, kao mesto rođenja ispisati *Nepoznato mesto*.

```
select indeks, ime, prezime, mesto_rodjenja
from   dosije
where  datum_rodjenja> '01.01.1991'
and    mesto_rodjenja is not null
```

```
union
```

```
select indeks, ime, prezime, 'Nepoznato mesto'
from   dosije
where  datum_rodjenja> '01.01.1991'
and    mesto_rodjenja is null
```

13.3.3 Podupiti

U okviru upita mogu se pisati i podupiti. Na primer, u okviru klauzule where kao deo operatora in može se navesti podupit.

■ **Primer 13.33** Prikazati ime, prezime i broj indeksa za studente koji su polagali ispit u nekom od ispitnih rokova 2015. godine.

```
select ime, prezime, indeks
from   dosije
where  indeks in (
        select indeks
        from   ispit
        where  godina_roka=2015
      )
```

Za nekvalifikovanu kolonu u podupitu podrazumeva se da je kvalifikovana imenom tabele koja se pojavljuje u podupitu. U prethodnom primeru podrazumeva se da se kolona indeks u podupitu odnosi na tabelu ispit.

■ **Primer 13.34** Prikazati ime, prezime, broj indeksa studenta i naziv predmeta koje je student položio u nekom od ispitnih rokova 2015. godine.

```
select ime, prezime, indeks, naziv
from   dosije, predmet
where  (indeks, id_predmeta) in (
                                select indeks, id_predmeta
                                from   ispit
                                where  godina_roka=2015
                                and    ocena > 5
                                )
```

■

Poduput koji zavisi od kolone kojoj se vrednost dodeljuje u spoljašnjem delu upitu (tj. van podupita) se naziva korelirani upit. Korelirani upit se izvršava u ciklusima, po jedanput za svaku vrednost kolone iz spoljašnjeg dela.

■ **Primer 13.35** Za studente koji su položili barem jedan ispit, izdvojiti indekse i nazive predmeta koje su položili.

```
select indeks, (select naziv
                 from predmet p
                 where p.id_predmeta=i.id_predmeta) as predmet
from ispit i
where ocena>5
```

ili

```
select indeks, (select naziv
                 from predmet p
                 where id_predmeta=i.id_predmeta) as predmet
from ispit i
where ocena>5
```

■

u prethodnom primeru su korelirani jer se izvršavaju po jedanput za svaki ispit.

13.4 Predavanje 4

13.4.1 Kvantifikovana poređenja

Kvantifikovanjem poređenja moguće je porediti skalarnu vrednost sa tabelom skalarnih vrednosti[4]. Kvantifikovana poređenja su oblika

`izraz relacija kvantifikator (podupit)`

pri čemu kvantifikator može biti

- all
- any
- some

Pri upotrebi kvantifikatora all uslov je tačan ako rezultat podupita nije prazan i rezultat izraza je u datoj relaciji sa svim redovima u rezultatu podupita.

Pri upotrebi kvantifikatora any ili some uslov je tačan ako rezultat podupita nije prazan i rezultat izraza je u datoj relaciji sa barem jednim redom u rezultatu podupita.

■ **Primer 13.36** Pronaći ispit koji je položen sa najvećim brojem bodova.

```
select *
from ispit
where bodovi >= all (select bodovi
                    from ispit
                    where bodovi is not null)
```

■

■ **Primer 13.37** Pronaći ispit koji nije položen sa najvećim brojem bodova.

```
select *
from ispit
where bodovi < any (select bodovi
                   from ispit
                   where bodovi is not null)
```

■

13.4.2 Egzistencijalni kvantifikator

U SQL-u postoji podrška za egzistencijalni kvatifikator exists koji se koristi za postavljanje uslova u obliku

```
exists (podupit)
```

Uslov je tačan akko je rezultat podupita neprazan.

Univerzalini kvantifikator nije podržan u upitnom jeziku SQL, ali se uslov koji bi se izrazio pomoću univerzalnog kvantifikatora može izraziti pomoću negacije egzistencijalnog kvantifikatora:

```
forall x (uslov) = not exists x (not uslov)
```

■ **Primer 13.38** Prikazati broj indeksa, ime i prezime studenta koji ima barem jednu ocenu 10.

```
select indeks, ime, prezime
from   dosije a
where  exists (
        select *
        from   ispit b
        where  b.indeks=a.indeks
              and ocena=10
      )
```

■ **Primer 13.39** Prikazati broj indeksa, ime i prezime studenta koji ni na jednom ispitu nije dobio ocenu 10.

```
select indeks, ime, prezime
from   dosije a
where  not exists (
        select *
        from   ispit b
        where  b.indeks=a.indeks
              and ocena=10
      )
```

■ **Primer 13.40** Prikazati nazive predmeta koji su polagani u nekom od ispitnih rokova iz 2015. godine, i za koje važi da niko od studenata koji su ih polagali nije pao.

```
select naziv
from   predmet a
```

```

where exists ( select *
               from   ispit b
               where   b.id_predmeta=a.id_predmeta and
                       godina_roka=2015)
and not exists ( select *
                from   ispit b
                where   b.id_predmeta=a.id_predmeta
                       and godina_roka=2015 and ocena=5)

```

ili

```

select naziv
from   predmet p
where  not exists (
        select *
        from   dosije d
        where  not exists (
                select *
                from ispit i
                where p.id_predmeta = i.id_predmeta
                       and d.indeks=i.indeks
                       and i.ocena>5
                )
        )

```

ili

```

select naziv
from   predmet p
where  not exists (
        select 1
        from   dosije d
        where  not exists (
                select 1
                from ispit i
                where p.id_predmeta = i.id_predmeta
                       and d.indeks=i.indeks
                       and i.ocena>5
                )
        )

```

ili

```

select naziv
from   predmet p
where  not exists (
        select *
        from dosije d
        where indeks not in ( select indeks
                              from ispit i
                              where i.id_predmeta = p.id_predmeta
                              and i.ocena>5
                            )
      )

```

■

13.4.3 Tabela sa konstantnim redovima

Za prikaz tabele sa konstantnim redovima koristi se klauzula values u obliku

```
values red1, red2, ... redN
```

a svaki red se navodi u obliku

```
(kolona1, kolona2, ..., kolonaM)
```

Svi redovi moraju biti istog tipa, tj. vrednosti koje će biti u istoj koloni u rezultatu moraju biti istog tipa.

■ **Primer 13.41** Prikazati tabelu sa sledećim sadržajem

1
a
b
c

```
values ('a'), ('b'), ('c')
```

■

■ **Primer 13.42** Prikazati tabelu sa sledećim sadržajem

```
values ('a', 1), ('b', 2), ('c', 3)
```

■

1	2
a	1
b	2
c	3

13.4.4 Specijalni registri

U DB2 sistemu postoje specijalni registri koji čuvaju informacije koje se mogu koristiti u SQL naredbama. Neki od specijalnih registara su:

- `current schema` koji čuva naziv podrazumevane šeme;
- `current date` (ili `current_date`) koji se koristi za dobijanje datuma izvršavanja naredbe;
- `current time` (ili `current_time`) koji se koristi za dobijanje vremena izvršavanja naredbe;
- `current timestamp` (ili `current_timestamp`) koji se koristi za dobijanje kombinacije datuma i vremena izvršavanja naredbe (npr. u formatu `yyyy-mm-dd-hh.mm.ss.nnnnnn`, primer: `2022-10-29 12:10:42.933732`);
- `current timezone` koji se koristi za dobijanje vrednosti koja predstavlja razliku između UTC (eng. Coordinated Universal Time - koordinisano univerzalno vreme) i lokalnog vremena koje je definisano na serveru;
- `user` koji čuva ime korisnika koji izvršava naredbu.

■ **Primer 13.43** Izlistati trenutni datum.

```
select current date
from dosije
```

U rezultatu će datum izvršavanja naredbe biti prikazan u onoliko redova koliko ih ima u tabeli dosije. Da ne bi bilo ponavljanja redova, umesto tabele dosije može se koristiti tabela `sysibm.sysdummy1` koja ima samo jedan red i jednu kolonu.

```
select current date
from sysibm.sysdummy1
```

ili klauzula `values`

```
values (current date)
```

■

13.4.5 Skalarnе funkcije

Skalarnе funkcije su one funkcije koje kao argumente dobijaju skalarnе vrednosti (jedan argument je jedna vrednost), i kao rezultat vraćaju skalarnu vrednost.

U tabeli 13.8 je dat opis nekih specifičnih skalarnih funkcija, a u tabelama 13.9 i 13.10 je dat opis nekih skalarnih funkcija za rad sa niskama.

Funkcija	Opis
<code>coalesce(arg1, arg2, ...)</code>	vraća prvi argument iz liste argumenata čija vrednost nije null
<code>value(arg1, arg2, ...)</code>	isto kao <code>coalesce(arg1, arg2, ...)</code>

Tabela 13.8: Specifične skalarne funkcije

■ **Primer 13.44** Za svaki ispit izdvojiti dobijenu ocenu i datum polaganja ispita. Ako je dobijena ocena na ispitu nepoznata, umesto null vrednosti ispisati -10, a ako je datum polaganja nepoznat umesto null vrednosti ispisati današnji datum.

```
select value(ocena,-10), coalesce(datum_ispita, current date)
from ispit
```

■

■ **Primer 13.45** Za svaki predmet izdvojiti drugo i treće slovo iz naziva i na njih nadovezati šifru predmeta.

```
select substr(naziv,2,2) concat sifra
from predmet
```

ili

```
select concat(substr(naziv,2,2), sifra)
from predmet
```

ili

```
select substr(naziv,2,2) || sifra
from predmet
```

Primer rezultata: naM113

■

■ **Primer 13.46** Ispisati nisku koja reč *Tekst* sadrži 3 puta.

```
values (repeat('Tekst',3))
```

ili

```
select repeat('Tekst',3)
from sysibm.sysdummy1
```

Funkcija	Opis
substr(niska_arg, početak_arg, dužina_arg)	vraća podnisku iz <i>niska_arg</i> dužine <i>dužina_arg</i> koja počinje od pozicije <i>početak_arg</i>
concat(arg1, arg2) ili sa korišćenjem operatora arg1 arg2	spajanje niski
space(arg1)	vraća <i>arg1</i> praznih mesta
posstr(arg1, arg2)	traži gde počinje niska <i>arg2</i> u <i>arg1</i>
repeat(arg1, arg2)	ponavlja nisku <i>arg1</i> <i>arg2</i> . puta
replace(arg1, arg2, arg3)	zamenjuje sva pojavljivanja niske <i>arg2</i> u <i>arg1</i> sa niskom <i>arg3</i>
length(arg1)	vraća dužinu niske <i>arg1</i>
lcase(arg1) ili lower(arg1)	vraća nisku <i>arg1</i> u kojoj su velika slova zamenjena malim slovima
ucase(arg1) ili upper(arg1)	vraća nisku <i>arg1</i> u kojoj su mala slova zamenjena velikim slovima
left(arg1, arg2)	vraća prefiks niske <i>arg1</i> dužine <i>arg2</i>
right(arg1, arg2)	vraća sufiks niske <i>arg1</i> dužine <i>arg2</i>
rtrim(arg1)	briše beline sa kraja niske <i>arg1</i> i vraća nisku koju dobije kao rezultat
ltrim(arg1)	briše beline sa početka niske <i>arg1</i> i vraća nisku koju dobije kao rezultat
trim(arg1)	briše beline sa početka i kraja niske <i>arg1</i> i vraća nisku koju dobije kao rezultat

Tabela 13.9: Opis nekih skalarnih funkcija za rad sa niskama

Rezultat: TekstTekstTekst

■ **Primer 13.47** Ispisati mesta rođenja studenata i umesto *Beograd* ispisati *Bg*.

```
select replace(mesto_rodjenja, 'Beograd', 'Bg')
from dosije
```

■ **Primer 13.48** Ispisati nazive predmeta samo sa malim slovima, a zatim samo sa velikim slovima.

```
select lcase(naziv), lower(naziv), ucase(naziv), upper(naziv)
from predmet
```

Funkcija	Opis
<code>char(arg1)</code>	pretvara vrednosti različitih tipova u nisku koju vraća kao rezultat. Niska će imati dužinu koja zavisi od tipa argumenta
<code>char(niska_arg1, dužina_arg2)</code>	vraća prvih <i>dužina_arg2</i> karaktera niske <i>niska_arg1</i>
<code>insert(niska_arg, početak_arg, dužina_arg, zamena_arg)</code>	podnisku dužine <i>dužina_arg</i> koja počinje na poziciji <i>početak_arg</i> u okviru niske <i>niska_arg</i> zamenjuje sa <i>zamena_arg</i>
<code>locate_in_string(niska_arg1, podniska_arg2, počevšiod_arg3, brpojavljivanja_arg4)</code>	vraća poziciju <i>brpojavljivanja_arg4</i> . pojavljivanja niske <i>podniska_arg2</i> u okviru niske <i>niska_arg1</i> počevši od pozicije <i>počevšiod_arg3</i>
<code>locate(podniska_arg1, niska_arg2, pozicija_arg3)</code>	vraća poziciju prvog pojavljivanja niske <i>podniska_arg1</i> u okviru niske <i>niska_arg2</i> počevši od pozicije <i>pozicija_arg3</i>
<code>translate(niska_arg1, zamenisa_arg2, zameni_arg3, podrazumevano_arg4)</code>	vraća nisku <i>niska_arg1</i> u kojoj su karakteri navedeni u <i>zameni_arg3</i> zamenjeni sa <i>zamenisa_arg2</i> . Karakter koji se menja i njegova zamena se uparuju prema poziciji. Karakterom <i>podrazumevano_arg4</i> se zamenjuju karakteri u <i>zameni_arg3</i> koji nemaju svog para u <i>zamenisa_arg2</i> .

Tabela 13.10: Opis nekih skalarnih funkcija za rad sa niskama

■ **Primer 13.49** Izdvojiti prva dva slova iz naziva predmeta i poslednja dva slova iz naziva predmeta.

```
select left(naziv,2), right(naziv,2)
from predmet
```

Primer rezultata za Algoritmi i strukture podataka
Al ka

■

■ **Primer 13.50** Prikazati

- broj 45 kao nisku i odrediti dužinu tako dobijene niske;
- broj 475.6 kao nisku i odrediti dužinu tako dobijene niske;
- nisku *Tekst* sa 10 karaktera;
- nisku *Ovo je niska karaktera* sa 10 karaktera.

```
select char(45), length(char(45)), length(rtrim(char(45))),
       char(475.6), length(char(475.6)), length(rtrim(char(475.6))),
       char('Tekst',10), length(char('Tekst',10)),
       char('Ovo je niska karaktera',10)
from sysibm.sysdummy1
```

Rezultat: 45 11 2 475.6 6 5 Tekst 10 Ovo je nis

■ **Primer 13.51** Prikazati šifre predmeta i šifre u kojima je

- podniska od 2. do 4. pozicije zamenjena sa niskom *abab*;
- na početak dodata niska *ss*.

```
select sifra, insert(sifra,2,2,'abab'), insert(sifra,1,0,'ss')
from predmet
```

Primer rezultata:

M111 Mabab1 ssM111

■ **Primer 13.52** Odrediti na kojoj poziciji počinje niska *ra* u reči *Abrakadabra*.

```
values posstr('Abrakadabra','ra')
```

ili

```
values locate('ra','Abrakadabra',1)
```

ili

```
values locate('ra','Abrakadabra')
```

ili

```
values locate_in_string('Abrakadabra','ra',1,1)
```

ili

```
values locate_in_string('Abrakadabra','ra')
```

Rezultat: 3

- **Primer 13.53** Odrediti poziciju prvog pojavljivanja niske *ra* u reči *Abrakadabra* počevši od 4. pozicije.

```
values locate('ra','Abrakadabra',4)
```

ili

```
values locate_in_string('Abrakadabra','ra',4,1)
```

ili

```
values locate_in_string('Abrakadabra','ra',4)
```

Rezultat: 10

- **Primer 13.54** Odrediti poziciju trećeg pojavljivanja niske *ra* u reči *Abrakadabrara*.

```
values locate_in_string('Abrakadabrara','ra',1,3)
```

Rezultat: 12

- **Primer 13.55** Prikazati šifre predmeta i šifre u kojima je

- 0 zamenjena sa 9;
- 1 zamenjena sa 8;
- 5 i 8 zamenjeni sa -.

```
select sifra, translate (sifra, '98', '0158', '-')  
from predmet
```

Primer rezultata M105 M89-

13.5 Predavanje 5

U tabeli 13.11 je dat opis nekih skalaranih funkcija za rad sa datumima, a u tabeli 13.12 za rad sa vremenom.

- **Primer 13.56** Prikazati današnji datum u različitim formatima.

Funkcija	Opis
<code>date(arg1)</code>	vraća vrednost tipa <i>date</i> za <i>arg1</i> , a datum se zadaje kao niska
<code>cast(arg1 as arg2)</code>	vraća vrednost <i>arg1</i> tipa <i>arg2</i>
<code>timestamp(arg1, arg2)</code>	vraća vrednost tipa <i>timestamp</i> definisanu argumentima od kojih <i>arg1</i> predstavlja datum, a <i>arg2</i> vreme
<code>char(datetime_arg, format_arg)</code>	vraća datum ili vreme koji su zadatih sa <i>datetime_arg</i> u željenom formatu koji se zadaje sa <i>format_arg</i>
<code>year(datum_arg1)</code>	vraća godinu iz <i>datum_arg1</i> , rezultat je ceo broj
<code>month(datum_arg1)</code>	vraća mesec iz <i>datum_arg1</i> , rezultat je ceo broj
<code>day(datum_arg1)</code>	vraća dan iz <i>datum_arg1</i> , rezultat je ceo broj
<code>week(datum_arg1)</code>	vraća sedmicu u godini za <i>datum_arg1</i> (rezultat je u intervalu 1- 54, sedmica se broji od nedelje)
<code>dayofyear(datum_arg1)</code>	vraća dan u godini za <i>datum_arg1</i>
<code>dayofweek(datum_arg1)</code>	vraća redni broj dana u sedmici za <i>datum_arg1</i> (rezultat u intervalu od 1 do 7, gde 1 označava nedelju)
<code>dayname(datum_arg1)</code>	vraća naziv dana u sedmici za <i>datum_arg1</i>
<code>monthname(datum_arg1)</code>	vraća naziv meseca za <i>datum_arg1</i>
<code>monthname(datum_arg1, zapis_arg2)</code>	vraća naziv meseca za <i>datum_arg1</i> u zapisu <i>zapis_arg2</i> videti cldr.unicode.org
<code>last_day(datum_arg1)</code>	vraća datum poslednjeg dana meseca u kome je <i>datum_arg1</i>
<code>next_day(datum_arg1, dan_arg2)</code>	vraća datum koji odgovara prvom datumu koji je posle <i>datum_arg1</i> i čije je ime dana zadato sa <i>dan_arg2</i> , koji može biti MON, TUE, WED ...
<code>days(datum_arg1)</code>	vraća celobrojnu reprezentaciju datuma

Tabela 13.11: Opis funkcija za rad sa datumima

```
values (char(current date, EUR), 'EUR'),  
       (char(current date, USA), 'USA'),  
       (char(current date, ISO), 'ISO'),  
       (char(current date, JIS), 'JIS'),  
       (char(current date, LOCAL), 'LOCAL')
```

Primer rezultata

30.10.2022 EUR

10/30/2022 USA

2022-10-30 ISO

2022-10-30 JIS

10-30-2022 LOCAL

■ **Primer 13.57** Zapisati datum 5.5.2023.

```
values date('5.5.2023')
```

ili

```
values cast('5.5.2023' as date)
```

Primer rezultata: 2023-05-05

■ **Primer 13.58** Zapisati kombinaciju datuma 5.5.2023. i vremena 1.02.

```
values timestamp('05/05/2023', '1.02')
```

Primer rezultata: 2023-05-05 01:02:00.0

■ **Primer 13.59** Izdvojiti za datum kada se izvršava naredba: redni broj nedelje u godini, godinu, mesec i dan.

```
select week(current date) redni_broj_nedelje ,  
       year(current date) godina,  
       month(current date) mesec,  
       day(current date) dan  
from sysibm.sysdummy1
```

■ **Primer 13.60** Odrediti redni broj dana u nedelji i ime za datum izvršavanja naredbe.

```
select dayofweek_iso(current date) dan_u_nedelji_iso,
       dayofweek(current date) dan_u_nedelji,
       dayname(current date) ime_dana
from sysibm.sysdummy1
```

Primer rezultata: 7 1 Sunday

■ **Primer 13.61** Ispisati ime dana za datum izvršavanja naredbe.

```
select dayname(current_date),
       dayname(current_date, 'CLDR181_sr_SR'),
       dayname(current_date, 'CLDR181_sr_Latn_SR')
from sysibm.sysdummy1
```

Primer rezultata: Sunday nedelja

■ **Primer 13.62** Izdvojiti

- dan u godini za datum izvršavanja naredbe;
- poslednji dan u mesecu za mesec u kome se izvršava naredba;
- poslednji dan u mesecu za datum 31.12.2022;
- datum za prvu nedelju nakon dana kada se izvršava naredba.

```
select dayofyear(current date) dan_u_godini,
       last_day(current date) poslednji_u_mesecu,
       last_day('31.12.2022') poslednji_u_mesecu2,
       next_day(current date, 'SUN') sledeca_nedelja
from sysibm.sysdummy1
```

Primer rezultata: 303 2022-10-31 2022-12-31 2022-11-06

■ **Primer 13.63** Odrediti koliko je dana prošlo od 1.1.0001. do a) 1.1.0001. i do b) dana kada se izvršava naredba.

```
values (days('1.1.0001') , days(current date))
```

Primer rezultata: 1 738458

■
Na dva datuma je moguće primeniti razliku pri čemu je rezultat ceo broj u obliku *ymmd*, gde *y* označava broj godina koje su protekle između navedenih datuma, *mm* broj meseci, a

dd broj dana. Broj meseci i dana se uvek označava sa 2 cifre, pri čemu se vodeće nule ne ispisuju.

Na neki datum je moguće dodati ili oduzeti neki broj kojem se obavezno dodeljuje i značenje pomoću reči

- *day*, *days* kada se dodaju ili oduzimaju dani;
- *month*, *months* kada se dodaju ili oduzimaju meseci;
- *year*, *years* kada se dodaju ili oduzimaju godine.

Takođe, i na neko vreme je moguće dodati ili oduzeti neki broj kojem se obavezno dodeljuje i značenje pomoću reči

- *second*, *seconds* - kada se dodaju ili oduzimaju sekunde;
- *minute*, *minutes* - kada se dodaju ili oduzimaju minuti;
- *hour*, *hours* - kada se dodaju ili oduzimaju sati.

■ **Primer 13.64** Odrediti:

- koji je datum bio pre godinu dana;
- koje će vreme biti za 3 sata 20 minuta i 10 sekund.

```
select current date - 1 year,  
       current time + 3 hour + 20 minute + 10 seconds  
from sysibm.sysdummy1
```

Primer rezultata: 2021-10-30 17:12:31

■ **Primer 13.65** Koliko vremena je prošlo od 5.5.2020?

```
values current date - date('5.5.2020')
```

Primer rezultata:20525

■ **Primer 13.66** Prikazati trenutno vreme u različitim formatima.

```
values (char(current time, EUR), 'EUR'),  
       (char(current time, USA), 'USA'),  
       (char(current time, ISO), 'ISO'),  
       (char(current time, JIS), 'JIS'),  
       (char(current time, LOCAL), 'LOCAL')
```

Primer rezultata:

13.44.04 EUR

01:44 PM USA

13.44.04 ISO

13:44:04 JIS

13:44:04 LOCAL

Funkcija	Opis
time(arg1)	vraća vrednost tipa time, a vreme se zadaje kao niska
hour(arg1)	izdvaja sate iz vremena zadatog sa <i>arg1</i>
minute(arg1)	izdvaja minute iz vremena zadatog sa <i>arg1</i>
second(arg1)	izdvaja sekunde iz vremena zadatog sa <i>arg1</i>
microsecond(arg1)	izdvaja mikrosekunde iz vremena zadatog sa <i>arg1</i>

Tabela 13.12: Skalarne funkcije za rad sa vremenom

■ **Primer 13.67** Izdvojiti minute, sekunde i mikrosekunde iz trenutka u kome se izvršava naredba.

```
select minute(current time) minuta,
       second(current time) sekundi,
       microsecond(current timestamp) mikrosekundi
from sysibm.sysdummy1
```

Primer rezultata: 7 48 501459

■

U tabeli 13.13 je dat opis nekih funkcija za konverziju i rad sa brojevima.

Funkcija	Opis
integer(arg1)	konvertuje vrednost <i>arg1</i> (broj ili niska) u tip integer
decimal(arg1, arg2, arg3)	vraća decimalnu reprezentaciju <i>arg1</i> (broj ili niska) pri čemu <i>arg2</i> predstavlja ukupan broj cifara, a <i>arg3</i> broj mesta iza decimalne tačke
digits(arg1)	vraća <i>arg1</i> kao nisku koja sadrži vrednost <i>arg1</i> bez znaka ili decimalne tačke. Vodeće 0 će biti dodate ako je potrebno da bi rezultat imao minimalnu dužinu koja zavisi od tipa argumenta.
double(arg1)	konvertuje vrednost <i>arg1</i> (broj ili niska) u tip double (zapis u pokretnom zarezu dvostruke tačnosti)
real(arg1)	konvertuje vrednost <i>arg1</i> (broj ili niska) u tip real (zapis u pokretnom zarezu jednostruke tačnosti)
round(arg1, arg2)	vraća vrednost <i>arg1</i> zaokruženu na <i>arg2</i> broj decimala ako je <i>arg2</i> pozitivan broj, a ako je negativan vraća zaokružen broj <i>arg1</i> na <i>arg2</i> broj mesta u levo od tačke
ceil(arg1)	vraća najmanji ceo broj koji je jednak ili veći od <i>arg1</i>
floor(arg1)	vraća najveći ceo broj koji je jednak ili manji od <i>arg1</i>

Tabela 13.13: Funkcije za konverziju i rad sa brojevima

- **Primer 13.68** Prikazati broj 1234.5656 u formatu 8.2.

```
select    decimal('1234.5656',8,2)
from sysibm.sysdummy1
```

Rezultat: 1234.56

- **Primer 13.69** Prikazati samo cifre brojeva 1234.56 i broj 101 kao vrednost tipa double.

```
select digits(1234.56), double(101)
from sysibm.sysdummy1
```

Rezultata: 123456 101.0

- **Primer 13.70** Prikazati rezultat primene funkcija floor i ceil na broj 65.6.

```
select floor(65.6), ceil(65.6)
from sysibm.sysdummy1
```

Rezultata: 65 66

- **Primer 13.71** Zaokružiti broj 873.726 na različite pozicije cifara.

```
select round(873.726,2), round(873.726,1),
       round(873.726,0),round(873.726,-1),
       round(873.726,-2), round(873.726,-3)
from sysibm.sysdummy1
```

Rezultata: 873.730 873.700 874.000 870.000 900.000 1000.000

13.5.1 Agregatne funkcije

Agregatne funkcije se koriste kada je potrebno izvršiti neke operacije nad svim entitetima koji ulaze u rezultat upita[4]. Osnove agregatne funkcije su: brojanje, sabiranje, izračunavanje najveće, najmanje ili srednje vrednosti. Važna pravila pri upotrebi agregatnih funkcija su:

- agregatne funkcije se upotrebljavaju u obliku

`funkcija(lista-kolona)`

- ako je upit oblika

```
select ...
from ...
[where ...]*
[order by ...]*
```

i u klauzi select se koristi agregatna funkcija, tada se u klauzi select mogu navoditi samo agregatne funkcije;

- agregatne funkcije ne mogu da se upotrebljavaju u klauzi where kao deo izraza, ali mogu da se navode u klauzuli select podupita čiji se rezultat koristi kao skalarna vrednost, npr.

```
select ime, prezime
from dosije d
where 7 < (select avg(ocena*1.0) from ispit where d.indeks=indeks)
```

Neke agregatne funkcije su:

- count – koristi se za prebrojavanje entiteta u rezultatu upita i tada se koristi u obliku count(*) ili count, a za prebrojavanje vrednosti koje nisu null u nekoj koloni koristi se u obliku count(ime-kolone). Ako je potrebno prebrojati samo različite i ne null vrednosti mora se navesti ključna reč distinct u obliku count(distinct ime-kolone);
- sum – koristi se za sabiranje vrednosti kolone numeričkog tipa navedene kao argument; koristi se u obliku sum(ime-kolone) ili u sum(distinct ime-kolone) za računanje zbira nad različitim vrednostima;
- max - koristi se za računanje najveće vrednosti kolone; koristi se u obliku max(ime-kolone); min - koristi se za računanje najmanje vrednosti kolone; koristi se u obliku min(ime-kolone);
- avg - koristi se za računanje srednje vrednosti kolone numeričkog tipa; koristi se u obliku avg(ime-kolone);
- stddev – koristi se za računanje standardne devijacije vrednosti kolone numeričkog tipa; koristi se u obliku stddev(ime-kolone);
- correlation – koristi se za računanje korelacije vrednosti zadatih kolona numeričkog tipa; koristi se u obliku correlation(ime-kolone1, ime-kolone2).

■ **Primer 13.72** Naći ukupan broj studenata koji su upisani školske 2013/2014. godine.

```
select count(*)
from dosije
where indeks/10000=2013
```

ili

```
select 'Ukupan broj upisanih studenata skolske 2013/2014 godine je',
       count(*)
from dosije
where indeks/10000=2013
```

ali ne može

```
select indeks, count(*)  
from   dosije  
where  indeks/10000=2013
```

kao ni

```
select indeks/10000, count(*)  
from   dosije  
where  indeks/10000=2013
```

■

13.5.2 Grupisanje rezultata

U SQL-u je moguće grupisati entitete koji zadovoljavaju uslove upita prema vrednostima jedne ili više kolona, tako da u jednoj grupi budu svi entiteti koji imaju iste vrednosti u kolonama po kojima se vrši grupisanje. Tada je moguće primeniti agregatne funkcije nad svakom grupom entiteta posebno. Nakon grupisanja entiteta, kolone po kojima se vrši grupisanje se mogu navesti u okviru klauzule `select`, kao i agregatne funkcije. Grupisanje se zadaje pomoću klauzule `group by` koja sledi iza klauzule `where`:

```
select lista-kolona, agregatne-funkcije  
from   tabele  
where  uslovi  
group by lista-kolona  
order by lista-kolona
```

Klauzula `group by` se izvršava posle klauzule `where`. Ukoliko se koristi klauzula `group by` sve kolone koje se navode u okviru klauzule `select` bez primene agregatne funkcije nad tom kolonom moraju se navesti u okviru klauzule `group by`.

■ **Primer 13.73** Prikazati datume kada su polagani ispiti i za svaki datum ukupan broj studenata koji su tog datuma polagali neki od ispita.

```
select datum_ispita, count(*) as "Broj studenata koji su polagali"  
from   ispit  
group by datum_ispita
```

■

■ **Primer 13.74** Prikazati ime i prezime studenta i ukupan broj bodova koje je student do sada sakupio.

```

select ime as "Ime", prezime as "Prezime",
       sum(p.bodovi) as "Položio bodova"
from   dosije d join ispit i on d.indeks=i.indeks
       join predmet p on i.id_predmeta=p.id_predmeta
where  ocena>5
group  by ime,prezime

```

■

■ **Primer 13.75** Naći stvarnu dužinu januarskog ispitnog roka 2015. godine, tj. interval od kada do kada su polagani ispiti u tom roku.

```

select 'Ispiti u januarskom ispitnom roku 2015. godine su polagani od',
       min(datum_ispita), ' do ', max(datum_ispita)
from   ispit
where  godina_roka=2015 and oznaka_roka='jan'

```

■

■ **Primer 13.76** Za svakog studenta prikazati ime, prezime, broj indeksa, najmanju ocenu, najveću ocenu, prosečnu ocenu i standardnu devijaciju ocena koje je dobio na ispitima u školskoj 2013/2014, uključujući i ispite koje nije položio.

```

select ime, prezime, a.indeks, max(ocena) as "Najveca ocena",
       min(ocena) as "Najmanja ocena",
       avg(ocena) as "Prosecna ocena",
       avg(ocena*1.0) as "Prosek ocena*1.0",
       dec(avg(ocena),7,2) as "dec(prosek ocena),7,2)",
       round(avg(dec(ocena,7,2)),2) as "Zaokruzena prosečna decimalna
                                   vrednost ocene" ,
       dec(round(avg(dec(ocena,7,4)),2),7,2) as "Zaokruzena prosečna
                                   decimalna vrednost ocene
                                   prikazana na dve decimale" ,
       stddev(ocena) as "Stddev(ocena)"
from   dosije a join ispit b on a.indeks=b.indeks
group  by ime,prezime,a.indeks

```

13.5.3 Izdvajanje rezultata za željene grupe

U SQL upitu može se koristiti klauzula `having` za izbor grupa koje su od interesa, tj. grupa za koje će se prikazati podaci u rezultatu. Za izbor grupa mogu se navesti uslovi sa agregatnim funkcijama, kao i uslovi koji koriste kolone navedene u klauzuli `group by`.

Zadati uslovi se proveravaju za svaku grupu entiteta koja je nastala primenom klauzule `group by`, i u rezultatu će se naći samo one grupe koje zadovoljavaju uslove navedene u klauzuli `having`. Kada se koristi klauzula `having`, naredba `select` ima oblik

```
select lista-kolona, agregatne-funkcije 5
from tabele                             1
where uslovi                             2
group by lista-kolona                    3
having uslovi-za-grupe                   4
order by lista-kolona                    6
```

Navedeni broj uz klauzule predstavlja redosled izvršavanja klauzula.

■ **Primer 13.77** Prikazati imena i prezimena studenata koji su položili barem dva ispita sa ocenom većom od 7. Izveštaj urediti po imenima i prezimenima studenata.

```
select ime, prezime, a.indeks
from   dosije a, ispit b
where  a.indeks=b.indeks and ocena>7
group by ime, prezime, a.indeks
having count(*)>1
order by ime, prezime
```

■

■ **Primer 13.78** Prikazati broj indeksa, ime i prezime studenta koji je u 2015. godini imao prosečnu ocenu na ispitima veću od 7,5.

```
select indeks, ime, prezime
from   dosije a
where  7.5 < (
        select avg(ocena*1.0)
        from   ispit b
        where  b.indeks=a.indeks and godina_roka=2015
      )
```

■

■ **Primer 13.79** Prikazati uređene četvorke (`naziv_predmeta_1`, `naziv_roka_1`, `naziv_predmeta_2`, `naziv_roka_2`) tako da važi da je predmet sa nazivom `naziv_predmeta_1` u ispitnom roku `naziv_roka_1` položilo više studenata nego predmet sa nazivom `naziv_predmeta_2` u ispitnom roku `naziv_roka_2`.

```
select a.naziv, c.naziv, b.naziv, d.naziv
from   predmet a, predmet b, ispitni_rok c, ispitni_rok d
```

```

where (select count(*) from ispit
      where id_predmeta=a.id_predmeta and oznaka_roka=c.oznaka_roka
      and godina_roka=c.godina_roka and ocena>5
      having count(*)>0
    )
>
(select count(*) from ispit
  where id_predmeta=b.id_predmeta and oznaka_roka=d.oznaka_roka
  and godina_roka=d.godina_roka and ocena>5
  having count(*)>0
)

```

■

Agregatna funkcija listagg

Agregatna funkcija listagg se koristi u obliku

```

listagg(ime-kolone[, separator]?)
[within group (order by lista-kolona sa poretком)]?

```

i pravi jednu nisku spajanjem niski iz zadate kolone (ili izraza). Može se obezbediti niska-separatora koja se dodaje između susednih niski pri spajanju. Sa opcijom `within group (order by lista-kolona sa poretком)` se zadaje redosled po kome će se vršiti spajanje niski.

■ **Primer 13.80** Prikazati nisku koja sadrži imena studenata koja su uređena prema godini njihovog rođenja.

```

select listagg(ime, ' ,')
      within group (order by datum_rođenja)
from   dosije a

```

■

13.5.4 Imenovanje međurezultata

Da bi se poboljšala brzina upita za složene upite može da se koristi klauzula `with` pomoću koje se mogu imenovati rezultati podupita. Klauzula `with` je korisna kada se isti podupit koristi više puta i koristi se u obliku

```

with ime-rezultata1 [(imena-kolona-rezultata)]? as
  (podupit1)

```



```
[,ime-rezultataN [(imena-kolona-rezultata)]? as (podupit)]*
select lista-kolona, agregatne-funkcije
from tabele
where uslovi-za-redove
group by lista-kolona
having uslovi-za-grupe
order by lista-kolona
```

Nakon imenovanja nekog međurezultata, on se u ostatku upita može koristiti na isti način kao i tabela u bazi podataka. Svaka kolona imenovanog međurezultata mora imati ime i imena kolona moraju biti jedinstvena.

■ **Primer 13.81** Izdvojiti parove imena i prezimena studenata čije su prosečne ocene veće od polovine prosečne ocene svih studenata u tabeli ispit.

```
with student_prosek(indeks,ime,prezime,prosek)
as
(select x.indeks,ime, prezime, avg(ocena*1.0)
 from ispit x, dosije y
 where x.indeks=y.indeks
 group by x.indeks, ime, prezime
),
prosek as
(select avg(ocena*1.0) as prosek
 from ispit
)
select A.ime,A.prezime, B.ime, B.prezime, C.prosek
from student_prosek A, student_prosek B, prosek C
where A.prosek>C.prosek/2 and B.prosek>C.prosek/2
and A.indeks<B.indeks
```

■

13.5.5 Izrazi case

Izrazi case omogućavaju da se na osnovu provere važenja jednog ili više uslova izabere odgovarajuća vrednost, ili rezultat izraza, koja će se vratiti za svaki entitet u upitu. Prvi oblik izraza case se koristi za proveru vrednosti određenog izraza koristeći jednakost. Zavisno od vrednosti izraza vratiće se odgovarajući rezultat. Moguće je postaviti i podrazumevanu vrednost koja se vraća ukoliko rezultat izraza nije jednak nijednoj vrednosti navedenoj uz when. Vrednosti rezultata moraju biti kompatibilnih tipova. Prvi oblika izraza case je

```

case izraz
  when vrednost1 then rezultat1
  [when vrednostI then rezultatI]*
  [else rezultat]?
end

```

Drugi oblik izraza case je

```

case
  when uslov1 then rezultat1
  [when uslovI then rezultatI]*
  [else rezultat]?
end

```

koristi se za ispitivanje proizvoljnih uslova. Uslovi se proveravaju redosledom kojim su navedeni u okviru izraza case i biće vraćen rezultat koji je pridružen prvom uslovu koji je tačan.

■ **Primer 13.82** Prikazati tabelu koja sadrži spisak predmeta i identifikacije grupa kojoj ti predmeti pripadaju pri čemu predmeti čija šifra počinje sa

- M pripadaju grupi matematičkih;
- P i R grupi računarskih;
- O grupi opšteobrazovnih predmeta.

Ostali predmeti pripadaju grupi predmeta sa identifikacijom *Nepoznato*.

```

select naziv, case substr(sifra,1,1)
               when 'M' then 'Matematicki'
               when 'P' then 'Racunarki'
               when 'O' then 'Opsteobrazovni'
               when 'R' then 'Racunarski'
               else      'Nepoznato'
               end as "Grupe Predmeta"
from   predmet

```

■

13.5.6 Skalarna funkcija raise_error

Funkcija raise_error obezbeđuje da se vrati greška sa zadatim sql stanjem (eng. sqlstate) i zadatim objašnjenjem greške u obliku niske, dok je kod greške koja se prijavljuje -438. Navodi se u obliku

```
raise_error(sql-stanje, opis-greške)
```

pri čemu sql-stanje mora biti niska dužine 5 koja zadovoljava sledeće uslove:

- svaki karakter može biti cifra od 0 do 9 ili veliko slovo od A do Z;
- sqlstate klasa (prva dva karaktera) ne može biti neka od već definisanih, npr.
 - 00 - koristi se za označavanje uspešnog izvršavanja;
 - 01 - koristi se za označavanje upozorenja;
 - 02 - koristi se za označavanje da nema podataka u rezultatu;
 može se npr. koristiti sql-stanje čiji je prvi karakter 7, 8 ili 9.

■ **Primer 13.83** Prikazati tabelu koja sadrži spisak imena i prezimena studenata i godine upisa. Ako je godina upisa tekuća ili prošla godina, prikazati trenutni prosek ocena. U slučaju da je godina upisa barem za 6 manja od tekuće godine, prijaviti grešku. U svim ostalim slučajevima prikazati ukupan broj položenih ispita.

```
with student_prosek(indeks, ime, prezime, prosek)
as (select x.indeks, ime, prezime, avg(ocena*1.0)
     from ispit x, dosije y where x.indeks=y.indeks
     group by x.indeks, ime, prezime )
select ime, prezime,
case
when (year(current_date)> indeks/10000 - 1) then decimal(prosek)
when (year(current_date)> indeks/10000+6)
then RAISE_ERROR('70014',
                 'Provera da li je izgubljeno pravo na studiranje')
else (select count(*) from ispit where indeks=a.indeks and ocena>5)
end as "Podatak"
from student_prosek a
```

■

13.5.7 Korisnički definisane funkcije

U SQL-u je moguće definisati i korisničke funkcije. Sintaksa najjednostavnijeg oblika korisnički definisane funkcije je

```
create function ime (arg1 tip-arg1,
                    arg2 tip-arg2,
                    ...
                    argN tip-argN)
returns tip-povratne-vrednosti
return izraz-koji-računa-rezultat
```

Izraz koji računa rezultat može da bude jednostavan izraz ili upit.

■ **Primer 13.84** Napisati korisnički definisanu funkciju koja za predmet sa zadatim broj espb bodova kao argument računa cenu slušanja za samofinansirajuće studente. Cena jednog espb boda je 2000 rsd.

```
create function cena(espb_predmeta smallint)
returns float
return espb_predmeta*2000.0
```

Primer upotrebe

```
values cena(10)
```

■ **Primer 13.85** Napisati korisnički definisanu funkciju koja za zadati id predmeta vraća broj studenata koji su taj predmet položili.

```
create function br_polozenih (id int)
returns integer
return select count(distinct indeks)
       from ispit
       where ocena>5 and id=id_predmeta
```

Primer upotrebe

```
select id_predmeta, br_polozenih(id_predmeta)
from predmet
```

Primer rezultata: 1001 7

13.5.8 Dodavanje entiteta

Za dodavanje novih entiteta u tabelu koristi se naredba insert koja se upotrebljava u dva oblika

- za dodavanje konstantnih entiteta u obliku

```
insert into ime-tabele [(lista-kolona)]?
values (lista-vrednosti)
```

- za dodavanje novih redova na osnovu rezultata upita u obliku

```
insert into ime-tabele [(lista-kolona)]?
<upit>
```

U listama se imena kolona i vrednosti razdvajaju sa „. Ako se ne navedu imena kolona za koje će biti zadate vrednosti, moraju se zadati vrednosti za sve kolone i to redosledom kako su kolone definisane u tabeli. Za kolone koje mogu da sadrže nedostajuću vrednost ili su definisane sa opcijom default ne moraju se navesti vrednosti. Takođe, kao vrednost kolone može se navesti null ili default. U okviru upita je moguće navesti i klauzulu with i tada je redosled klauzula

```
insert into ime-tabele [(lista-kolona)]?
with <ime-rezultata> ...
select ...
from ...
...
```

■ **Primer 13.86** Uneti u tabelu predmet podatke o predmetu Razvoj softvera, koji ima id 4005, šifru R103 i 6 espb.

```
insert into predmet
values (4005, 'R103', 'Razvoj softvera', 6)
```

■

■ **Primer 13.87** Uneti u tabelu predmet podatke o predmetu Uvod u relacione baze podataka, koji ima id 4006, šifru R105 i 6 espb.

```
insert into predmet (sifra, naziv, id_predmeta, bodovi)
values ('R105', 'Uvod u relacione baze podataka', 4006, 6)
```

■

■ **Primer 13.88** U tabelu ispit uneti podatke o polaganju ispita iz predmeta Razvoj softvera (id 4005) za studente iz Beograda koji su polagani u poslednjem održanom roku i na kojima su studenti dobili ocenu 9.

```
insert into ispit
(indeks, id_predmeta, godina_roka, oznaka_roka, ocena)
with poslednji_rok as (
select distinct godina_roka, oznaka_roka
from ispit
where datum_ispita = (select max(datum_ispita)
```

```
        from ispit))
select indeks, 4005, godina_roka, oznaka_roka, 9
from dosije, poslednji_rok
where mesto_rodjenja='Beograd'
```

■

13.5.9 Promena vrednosti tabele

Vrednosti kolona u tabelama se menjaju komandom update koja ima oblik

```
update tabela [as alias]?
set dodela
[where uslov]?
```

Dodele se u klauzuli set navode u obliku

```
ime-kolone=izraz
```

Može se navesti i više dodela koje se razdvajaju sa karakterom „. Umesto izraza mogu se navesti i null ili default ili podupit. Ako se ne navede klauzula where, biće promenjeni svi entiteti tabele.

■ **Primer 13.89** Ažurirati ispite na kojima je polagan predmet Razvoj softvera (id 4005) i postaviti da je dobijeni broj bodova 85.

```
update ispit
set bodovi = 85
where id_predmeta=4005
```

■

■ **Primer 13.90** Ažurirati ispite na kojima je polagan predmet Razvoj softvera (id 4005) od strane studenata koji su rođeni pre više od 20 godina i postaviti da je na tim ispitima nepoznat broj dobijenih bodova i da su ispiti polagani 3 dana nakon poslednjeg ispita u roku u kome su polagani.

```
update ispit as i
set (bodovi, datum_ispita) =
    (null, ( select max(datum_ispita)+ 3 days
              from ispit i1
              where i1.oznaka_roka=i.oznaka_roka
                  and i1.godina_roka=i.godina_roka))
```

```
where id_predmeta=4005
      and indeks in ( select indeks
                      from dosije
                      where datum_rodjenja < current date - 20 years)
```

■

13.6 Predavanje 6

13.6.1 Uklanjanje entiteta

Entiteti iz tabele se uklanjaju naredbom `delete` koja se koristi u obliku

```
delete from tabela
[where uslov]?
```

Kao rezultat primene naredbe biće uklanjanju svi entiteti koji zadovoljavaju uslov zadat u klauzi `where`. Ako se ne navede klauzula `where`, svi postojeći redovi iz tabele će biti obrisani. Za postavljanje složenijih uslova koriste se podupiti u okviru klauzule `where`.

■ **Primer 13.91** Obrisati sve ispite na kojima je polagan predmet Razvoj softvera (id 4005).

```
delete from ispit
where id_predmeta=4005
```

■

■ **Primer 13.92** Obrisati sve ispite na kojima je polagan predmet Razvoj softvera (id 4005) od strane studenata koji su rođeni pre više od 20 godina.

```
delete from ispit
where id_predmeta=4005
      and indeks in ( select indeks
                      from dosije
                      where datum_rodjenja < current date - 20 years)
```

■

13.6.2 Menjanje entiteta tabele na osnovu sadržaja druge tabele/tabela

Naredba `merge` menja entitete u ciljnoj tabeli koristeći podatke iz druge tabele ili rezultata upita. Sintaksa naredbe `merge` je

```
merge into ciljna-tabela [as alias1]?
using upit [as alias2]?
on uslov-spajanja-redova-iz-cilja-i-upita
[when matched [and uslov]? then
    akcija1]*
[when not matched [and uslov]? then
    akcija2]*
```

Redovi u ciljnoj tabeli koji se podudaraju sa nekim redom upita po zadatom uslovu spajanja (when matched) i zadovoljen je dodatni uslov (uslov), ako je naveden, mogu se obrisati ili ažurirati (akcija1). Redovi upita koji nemaju svog para po zadatom uslovu spajanja u ciljnoj tabli mogu biti dodati u ciljnu tabelu (akcija2).

Ako je potrebno sa akcija1 zadati brisanje, navodi se samo delete, a ako je potrebno izvršiti ažuriranje navodi se

```
update
set izraz=dodele
```

Za definisanje akcija2 koristi se sintaksa

```
insert [lista-kolona]?
values (lista-vrednosti)
```

■ **Primer 13.93** Napraviti tabelu dosije_prosek koja ima tri kolone: indeks studenta, prosek studenta i broj položenih ispita.

```
create table dosije_prosek (
    indeks integer not null,
    prosek float,
    broj_polozenih smallint
)
```

■

■ **Primer 13.94** U tabelu dosije_prosek uneti indekse studenata iz Beograda.

```
insert into dosije_prosek (indeks)
select indeks
from dosije
where mesto_rodjenja='Beograd'
```


■ **Primer 13.95** Napisati naredbu za menjanje sadržaja tabele `dosije_prosek` koja

- za studente o kojima već postoje podaci i imaju prosek barem 8,5 ažurira prosek i broj položenih ispita;
- briše podatke o studentima o kojima već postoje podaci u tabeli `dosije_prosek`, a prosek im je manji od 8,5;
- unosi podatke o studentima koji imaju prosek barem 8,5 i o njima nema podataka u tabeli `dosije_prosek`.

```
merge into dosije_prosek dp
using ( select d.indeks, count polozeno, avg(ocena*1.0) prosek
        from dosije d join ispit i
          on d.indeks=i.indeks and ocena>5
        group by d.indeks ) as t
on dp.indeks=t.indeks
when matched and t.prosek>=8.5 then
  update
  set (prosek, broj_polozenih)=(t.prosek, t.polozeno)
when matched and t.prosek<8.5 then
  delete
when not matched and t.prosek>=8.5 then
  insert
  values(indeks, prosek, polozeno)
```

13.6.3 Prikaz željenog broja redova

Za ograničenje broja redova iz rezultata upita koji će biti prikazani može da se koristi klauzula `fetch first`. Klauzula se koristi radi poboljšanja performansi upita sa potencijalno velikim brojem redova u rezultatu i kada je potrebno prikazati samo ograničen broj redova. Klauzula se navodi u obliku

```
fetch first broj-redova rows only
```

i zadaje se posle klauzule `order by`.

■ **Primer 13.96** Prikazati prva dva sloga iz tabele `dosije`.

```
select *
from   dosije
fetch first 2 rows only
```

Umesto klauzule `fetch first` može se koristiti klauzula `limit` za prikaz željenog broja redova.

Klauzula `limit` se navodi u obliku

```
limit broj-redova-za-izdvajanje  
offset broj-prvih-redova-za-preskakanje
```

i zadaje se posle klauzule `order by`.

Sa broj-redova-za-izdvajanje se zadaje maksimalan broj redova koje je potrebno prikazati u rezultatu, ali uz uslov da se preskoči prvih broj-prvih-redova-za-preskakanje redova.

■ **Primer 13.97** Urediti studente po indeksu i prikazati 5 studenata, ali izostaviti prva 3 studenta.

```
select *  
from dosije  
order by indeks  
limit 5 offset 2
```

13.6.4 DDL

Za pravljenje relacione baze podataka i novih objekata u bazi podataka koristi se naredba `create` u obliku

```
create objekat ime-objekta ...
```

dok se za uklanjanje baze podataka ili objekata u bazi podataka koristi naredba `drop` u obliku

```
drop objekat ime-objekta
```

Za menjanje objekta u bazi podataka koristi se naredba `alter` u obliku

```
alter objekat ime-objekta ...
```

U okviru navedenih naredbi objekat može biti

- relacionala baza podataka (database)
- šema (schema);
- tabela (table);
- pogled (view);
- ...

13.6.5 Pravljenje baze podataka

Naredba `create database` inicijalizuje novu bazu podataka i koristi se u obliku

```
create [database|db] ime-baze-podataka  
[alias alias-baze-podataka]?  
[using codeset kod territory teritorija]?  
[pagesize ceo-broj [k]?]?
```

Imena baza podataka moraju biti jedinstvena u direktorijumu baza podataka. Opcijom `alias alias-baze-podataka` definiše se alias za bazu podataka u direktorijumu baza podataka. Ako nije naveden alias, koristi se navedeno ime baze podataka. Klauzulom `using codeset kod` navodi se koji će se kod koristiti za podatke unete u bazu podataka. Nakon što se baza podataka napravi kod se ne može promeniti. Podrazumevani kod za bazu podataka je UTF-8. Sa `territory teritorija` se određuje identifikator teritorije koji će se koristiti za podatke unete u bazu podataka. Nakon što se napravi baza podataka, ne može se promeniti teritorija. Klauzulom `pagesize ceo-broj [k]?` zadaje se veličina stranice podrazumevanog bafera i podrazumevanih prostora tabela. Mogu se navesti sledeći celi brojevi bez sufiksa K: 4096, 8192, 16384 ili 32768 ili sledeće vrednosti sa sufiksom K: 4, 8, 16 ili 32. Najmanje jedan razmak je potreban između celog broja i sufiksa K. Podrazumevana je veličina stranice od 4096 bajtova (4 K). Pri pravljenju nove baza podataka moguće je navesti i druge klauzule, npr. gde će biti sačuvane datoteke pridružene bazi podataka.

■ **Primer 13.98** Napraviti bazu podataka stud.

```
create db stud  
using codeset utf-8 territory sp  
pagesize 32768
```

■

■ **Primer 13.99** Obrisati bazu podataka stud.

```
drop db stud
```

■

13.6.6 Pravljenje šeme baze podataka

Šema je skup objekata baze podataka koji predstavljaju logičku celinu. Neki od objekata baze podataka koje šema može da sadrži su: tabele, pogledi, okidači. Pri pravljenju šeme baze podataka mogu se odmah navesti i naredbe za pravljenje objekata u okviru te šeme, kao i ime vlasnika šeme (navođenjem klauzule *authorization*) kome se dodeljuje pravo da u novoj šemi pravi, briše ili menja objekte. Ako se ne navede ime vlasnika šeme, podrazumevano je vlasnik šeme korisnik koji izvršava naredbu. Naredba za pravljenje šeme baze podataka je oblika

```
create schema ime-sheme
[authorization ime-vlasnika-sheme]?
[create table ...]?
[create view ... ]?
[create index ... ]?
```

Pri pravljenju novih objekata u postojećoj šemi, potrebno je u okviru *create* naredbe navesti ime u obliku *ime-sheme.ime-objekta*.

Šema se briše iz baze podataka naredbom

```
drop schema ime-sheme restrict
```

Ukoliko šema sadrži neki objekat, ne može biti obrisana.

■ **Primer 13.100** Napraviti šemu stud.

```
create schema stud
```

■

■ **Primer 13.101** Obrisati šemu stud.

```
drop schema stud restrict
```

■

■ **Primer 13.102** Napraviti šemu stud i u njoj tabelu semestar sa dve kolone: *sk_godina* i *semestar*.

```
create schema stud
  create table semestar (
    sk_godina smallint,
    semestar smallint )
```

■

13.6.7 Pravljenje tabele

Za pravljenje tabele se koristi naredba

```
create table [shema]? .ime-tabele (
  def-kolone [, def-kolone]*
  [, def-primarni-ključ]?
  [, def-strani-ključ]*
  [, def-uslov-ogranicjenja]* )
```

Definicija kolone def-kolone ima oblik

```
ime-kolone tip-podataka
[not null]?
[[with]? default [vrednost]]?
[generated always as identity
[start with vrednost]?
[increment by vrednost]?
[minvalue vrednost]?
[maxvalue vrednost]]?
```

Kao tip kolone (tip-podataka) može se navesti jedan od osnovnih ugrađenih tipova podataka:

- za cele brojeve:
 - smallint koji koristi 16 bita za čuvanje vrednosti;
 - integer ili int koji koristi 32 bita za čuvanje vrednosti;
 - bigint koji koristi 64 bita za čuvanje vrednosti;
- za realne brojeve:
 - real koji se koristi za čuvanje vrednosti u jednostrukoj tačnosti;
 - double koji se koristi za čuvanje vrednosti u dvostrukoj tačnosti;
 - decimal(p , q) koji se koristi za zapis vrednosti u fiksnom zarezu, gde je p ukupan broj cifara koje se koriste za čuvanje vrednosti, a q je broj cifara za čuvanje razlomljenog dela.
- za jednobajtno niske:

- `char(n)` ili `character(n)` za niske fiksne dužine, gde je n od 1 do 255;
- `varchar(n)` za niske promenljive dužine, gde je n od 1 do 32672;
- `clob(nc)` za niske promenljive dužine do n jedinica c , gde c može biti:
 - * K
 - * M
 - * G (n može biti 1-2)
- za dvobajtnne niske:
 - `graphic(n)` za niske fiksne dužine, gde je n u opsegu 1 do 127;
 - `vargraphic(n)` za niske promenljive dužine, gde je n do 16336;
- za datume i vremena:
 - `time` za vreme;
 - `date` za datum;
 - `timestamp` za datum i vreme.

Kolone podrazumevano mogu da sadrže nedostajuće (null) vrednosti. Opcija `not null` postavlja ograničenje da kolona ne može da sadrži nedostajuće vrednosti. Opcija `default` postavlja podrazumevanu vrednost kolone koja će biti dodeljena novom entitetu ukoliko za njega nije definisana vrednost za tu kolonu. vrednost u opciji `default` ne mora da se navede, i u tom slučaju će se koristiti predefinisana vrednost prema tipu kolone, npr. belina tj. blank za tip `char` ili 0 za tip `int`). U RSUPB DB2 se u definiciji kolone može postaviti da vrednost kolone bude jedinstvena i da se automatski računa korišćenjem opcije

```
generated always as identity
[start with vrednost]?
[increment by vrednost]?
[minvalue vrednost]?
[maxvalue vrednost]?
```

u okviru koje je moguće navesti minimalnu ili maksimalnu vrednost u koloni, kao i vrednost koja će biti dodeljena prvom entitetu koji se unese u tabelu ili za koliko će se razlikovati vrednosti dva uzastopno uneta entiteta.

Definicija primarnog ključa je oblika

```
[constraint ime]? primary key (lista-imena-kolona)
```

Svaka kolona koja čini primarni ključ mora biti eksplicitno definisana sa opcijom `not null`. Definicija primarnog ključa koji se sastoji od jedne kolone može se zadati i u samoj definiciji te kolone, navođenjem opcije `primary key`.

Definicija stranog ključa je oblika

```
[constraint ime]? foreign key (lista-kolona>)  
    references ime-bazne-tabele  
    [on delete akcija]?
```

Strani ključ koji se sastoji od jedne kolone može se zadati i u samoj definiciji te kolone, navođenjem kompletne definicije stranog ključa.

Sa opcijom `on delete` definiše se koja se akcija primenjuje na redove u zavisnoj tabeli pri pokušaju brisanja reda u baznoj tabeli na koje oni referišu. Može se primeniti jedna od sledećih akcija brisanja[4]:

- `no action` koje omogućuje brisanje entiteta u roditelj tabeli samo ako ne postoji entitet u dete tabeli koji zavisi od entiteta koji je potrebno obrisati (tj. ako ne postoji entitet u dete tabeli čija je vrednost stranog ključa jednaka vrednosti primarnog ključa entiteta koji se briše). Ovo je i podrazumevana akcija.
- `restrict` akcija ima isti efekat kao i akcija `no action`. Navedene dve akcije se razlikuju ukoliko na roditeljsku tabelu referiše više dete tabela, jer se pri pokušaju brisanja prvo proveravaju zavisne tabele sa definisnom `restrict` akcijom, a poslednje zavisne tabele sa definisnom `no action` akcijom.
- `set null` akcija omogućava brisanje entiteta u roditeljskoj tabeli i postavljanje null vrednosti u kolone stranog ključa zavisne tabele ukoliko je to moguće;
- `cascade` akcija omogućava brisanje entiteta iz roditelj tabele, ali će biti obrisani i svi zavisni entiteti, uz poštovanje pravila brisanja njihovih zavisnih tabela.

Uslov ograničenja je logički izraz u kome mogu učestvovati kolone tabele, konstante i izrazi nad njima, i predstavlja ograničenje nad podacima koji se unose ili manjaju u tabeli. Uslova ograničenja se zadaje u obliku

```
constraint ime check ( uslov)
```

13.6.8 Korisnički definisani tipovi

Pored ugrađenih tipova podataka, mogu se koristiti i korisnički definisani tipovi. Korisnički definisani tipovi (KDT) imaju reprezentaciju kao njihov izvorni tip koji je jedan od ugrađenih tipova, ali se novi tip smatra drugačijim i nekompatibilnim sa izvornim tipom za većinu operacija. Naredba za pravljenje novog tipa ima oblik

```
create distinct type ime-novog-tipa as izvorni-tip  
[with comparisons]?
```

Za novonapravljeni tip dostupni su samo operatori za poređenje dve vrednosti tog tipa,

kao i dve funkcije za konverziju:

- funkcija za konverziju vrednosti izvornog tipa u novi tip (`ime-novog-tipa(arg)`);
- funkcija za konverziju vrednosti novog tipa u izvorni (`izvorni-tip(arg)`).

Za proširivanje mogućnosti korišćenja novih tipova potrebno je definisati odgovarajuće funkcije.

■ **Primer 13.103** Napraviti korisnički definisan tip `prosek` nad tipom `dec(7,2)`.

```
create distinct type
    prosek as dec(7,2)
    with comparisons
```

Poređenje u sledećem upitu je moguće jer su operatori za poređenje dve vrednosti tipa `prosek` napravljeni kada je definisan tip `prosek`.

```
select *
from dosije
where prosek(8.5)>prosek(7.5)
```

Poređenje u sledećem upitu nije moguće jer se vrednosti tipa `prosek` i vrednost nekog ugrađenog tipa ne mogu porediti.

```
select *
from dosije
where prosek(8.5)>7.5
```

■
■ **Primer 13.104** Napraviti korisnički definisan tip `država` čiji je izvorni tip `varchar(30)`.

```
create distinct type
    drzava as varchar(30)
    with comparisons
```

■ **Primer 13.105** Napraviti korisnički definisan tip `char10` čiji je izvorni tip `varchar(10)`.

```
create distinct type char10 as char(10)
```

■ **Primer 13.106** Napraviti tabelu `student` u shemi `stud`.


```

create table stud.student (
    indeks            integer      not null,
    korisnik          char10       not null,
    ime               char(10)     not null with default user,
    prezime           varchar(15)  not null,
    god_rodjenja      smallint     ,
    mesto_rodjenja    varchar(20)  ,
    tekuci_prosek     prosek       not null with default prosek(0),
    drzava_rodjenja   drzava       with default 'Srbija',
    primary key       (indeks)     ,
    constraint god_indeksa check (indeks/10000>=2010)
)

```

■

13.6.9 Promena bazne tabele

Postojeća tabela u bazi podataka se može izmeniti dodavanjem ili brisanjem kolona, primarnog ključa, stranih ključava, opštih uslova ograničenja, ili izmenom postojećih kolona. Naredba za menjanje tabele se koristi u obliku

```

alter table ime-tabele
[add def-kolone]*
[add def-prim-ključa]?
[add def-str-ključa]*
[add uslov-ograničenja]*
[drop column ime-kolone]?
[drop primary key]?
[drop foreign key ime-ograničenja]*
[drop check ime-ograničenja]*
[drop constraint ime-ograničenja]*
[alter column ime-kolone set data type tip]*
[alter column ime-kolone set default vrednost]*
[alter column ime-kolone drop default]*

```

Klauzula `add` se koristi za dodavanje nove kolone ili ograničenja. Nova kolona ili ograničenje, uključujući primarni i strani ključ, se definišu na isti način kao u okviru naredbe za pravljenje tabele. Klauzula `drop` se koristi za uklanjanje postojeće kolone ili ograničenja, a klauzula `alter column` za izmenu postojeće kolone (npr. za promenu tipa kolone (`set data type tip`), postavljanje podrazumevane vrednosti (`set default vrednost`) ili uklanjanje podrazumevane vrednosti (`drop default`)).

■ **Primer 13.107** Tabeli `stud.student` dodati ograničenje da student mora biti rođen posle

1950. godine i postaviti da je podrazumevana vrednost kolone korisnik id korisnika koji izvršava naredbu.

```
alter table stud.student
    add constraint starost check (god_rodjenja>1950)
    alter column korisnik set default user
```

■

13.6.10 Indeksi

Indeks je mehanizam koji ubrzava pristup vrstama tabele i može se posmatrati kao uređeni skup pokazivača na vrste bazne tabele[6]. Indeksi se prave na osnovu vrednosti jedne ili više kolona tabele i uređuju na osnovu vrednosti tih kolona u rastućem (asc) ili opadajućem (desc) poretku. Naredba za pravljenje indeksa se koristi u obliku

```
create [unique]? index ime-indeksa on tabela
(
    ime-kolone [poredak]? [, ime-kolone [poredak]? ]*
)
```

Opcija unique se koristi kada je potrebno osigurati jedinstvenost vrednosti za svaki red tabele po koloni ili kolonama na osnovu kojih se pravi indeks. U RSUBP DB2 se na osnovu kolona primarnog ključa tabele pravi indeks sa opcijom unique.

■ **Primer 13.108** Napraviti indeks koji obezbeđuje da ne mogu da postoje dva studenta sa istim imenom.

```
create unique index studime on stud.student(ime)
```

■

13.6.11 Dodatni primeri

■ **Primer 13.109** Napraviti korisnički definisane tipove datum i vreme.

```
create distinct type datum as date with comparisons;
create distinct type vreme as time with comparisons;
```

■

■ **Primer 13.110** U šemi stud napraviti tabelu ispit koja pored informacija koje ima tabela ispit sadrži vreme ispita i redni broj prijave.

```

create table stud.ispit (
    indeks            integer      not null
    id_predmeta       integer      not null
    godina_roka       smallint     not null
    oznaka_roka       char(5)      not null
    ocena             smallint     not null with default 5
    datum_ispita      datum
    vreme_ispita      vreme
    rbr_prijave       integer      not null generated always
                                as identity (start with 1)
    primary key (indeks, id_predmeta, godina_roka,
                oznaka_roka,rbr_prijave),
    foreign key (godina_roka, oznaka_roka) references ispitni_rok,
    constraint indeks_ref foreign key (indeks) references dosije ,
    foreign key (id_predmeta) references predmet on delete cascade
)

```

■ **Primer 13.111** Unos u tabelu stud.ispit.

```

insert into stud.ispit (indeks, id_predmeta, godina_roka, oznaka_roka,
                        ocena, datum_ispita, vreme_ispita)
select indeks, id_predmeta, godina_roka, oznaka_roka,
                        ocena, datum_ispita, time('9.00.00')
from ispit

```

■ **Primer 13.112** Napraviti tabelu nalik_na_dosije koja ima sve kolone kao i tabela dosije.

```
create table nalik_na_dosije like dosije
```

Pravi se tabela koja ima iste kolone kao i izvorna tabela (dosije), ali ne važi nijedno ograničenje koje je definisano za izvornu tabelu (primarni ključ, strani ključ, ...).

■ **Primer 13.113** Napraviti korisnički definisan tip stipendija nad tipom decimal(7,2) i funkciju za sabiranje dve vrednosti tog tipa.

```
create distinct type stipendija as dec(7,2) with comparisons;
```

```

create function "+" (stipendija,stipendija)
returns stipendija
source sysibm."+" (dec(7,2),dec(7,2));

```

■ **Primer 13.114** Napraviti tabelu stipendije.

```
create table stipendije (  
    indeks          integer not null ,  
    jmbg            char(15) not null ,  
    ovaj_mesec      stipendija      ,  
    proslimesec     stipendija      ,  
    narednimesec    stipendija      ,  
    primary key     (indeks)  
)
```

Pošto je prethodno definisan operator za sabiranje (prethodni primer), moguće je izvršiti upit:

```
select indeks, ovaj_mesec + narednimesec+ proslimesec  
from   stipendije
```

■

13.6.12 Okidači

Okidač (eng. trigger) je niz akcija koje su pridružene određenom događaju (unos novog reda, promena ili brisanje postojećeg reda) koji vrši promenu sadržaja jedne (ciljne) tabele. Niz akcija definisanih u okviru okidača se izvršava svaki put kada se takav događaj dogodi. Okidači se koriste za definisanje uslova integriteta koji moraju da važe u bazi podataka. Primeri upotrebe okidača su:

- sprečavanje unosa nedozvoljenih vrednosti u tabelu;
- za automatsko generisanje vrednosti koje će biti unete;
- za automatsku izmenu drugih tabela.

Okidači se dele na dve grupe na osnovu vremena kada se izvršava pridružen niz akcija:

- *pre* ili *before* okidači kod kojih se niz akcija izvršava pre nego što se izvrši događaj za koji je vezan okidač. Koriste se npr. za sprečavanje unosa nedozvoljenih vrednosti ili za automatsko generisanje vrednosti koje će biti unete u tabelu.
- *posle* ili *after* okidači kod kojih se niz akcija izvršava nakon nego što se izvrši događaj nad ciljnom tabelom za koji je vezan okidač. Koriste se npr. za automatsku izmenu drugih tabela nakon izmene sadržaja ciljne tabele.

Naredba za pravljenje okidača se koristi u obliku

```
create trigger ime-okidaca  
vreme-izvršavanja-akcija događaj on ciljna-tabela
```

```

[referencing [old as ime-za-red-pre-promene]?
              [new as ime-za-red-posle-promene]? ]?
for each row
[when (uslov)]?
[begin [atomic]?]?
akcija1;
...
akcijaN;
[end]?

```

pri čemu

- vreme izvršavanja akcija okidača može biti before ili after, tj. pre ili nakon što se izvrši događaj za koji je vezan okidač nad ciljnom tabelom;
- događaj može biti unos u ciljnu tabelu (insert), promena sadržaja u ciljnoj tabeli (update) ili brisanje sadržaja iz ciljne tabele (delete);
- opcija referencing se koristi za dodelu *imena* novom redu koji će biti u tabeli nakon što se izvrši događaj za koji je vezan okidač (new as ime-za-red-posle-promene), kao i starom redu koji postoji u ciljnoj tabeli pre nego što se izvrši taj događaj (old as ime-za-red-pre-promene);
- opcija for each row označava da će se pridružene akcije izvršiti za svaki red obuhvaćen događajem;
- opcija when (uslov) se koristi za postavljanje dodatnog uslova koji mora da važi da bi se izvršio pridruženi niz akcija;
- blok begin i end se navodi kada se niz akcija sastoji od dve ili više akcija;
- svaka akcija okidača se završava sa karakterom ;

■ **Primer 13.115** Napisati okidač koji sprečava unos podataka u tabelu ispit ako je na novom ispitu ocena za 3 veća od prosečne vrednosti svih prethodnih ocena za isti predmet u istom ispitnom roku za koji se unosi novi ispit.

```

create trigger proveratriger
before insert on stud.ispit
referencing new as nova
for each row
  when (nova.ocena >
        (select avg(ocena)+3
         from   ispit
         where  id_predmeta=nova.id_predmeta
         and    godina_roka=nova.godina_roka
         and    oznaka_roka=nova.oznaka_roka
        )
  )

```

```
signal sqlstate '70123' ('Proveriti ocenu da li je dobro uneta');
```

Pri pokušaju unosa

```
insert into stud.ispit( indeks, id_predmeta, godina_roka, oznaka_roka,
                        ocena, datum_ispita,vreme_ispita)
values
(20140026, 1021, 2015, 'apr', 18,current_date,current_time);
```

javlja se greška:

Application raised error or warning with diagnostic text: "Proveriti ocenu da li je dobro uneta".. SQLCODE=-438, SQLSTATE=70123 ■

■ **Primer 13.116** Napisati okidače koji obezbeđuju da u slučaju postavljanja šifre predmeta na vrednost null postavlja na null i naziv tog predmeta, a ocene u ispitima iz tog predmeta menjaju na negativnu vrednost.

```
create trigger sifratrigger
before update of sifra on predmet
referencing new as nova
for each row
when (nova.sifra is null)
    set nova.naziv = null;

create trigger sifratrigger_ispit
after update of sifra on predmet
referencing new as nova
for each row
when (nova.sifra is null)
    update ispit set ocena=-ocena
    where id_predmeta=nova.id_predmeta;
```

■

13.6.13 Sigurnost u SQL-u

Postoje dva mehanizma koji su, nezavisno jedan od drugog, uključeni u sistem zaštite, a to su:

- **Pogledi**, koji mogu da se koriste za sakrivanje osetljivih podataka od neautorizovanih korisnika
- **Podsistem za autorizaciju**, koji dopušta korisniku sa određenim pravima pristupa da ta prava selektivno i dinamički prenosi na druge korisnike, i/ili da preneti prava povuče

13.6.14 Pogledi

Pogled u relacionom modelu je mehanizam kojim se omogućava da različiti korisnici imaju različiti uvid u podatke. Uvid u podatke se prilagođava izdvajanjem podataka koji su relevantni za prikaz, njihovim kombinovanjem, ili transformisanjem. Pogled je zapravo u suštini imenovani relacioni izraz. Kada se definiše pogled, izraz se sačuva, ali se ne izvršava i ne sadrži zaista podatke. Moglo bi se reći da je pogled virtualna relacija čiji se sadržaj izračunava tek u trenutku kada se zaista i zatraže ti podaci, na primer `select` klauzom.

Efektivno, pogled je prozor kroz koji se gledaju podaci. Svaka izmena nad podacima biće vidljiva preko pogleda ukoliko su podaci obuhvaćeni pogledom. Takođe, svaka izmena putem pogleda će biti izvršena nad podacima nad kojima je pogled definisan. Pogled nema svoje podatke, već je on prozor ka podacima u tabelama nad kojima je definisan.

Pogled može imati više funkcija, a to su da:

- obezbeđuje automatsku zaštitu za sakrivene podatke
- omogućuje da različiti korisnici (istovremeno) vide iste podatke na različite načine
- omogućuje logičku nezavisnost podataka

Logička nezavisnost podataka obezbeđuje da proširenje relacija baze ne sme da ima efekat na izvršavanje aplikativnih programa. Restruktuiranje baze takođe ne sme da ima efekat na postojeće aplikativne programe, pri čemu se podrazumeva da su nova i stara baza informaciono ekvivalentne.

13.6.15 Kreiranje pogleda u SQL

Sintaksa naredbe za kreiranje pogleda u SQL-u je:

```
CREATE [ OR REPLACE ]? VIEW  
<ime-pogleda> AS <izraz-nad-tabelom>  
[WITH [<kvalifikator>]? CHECK OPTION]?
```

Ukoliko se naredba za kreiranje pogleda koristi sa opcijom `OR REPLACE`, tada, ukoliko pogled postoji, biće zamenjen novim. Dok, ukoliko ova opcija nije navedena, a pogled postoji, prijaviće se poruka da pogled ne može da se kreira jer već postoji.

Opcija `WITH CHECK OPTION` se odnosi na način provere ispunjenosti uslova ograničenja prilikom unosa ili izmene podataka, gde kvalifikator može biti `LOCAL` ili `CASCADE`.

Ukoliko se ova opcija ne navede, definicija pogleda se ne koristi za validaciju podataka koji se unose ili menjaju.

Ukoliko je potrebno da se ograniči da se mogu unositi ili dobiti izmenom samo oni podaci koji su obuhvaćeni pogledom, onda se navodi `WITH CHECK OPTION`. Dodatno, sa `LOCAL` se označava da treba samo za neposredno prvi pogled da se proverava uslov ograničenja, dok se sa `CASCADE` (što je podrazumevano ponašanje) označava da treba

da se proveravaju kaskadno uslovi ograničenja i za sve poglede nad kojima je eventualno definisan pogled. Za poglede koji su definisani sa **WITH CHECK OPTION** svakako se proveravaju uslovi ograničenja.

Uslov ograničenja se ne proverava kod ažuriranja postojećih podataka ukoliko **WITH CHECK OPTION** nije navedena.

■ **Primer 13.117** Napraviti pogled kojim se izdvajaju indeks, ime, prezime studenta i njegova prosečna ocena na položenim ispitima.

```
create or replace view prosek_ocena
(indeks, ime, prezime, trenutni_prosek) as
select a.indeks, b.ime, b.prezime, dec(avg(ocena*1.0),4,2)
from ispit a, dosije b
where a.indeks=b.indeks
and a.ocena>5
group by a.indeks, b.ime, b.prezime;
```

upotreba

```
select *
from prosek_ocena;
```

■

13.6.16 Brisanje pogleda u SQL

Sintaksa naredbe za brisanje pogleda je:

```
DROP VIEW <ime-pogleda>
```

■ **Primer 13.118** Obrisati pogled `prosek_ocena`.

```
drop view prosek_ocena;
```

■

13.6.17 Ažuriranje pogleda u SQL-u

Pogledi mogu da se koriste i za ažuriranje. Zlatno pravilo se primenjuje i na poglede, ažuriranje pogleda ne sme da naruši ograničenja integriteta nad pogledima. Ograničenja integriteta nad pogledima su izvedena iz ograničenja integriteta osnovnih relacija.

Postavlja se pitanje, pri ažuriranju nekog pogleda kakve vrste ažuriranja treba izvesti nad osnovnim relacijama da bi se implementiralo originalno ažuriranje pogleda?

Preciznije, ako je D baza, V pogled nad D i X funkcija nad D kojom se definiše pogled V , tada za dati pogled $V = X(D)$ i operaciju ažuriranja U nad V , potrebno je odrediti operaciju ažuriranja $U1$ nad D tako da važi $U(X(D)) = X(U1(D))$. Moguće je da postoji više operacija $U1$. Koju odabrati?

U SQL-u podrška za ažuriranje pogleda je ograničena. Pogledi koji se smatraju mogućim za ažuriranje su pogledi koji su izvedeni iz jedne osnovne tabele preko kombinacija restrikcije i projekcije.

Da bi pogled mogao da se ažurira treba da bude ispunjeno sledeće:

- from klauzula sadrži referencu na tačno jednu tabelu koja je ili osnovna tabela ili pogled koji može da se ažurira
- Izraz kojim se definiše pogled je select izraz koji ne sadrži JOIN, UNION, INTERSECT ili EXCEPT
- select klauzula ne sadrži ključnu reč DISTINCT
- svaka select stavka sadrži ime koje predstavlja referencu na kolonu osnovne tabele
- where klauzula select izraza ne sadrži podupit u kome se from klauzula referiše na istu tabelu kao i from klauzula u select izrazu na najvišem nivou
- select izraz ne sadrži group by niti having klauzulu

■ **Primer 13.119** Napraviti pogled kojim se izdvajaju podaci o predmetima koji imaju 8 bodova i na nekom ispitu su položeni sa ocenom 10.

```
create view predmet_8_10_v1 as
select p.*
from ispit i join predmet p
    on i.id_predmeta = p.id_predmeta
where p.bodovi=8 and i.ocena=10;
```

prolazi:

```
select *
from predmet_8_10_v1
```

ne prolazi:

```
update predmet_8_10_v1
set naziv=naziv||' dopuna';
```

javlja se greška jer je u from klauzuli u definiciji pogleda izvršeno spajanje tabela ■

■ **Primer 13.120** Napraviti pogled kojim se izdvajaju podaci o predmetima koji imaju 8 bodova i na nekom ispitu su položeni sa ocenom 10. Pogled napisati tako da može da se koristiti za unos, menjanje i ažuriranje predmeta.

```
create view predmet_8_10_v2 as
select p.*
from predmet p
where p.bodovi=8 and
      exists (select *
              from ispit i
              where i.id_predmeta = p.id_predmeta
                  and i.ocena=10);
```

prolazi

```
update predmet_8_10_v2
set naziv=naziv||' dopuna';
```

prolazi

```
insert into predmet_8_10_v2
values (5001, 'P501', 'XML tehnologije', 15);
```

ne prolazi jer željeni predmet nije obuhvaćen pogledom

```
delete from predmet_8_10_v2
where id_predmeta=5001;
```

■

13.6.18 WITH CHECK OPTION

■ **Primer 13.121** Napraviti pogled kojim se izdvajaju podaci o predmetima koji imaju 8 bodova. Pogled napisati tako da može da se koristi samo za unos predmeta koji zadovoljavaju uslov pogleda.

```
create view predmet_8_v1 as
select p.*
from predmet p
where p.bodovi=8
with check option;
```

ne prolazi

```
insert into predmet_8_v1
values (5002, 'P502', 'XML tehnologije 2', 15);
```

prolazi

```
insert into predmet_8_v1
values (5002, 'P502', 'XML tehnologije 2', 8);
```

■

LOCAL / CASCADED kvalifikator

Ukoliko pogled u svojoj definiciji ima neke druge poglede nad kojima je definisan, tada će LOCAL / CASCADED kvalifikator odrediti da li se proveravaju uslovi i za potpoglede i to samo za potpoglede koji nisu definisani sa WITH CHECK OPTION. Ukoliko su pogledi definisani sa ovom opcijom, onda se uslovi svakako proveravaju.

■ **Primer 13.122** Napraviti pogled kojim se izdvajaju podaci o predmetima čiji naziv počinje na slovo P, a koji imaju 8 bodova. Pogled definisati nad pogledom predmet_8_v1 iz prethodnog primera (koji je definisan sa with check option). Pogled napisati tako da može da se koristi za unos i onih predmeta koji ne zadovoljavaju uslov pogleda.

```
create view predmet_p_8_v1 as
select p.*
from predmet_8_v1 p
where p.naziv like 'P%';
```

prolazi

```
insert into predmet_p_8_v1
values (5003, 'P503', 'XML tehnologije 3', 8);
```

ne prolazi sa bodovima koji su različiti od 8

```
insert into predmet_p_8_v1
values (5004, 'P504', 'XML tehnologije 4', 15);
```

■

Za pogled predmet_8_v1 nad kojim je definisan pogled predmet_p_8_v1 se svakako proveravaju uslovi zato što je predmet_8_v1 definisan sa with check option.

■ **Primer 13.123** Napraviti pogled kojim se izdvajaju podaci o predmetima koji imaju 8 bodova. Pogled napisati tako da može da se koristi i za unos predmeta koji ne zadovoljavaju uslov pogleda.

```
create view predmet_8_v2 as
select p.*
from predmet p
where p.bodovi=8;
```

■ **Primer 13.124** Napraviti pogled kojim se izdvajaju podaci o predmetima čiji naziv počinje na slovo P, a koji imaju 8 bodova. Pogled definisati nad pogledom predmet_8_v2 (koji je definisan bez with check option). Pogled napisati tako da može da se koristi za unos i onih predmeta koji ne zadovoljavaju uslove pogleda.

```
create view predmet_p_8_v2 as
select p.*
from predmet_8_v2 p
where p.naziv like 'P%';
```

prolazi

```
insert into predmet_p_8_v2
values (5003, 'P503', 'XML tehnologije 3', 15);
```

■ **Primer 13.125** Napraviti pogled kojim se izdvajaju podaci o predmetima čiji naziv počinje na slovo P, a koji imaju 8 bodova. Pogled definisati nad pogledom predmet_8_v2 (koji je definisan bez with check option). Pogled napisati tako da može da se koristi za unos predmeta koji zadovoljavaju njegove uslove, ali se ne proverava da li zadovoljavaju uslove pogleda nad kojim je definisan.

```
create view predmet_p_8_v3 as
select p.*
from predmet_8_v2 p
where p.naziv like 'P%'
with local check option;
```

ne prolazi

```
insert into predmet_p_8_v3
values (5005, 'P505', 'XML tehnologije 5', 8);
```

prolazi

```
insert into predmet_p_8_v3
values (5006, 'P506', 'Programiranje', 15);
```

Primetimo da nije proveravano da li važe ograničenja za pogled predmet_8_v2 nad kojim je definisan pogled predmet_p_8_v3. ■

■ **Primer 13.126** Napraviti pogled kojim se izdvajaju podaci o predmetima čiji naziv počinje na slovo P, a koji imaju 8 bodova. Pogled definisati nad pogledom predmet_8_v2 (koji je definisan bez with check option). Pogled napisati tako da može da se koristi samo za unos predmeta koji zadovoljavaju njegove uslove i uslove svih pogleda nad kojima je definisan.

```
create view predmet_p_8_v4 as
select p.*
from predmet_8_v2 p
where p.naziv like 'P%'
with cascaded check option;
```

ne prolazi

```
insert into predmet_p_8_v4
values (5007, 'P507', 'Programiranje', 15);
```

prolazi

```
insert into predmet_p_8_v4
values (5007, 'P507', 'Programiranje', 8);
```

■

13.6.19 Okidači nad pogledima

Nije moguće ažurirati sve poglede, jer nije sasvim jasno šta bi bile akcije koje bi trebalo da se preduzmu. Međutim, za neke poglede moguće je opisati logiku šta bi bile željene akcije ukoliko se pokuša unos, izmena ili brisanje podataka iz pogleda.

U DB2 se nad pogledima mogu kreirati INSTEAD OF okidači. Kao i kod drugih okidača, i kod INSTEAD OF okidača potrebno je definisati akciju koja ih pokreće, koja može biti INSERT, UPDATE ili DELETE. Razlika je u tome što će se kod INSTEAD OF okidača umesto pokrenute akcije izvršiti samo telo okidača.

■ **Primer 13.127** Neka su kreirane sledeće tabele

```
create table t1(c1 int, c2 float);
```

```
insert into t1 values  
(5, 6.0),  
(6, 7.0),  
(5, 6.0);
```

```
create table t2(c1 int, c2 float);
```

```
insert into t2 values  
(5, 9.0),  
(5, 4.0),  
(7, 5.0);
```

Kreirati pogled kojim se vide podaci iz tabele t1 u kojima su uklonjeni duplikati.

```
create view v7(c1, c2) as  
select distinct c1, c2  
from t1;
```

Ovaj pogled ne može da se ažurira jer sadrži distinct.

Definisati INSTEAD OF okidač kojim se prilikom brisanja nad pogledom v7 brišu svi sa zadatim vrednostima.

```
create trigger v7_delete instead of delete on v7  
referencing old as o  
for each row mode db2sql  
delete from t1  
where o.c1 = c1  
and o.c2 = c2;
```

Prilikom akcije:

```
delete from v7  
where c1 = 5;
```

brišu se svi redovi u kojima c1 ima vrednost 5.

■

13.6.20 Podsystem za autorizaciju

Za autorizovan pristup u SQL-u važi sledeće:

- Da bi se korisnik izvršio neku operaciju nad nekim objektom on mora da poseduje dozvolu (ili autorizaciju) za tu operaciju nad tim objektom
- Tipovi i vrste dozvola nisu isti kod svih SUBP
- Sistemski administrator (SYSADM nivo autorizacije) je inicijalni vlasnik svih dozvola
- Davanje dozvola se vrši GRANT naredbom, a povlačenje REVOKE naredbom

13.6.21 GRANT naredba

Sintaksa za dodeljivanje dozvola nad tabelama ili pogledima je sledeća:

```
GRANT <dozvola> [ ON [<tip>] <objekat> ] TO <korisnik>
```

Pri čemu:

- <dozvola> je lista vrsta dozvola, razdvojenih zarezima ili fraza ALL PRIVILEGES ili ALL (koja označava sve privilegije koje se mogu dati GRANT naredbom)
- <korisnik> je lista korisnika razdvojenih zarezima ili PUBLIC (svi korisnici)
- <objekat> je lista imena jednog ili više objekata (koji su svi istog tipa) razdvojenih zarezima
- <tip> označava tip objekta - ako se izostavi podrazumeva se TABLE

Dozvole koje se odnose na osnovne tabele i poglede su:

- CONTROL
- DELETE
- INSERT
- SELECT
- UPDATE (mogu da se navedu pojedinačne kolone)

Dozvole koje se odnose samo na osnovne tabele su;

- ALTER (dozvola za izvršavanje ALTER TABLE nad tabelom)
- INDEX (dozvola za izvršavanje CREATE INDEX nad tabelom)
- REFERENCES (dozvola za formiranje/brisanje spoljašnjeg ključa koji referiše tu tabelu kao roditelj tabelu)

■ **Primer 13.128** Dati dozvolu čitanja nad tabelom dosije korisniku korisnik01.

```
grant select on table dosije to korisnik01;
```

■

■ **Primer 13.129** Dati dozvolu čitanja nad tabelom predmet i ažuriranja atributa sifra i naziv nad tabelom predmet korisnicima korisnik02, korisnik03, korisnik09.

```
grant select, update (sifra, naziv) on table predmet to  
korisnik02, korisnik03, korisnik09;
```

■ **Primer 13.130** Dati sve dozvole korisnicima korisnik76, korisnik77 nad tabelama dosije, predmet i ispit.

```
grant all privileges on table dosije, predmet, ispit to  
korisnik76, korisnik77;
```

■ **Primer 13.131** Svima dati dozvolu čitanja nad tabelom dosije.

```
grant select on table dosije to public;
```

■ **Primer 13.132** Korisniku korisnik99 dati dozvolu brisanja iz tabele ispitni_rok.

```
grant delete on ispitni_rok to korisnik99;
```

13.6.22 REVOKE naredba

Sintaksa naredbe za povlačenje dozvole je:

```
REVOKE <dozvola> [ ON [<tip>] <objekat> ] FROM <korisnik> [BY ALL]
```

■ **Primer 13.133** Povuci dozvolu za čitanje nad tabelom kvota korisniku k2.

```
revoke select on table kvota from k2;
```

Nije moguće ukinuti UPDATE dozvolu samo za pojedine kolone.

Povlačenje dozvole za nekog korisnika uzrokuje da svi planovi/paketi zasnovani na toj dozvoli postanu neispravni i uzrokuju automatsko vezivanje/ponovno vezivanje prilikom pozivanja takvog plana/paketa.

13.6.23 GRANT naredba - WITH GRANT OPTION

Ako korisnik k1 želi da prenese dozvolu D korisniku k2, to može da uradi:

- naredbom GRANT dozvola
- naredbom GRANT dozvola ... WITH GRANT OPTION čime omogućuje korisniku k2 da dalje distribuira dozvolu koja mu je prenet

■ **Primer 13.134** Neka je dato

Korisnik k1:

```
grant select on table dosije to k2 with grant option;
```

Korisnik k2:

```
grant select on table dosije to k3 with grant option;
```

Korisnik k3:

```
grant select on table dosije to k4 with grant option;
```

Tada povlačenje dozvole

```
revoke select on table dosije from k2;
```

prouzrokuje lančano povlačenje dozvole i za korisnike k3, k4, ...

■

13.7 Predavanje 7

13.7.1 OLAP

13.7.2 Unakrsno tabeliranje

U okviru OLAP-a mogu se unakrsno prikazivati i analizirati podaci.

■ **Primer 13.135** Posmatrajmo sledeće upite nad bazom:

- Naći prosečne ocene na položenim ispitima
- Naći prosečne ocene na položenim ispitima po predmetima
- Naći prosečne ocene na položenim ispitima po ispitnom roku (bez obzira na godinu)
- Naći prosečne ocene na položenim ispitima po predmetima i roku (bez obzira na godinu roka)

Odgovarajući upiti koji su rešenje su:

```
select  dec(avg(ocena * 1.0), 4, 2)
from    ispit
where   ocena > 5;
```

```
select  char(naziv, 30), dec(avg(ocena * 1.0), 4, 2) as prosek
from    ispit a, predmet b
where   ocena > 5 and a.id_predmeta = b.id_predmeta
group by
        (naziv);
```

```
select  oznaka_roka, dec(avg(ocena * 1.0), 4, 2) as prosek
from    ispit
where   ocena > 5
group by      (oznaka_roka);
```

```
select  char(naziv, 30),
        oznaka_roka,
        dec(avg(ocena * 1.0), 4, 2) as prosek
from    ispit a, predmet b
where   ocena > 5 and a.id_predmeta = b.id_predmeta
group by      (naziv, oznaka_roka);
```

■

Problemi koji se javljaju su pravljenje više sličnih ali neznatno različitih upita je dosadno i zamorno za korisnika i izvršavanje svih upita može da bude jako skupa operacija.

Sa više nivoa agregacije u jednom upitu olakšava se posao korisniku, nudi se mogućnost da se sve agregacije izračunaju mnogo efikasnije (u jednom prolazu).

Grouping sets

Grouping sets se koristi kada treba grupisati po skupovima koji su navedeni.

■ **Primer 13.136** Naći prosečne ocene na položenim ispitima po predmetima, i posebno po ispitnim rokovima.

```
select
    char(naziv, 20),
    oznaka_roka,
    dec(avg(ocena * 1.0), 4, 2) as prosek
from
    ispit a,
    predmet b
where
    ocena > 5
    and a.id_predmeta = b.id_predmeta
group by
    grouping sets ((naziv), (oznaka_roka));
```

Rezultat se prikazuje u obliku jedne tabele (koja se vrlo teško može nazvati relacijom):

1	OZNAKA_ROKA	PROSEK
-----	-----	-----
NULL	apr	7.80
NULL	feb	6.75
NULL	jan	7.68
Analiza 1	NULL	8.00
Engleski jezik 1	NULL	6.50
Geometrija	NULL	7.50
Programiranje 1	NULL	7.83

■

ROLLUP

GROUP BY ROLLUP se koristi kada je potrebno da se analizira skup podataka preko jedne dimenzije, ali na više nivoa detalja.

■ **Primer 13.137** Naći ukupnu prosečnu ocenu na položenim ispitima, prosečnu ocenu na položenim ispitima po predmetima, i po predmetima za svaki ispitni rok posebno (nezavisno od godine polaganja).

```
select
    char(naziv, 20),
```

```

        oznaka_roka,
        dec(avg(ocena * 1.0), 4, 2) as prosek
from
        ispit a,
        predmet b
where
        ocena > 5
        and a.id_predmeta = b.id_predmeta
group by
        rollup ((naziv), (oznaka_roka));

```

Rezultat:

1	OZNAKA_ROKA	PROSEK
-----	-----	-----
NULL	NULL	7.56
Analiza 1	NULL	8.00
Engleski jezik 1	NULL	6.50
Geometrija	NULL	7.50
Programiranje 1	NULL	7.83
Analiza 1	feb	7.00
Analiza 1	jan	8.16
Engleski jezik 1	jan	6.50
Geometrija	apr	7.80
Geometrija	feb	7.00
Geometrija	jan	7.00
Programiranje 1	feb	6.50
Programiranje 1	jan	8.50

Dakle, kod rollup se postepeno uvode detalji u računicu, prvo predmet, pa onda i ispitni rok.

■

```

rollup ((naziv), (oznaka\_roka)) =
grouping sets ((), (naziv), (naziv, oznaka\_roka))

```

```

group by rollup (A, B, ....., Z) =
grupisanje preko:
()
(A)
(A, B)
.....

```

(A, B, ...)
 (A, B, ..., Z)

CUBE

GROUP BY CUBE se koristi kada je potrebno da se analizira skup podataka preko više dimenzija.

■ **Primer 13.138** Naći ukupnu prosečnu ocenu, prosečnu ocenu na položenim ispitima po predmetima, prosečnu ocenu posebno po ispitnim rokovima nezavisno od godine, i po predmetima za svaki ispitni rok posebno nezavisno od godine.

```
select
    char(naziv, 20),
    oznaka_roka,
    dec(avg(ocena * 1.0), 4, 2) as prosek
from
    ispit a,
    predmet b
where
    ocena > 5
    and a.id_predmeta = b.id_predmeta
group by
    cube ((naziv), (oznaka_roka));
```

Rezultat:

1	OZNAKA_ROKA	PROSEK
-----	-----	-----
NULL	apr	7.80
NULL	feb	6.75
NULL	jan	7.68
NULL	NULL	7.56
Analiza 1	NULL	8.00
Engleski jezik 1	NULL	6.50
Geometrija	NULL	7.50
Programiranje 1	NULL	7.83
Analiza 1	feb	7.00
Analiza 1	jan	8.16
Engleski jezik 1	jan	6.50
Geometrija	apr	7.80
Geometrija	feb	7.00
Geometrija	jan	7.00

Programiranje 1	feb	6.50
Programiranje 1	jan	8.50

■

CUBE označava da se podaci posmatraju kao da se nalaze u ćelijama višedimenzionalnog niza odnosno hiperkocke.

GROUP BY CUBE(A, B, ..., Z) znači grupisanje po svakom mogućem podskupu skupa A, B, ..., Z (ovim upitom se dobijaju rezultati sva 4 početna upita).

GROUPING

Agregatna funkcija GROUPING se koristi zajedno sa GROUPING SETS, ROLLUP ili CUBE i vraća vrednost koja je indikator da li slog vraćen sa group by klauzulom sadrži ili ne sadrži atribut koji je naveden kao argument funkcije.

■ **Primer 13.139** Dodati kolone Sadrži naziv i Sadrži oznaku roka koje će biti indikatori da li se grupisanje izvršilo po zadatom atributu.

```
select
    char(naziv, 20) as "Naziv",
    oznaka_roka as "Oznaka roka",
    dec(avg(ocena * 1.0), 4, 2) as "Prosek",
    grouping(naziv) as "Sadrzi naziv",
    grouping(oznaka_roka) as "Sadrzi oznaku roka"
from
    ispit a,
    predmet b
where
    ocena > 5
    and a.id_predmeta = b.id_predmeta
group by
    cube ((naziv), (oznaka_roka))
order by
    1;
```

Rezultat:

Naziv	Oznaka roka	Prosek	Sadrzi naziv	Sadrzi oznaku roka
-----	-----	-----	-----	-----
Analiza 1	NULL	8.00	0	1
Analiza 1	feb	7.00	0	0
Analiza 1	jan	8.16	0	0
Engleski jezik 1	NULL	6.50	0	1

Engleski jezik 1	jan	6.50	0	0
Geometrija	NULL	7.50	0	1
Geometrija	apr	7.80	0	0
Geometrija	feb	7.00	0	0
Geometrija	jan	7.00	0	0
Programiranje 1	NULL	7.83	0	1
Programiranje 1	feb	6.50	0	0
Programiranje 1	jan	8.50	0	0
NULL	apr	7.80	1	0
NULL	feb	6.75	1	0
NULL	jan	7.68	1	0
NULL	NULL	7.56	1	1

■

■ **Primer 13.140** Razlikovanje NULL-ova u prethodnoj tabeli postiže se upitom

```
select
    case
        grouping (naziv)
        when 1 then '??'
        else char(naziv, 20)
    end as naziv,
    case
        grouping (oznaka_roka)
        when 1 then '!!'
        else oznaka_roka
    end as oznaka_roka,
    dec(avg(ocena * 1.0), 4, 2) as prosek
from
    ispit a,
    predmet b
where
    ocena > 5
    and a.id_predmeta = b.id_predmeta
group by
    grouping sets ((naziv), (oznaka_roka));
```

Rezultat:

NAZIV	OZNAKA_ROKA	PROSEK
-----	-----	-----
??	apr	7.80

??	feb	6.75
??	jan	7.68
Analiza 1	!!	8.00
Engleski jezik 1	!!	6.50
Geometrija	!!	7.50
Programiranje 1	!!	7.83

■

13.7.3 Analitičke funkcije

Partition klauza

Pomoću PARTITION BY definiše se particija u okviru koje se primenjuje OLAP operacija. Ukoliko nije definisana order ni window frame klauza podrazumevani prozor podataka je ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.

Order klauza

Pomoću ORDER BY definiše se uređenje u okviru particije koje se koristi da se odrede vrednosti OLAP operacije. Ne definiše uređenje konačne tabele. Uređenja mogu biti ASC, DESC, NULLS FIRST i NULLS LAST. Ukoliko nije navedena window frame klauza podrazumevana je RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

Window frame klauza

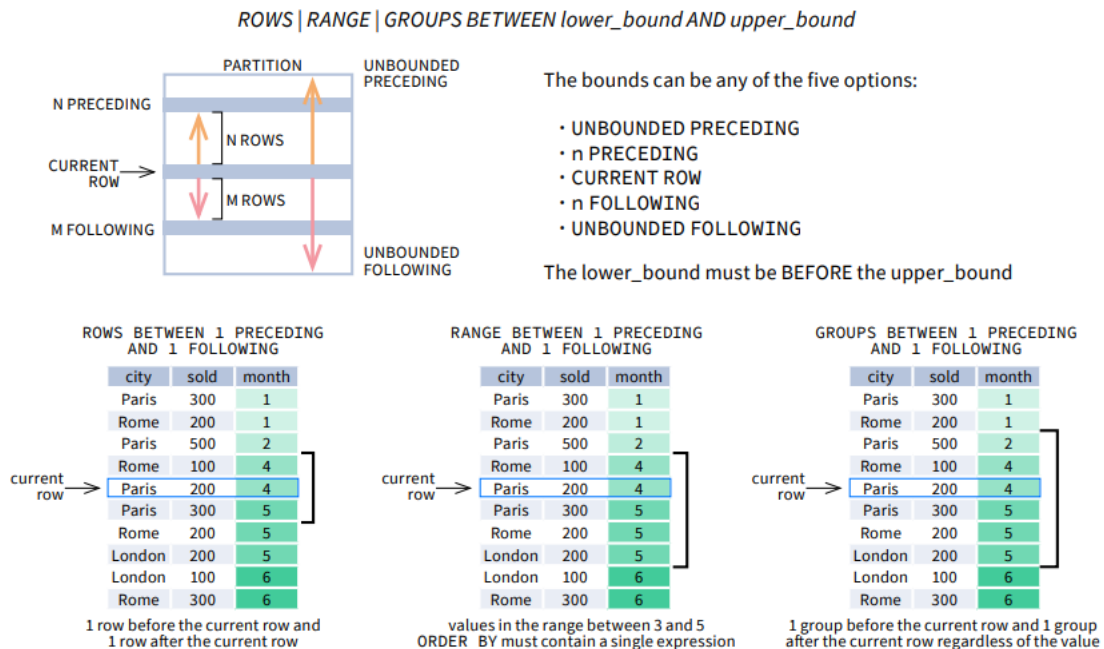
Pomoću ove klauze definiše se prozor redova za koje se vrše analitičke funkcije, kao na slici 13.1.

OLAP AVG

■ **Primer 13.141** Za svakog studenta, izračunati prosečan prosek ocena u grupi studenata koji imaju broj indeksa za jedan veći i jedan manji od tog studenta.

Za dodatni test, u tabelu ispita i dosijea uneti nove slogove tako da postoje po tri kontinuirana indeksa

```
with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
```

Slika 13.1: Window frame klauza

```

dosije b
where
ocena > 5
and a.indeks = b.indeks
and a.datum_ispita =(
    select
        max(datum_ispita)
    from
        ispit c
    where
        a.indeks = c.indeks
        and a.id_predmeta = c.id_predmeta
)
group by
    a.indeks,
    ime,
    prezime
)
select
    indeks,
    ime,
    prezime,
    prosek,

```

```

        dec(
            avg(prosek) over (
                order by
                    indeks range BETWEEN 1 PRECEDING
                    AND 1 FOLLOWING
            ),
            4,
            2
        ) as "Prosek grupe"
from
    proseci;

```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	Prosek grupe
20130023	Sanja	Terzic	8.66	8.16
20130024	Nikola	Vukovic	7.66	8.16
20130027	Milena	Stankovic	8.00	8.00
20140021	Milos	Peric	8.25	8.12
20140022	Marijana	Savkovic	8.00	8.12
20140025	Marijana	Savkovic	6.00	6.33
20140026	Zorica	Miladinovic	6.66	6.33

■ **Primer 13.142** Za svakog studenta, izračunati prosečan prosek ocena u grupi studenata čiji je broj indeksa manji za 1, 2 ili 3 u odnosu na broj njegovog indeksa, kao i prosečan prosek ocena u grupi studenata uključujući i tog studenta i studente sa brojevima indeksa koji su za 1 ili 2 manji od njegovog.

```

with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks

```

```

        and a.datum_ispita =(
            select
                max(datum_ispita)
            from
                ispit c
            where
                a.indeks = c.indeks
                and a.id_predmeta = c.id_predmeta
        )
    group by
        a.indeks,
        ime,
        prezime
)
select
    indeks,
    ime,
    prezime,
    prosek,
    dec(
        avg(prosek) over (
            order by
                indeks range BETWEEN 3 PRECEDING
                AND 1 PRECEDING
        ),
        4,
        2
    ) as "Prosek - br. indeksa manji za 3",
    dec(
        avg(prosek) over (
            order by
                indeks range BETWEEN 2 PRECEDING
                AND CURRENT ROW
        ),
        4,
        2
    ) as "Prosek - uključen student"
from
    proseci;

```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	Prosek 3	Prosek uklj.
--------	-----	---------	--------	----------	--------------

20130023	Sanja	Terzic	8.66	NULL	8.66
20130024	Nikola	Vukovic	7.66	8.66	8.16
20130027	Milena	Stankovic	8.00	7.66	8.00
20140021	Milos	Peric	8.25	NULL	8.25
20140022	Marijana	Savkovic	8.00	8.25	8.12
20140025	Marijana	Savkovic	6.00	8.00	6.00
20140026	Zorica	Miladinovic	6.66	6.00	6.33

■

FIRST_VALUE, LAST_VALUE

■ **Primer 13.143** Prikazati razlike, po studentima i predmetima, između ocene dobijene na ispitu u odnosu na prvu i poslednju ocenu (u odnosu na datum polaganja) dobijenu na tom ispitu.

```

select
    indeks,
    id_predmeta,
    ocena as "Ocena studenta",
    first_value(ocena) over (
        partition by id_predmeta
        order by
            datum_ispita asc
    ) as "Prva ocena",
    last_value(ocena) over (
        partition by id_predmeta
        order by
            datum_ispita asc
    ) as "Poslednja ocena",
    first_value(ocena) over (
        partition by id_predmeta
        order by
            datum_ispita asc
    ) - ocena as "Razlika do prve ocene",
    last_value(ocena) over (
        partition by id_predmeta
        order by
            datum_ispita asc
    ) - ocena as "Razlika do poslednje ocene"
from
    ispit;

```

Rezultat:

INDEKS	ID	OCENA	DATUM	Prva	Poslednja	Razlika do prve	Razlika do poslednje
20140021	2001	10	2015-01-25	10	5	0	-5
20140022	2001	9	2015-01-25	10	5	1	-4
20130023	2001	8	2015-01-25	10	5	2	-3
20130024	2001	7	2015-01-25	10	5	3	-2
20140025	2001	5	2015-01-25	10	5	5	0
20140025	2001	6	2015-02-10	10	7	4	1
20140026	2001	7	2015-02-10	10	7	3	0
20140026	2001	-7	NULL	10	-7	17	0
...							

Primetiti da se prva i poslednja ocena računaju u okviru istog datuma ispita. ■

LAG, LEAD

U odnosu na zadati redosled, LAG vraća prethodnu vrednost, dok LEAD vraća narednu vrednost.

■ **Primer 13.144** Prikazati proseke ocena studenata i razliku do proseka prvog narednog i prethodnog studenta (sortiranog po proseku ocena)

```
with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks
        and a.datum_ispita =(
            select
                max(datum_ispita)
            from
                ispit c
            where
                a.indeks = c.indeks
```

```

                                and a.id_predmeta = c.id_predmeta
                                )
                                group by
                                a.indeks,
                                ime,
                                prezime
                                )
select
    indeks,
    ime,
    prezime,
    prosek as "Prosek",
    lead(prosek, 1) over(
        order by
            prosek desc
    ) as "Prvi naredni prosek",
    lag (prosek, 1) over(
        order by
            prosek desc
    ) as "Prvi prethodni prosek",
    lead(prosek, 1) over(
        order by
            prosek desc
    ) - prosek as "Razlika do narednog proseka",
    lag (prosek, 1) over(
        order by
            prosek desc
    ) - prosek as "Razlika do prethodnog proseka"
from
    proseci
order by
    prosek desc;

```

Rezultat:

INDEKS	IME	PREZIME	Prosek	Prvi nar.	Prvi pret.	Razlika do nar.	Razlika do pret.
20130023	Sanja	Terzic	8.66	8.25	NULL	-0.41	NULL
20140021	Milos	Peric	8.25	8.00	8.66	-0.25	0.41
20130027	Milena	Stankovic	8.00	8.00	8.25	0.00	0.25
20140022	Marijana	Savkovic	8.00	7.66	8.00	-0.34	0.00
20130024	Nikola	Vukovic	7.66	6.66	8.00	-1.00	0.34

20140026	Zorica	Miladinovic	6.66	6.00	7.66	-0.66	1.00
20140025	Marijana	Savkovic	6.00	NULL	6.66	NULL	0.66

■

RANK

Pomoću funkcije rank određuje se rank u odnosu za zadati redosled. Ukoliko neki redovi imaju isti rank, tada se njima dodeljuje izračunati rank, ali se rank sledećeg izračunava tako da se uračunaju duplirane vrednosti.

■ **Primer 13.145** Odrediti redosled studenta u zavisnosti od proseka njihovih ocena.

```
with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks
        and a.datum_ispita = (
            select
                max(datum_ispita)
            from
                ispit c
            where
                a.indeks = c.indeks
                and a.id_predmeta = c.id_predmeta
        )
    group by
        a.indeks,
        ime,
        prezime
)
select
    indeks,
    ime,
    prezime,
    prosek,
```

```

        rank() over(
            order by
                prosek desc
        ) as redosled
from
    proseci
order by
    indeks;

```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	REDOSLED
20130023	Sanja	Terzic	8.66	1
20130024	Nikola	Vukovic	7.66	5
20130027	Milena	Stankovic	8.00	4
20140021	Milos	Peric	8.25	3
20140022	Marijana	Savkovic	8.50	2
20140025	Marijana	Savkovic	6.00	7
20140026	Zorica	Miladinovic	6.66	6

```

insert into
    ispit(
        indeks,
        id_predmeta,
        godina_roka,
        oznaka_roka,
        ocena,
        datum_ispita,
        bodovi
    )
values
    (20140022, 3001, 2015, 'jan', 7, '28.01.2015', 72);

```

Kada se ponovo izvrši prethodni upit dobija se:

Rezultat:

INDEKS	IME	PREZIME	PROSEK	REDOSLED
20130023	Sanja	Terzic	8.66	1
20130024	Nikola	Vukovic	7.66	5
20130027	Milena	Stankovic	8.00	3

20140021	Milos	Peric	8.25	2
20140022	Marijana	Savkovic	8.00	3
20140025	Marijana	Savkovic	6.00	7
20140026	Zorica	Miladinovic	6.66	6

■

DENSE_RANK

Pomoću funkcije `dense_rank` određuje se rang u odnosu za zadati redosled, s tim da ukoliko neki redovi imaju isti rang, tada se njima dodeljuje izračunati rang, ali se rang sledećeg izračunava tako da se ne uračunaju duplirane vrednosti, odnosno neće biti rupa u rangovima.

■ **Primer 13.146** Odrediti redosled studenta u zavisnosti od trenutnog proseka njihovih ocena.

```
with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks
        and a.datum_ispita = (
            select
                max(datum_ispita)
            from
                ispit c
            where
                a.indeks = c.indeks
                and a.id_predmeta = c.id_predmeta
        )
    group by
        a.indeks,
        ime,
        prezime
)
select
```

```

        indeks,
        ime,
        prezime,
        prosek,
        dense_rank() over(
            order by
                prosek desc
        ) as "Redosled"
from
    proseci --order by indeks;
    --order by dense_rank() over(order by prosek desc)

```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	Redosled
20130023	Sanja	Terzic	8.66	1
20140021	Milos	Peric	8.25	2
20130027	Milena	Stankovic	8.00	3
20140022	Marijana	Savkovic	8.00	3
20130024	Nikola	Vukovic	7.66	4
20140026	Zorica	Miladinovic	6.66	5
20140025	Marijana	Savkovic	6.00	6

■ **Primer 13.147** Rangirati studente prema mestu rođenja i trenutnom proseku ocena. U svakom mestu rangiranje studenata započeti od pozicije 1 za studenta sa najvećim prosekom.

```

with proseci(indeks, ime, prezime, mesto_rodjenja, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        mesto_rodjenja,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks
        and a.datum_ispita =(

```

```

        select
            max(datum_ispita)
        from
            ispit c
        where
            a.indeks = c.indeks
            and a.id_predmeta = c.id_predmeta
    )
    group by
        a.indeks,
        ime,
        prezime,
        mesto_rodjenja
)
select
    indeks,
    ime,
    prezime,
    mesto_rodjenja,
    prosek,
    DENSE_RANK() OVER (
        PARTITION BY mesto_rodjenja
        ORDER BY
            prosek DESC
    ) AS "Redosled"
from
    proseci;

```

Rezultat:

INDEKS	IME	PREZIME	MESTO_RODJENJA	PROSEK	Redosled
20130023	Sanja	Terzic	Beograd	8.66	1
20140021	Milos	Peric	Beograd	8.25	2
20140022	Marijana	Savkovic	Kraljevo	8.00	1
20140025	Marijana	Savkovic	Kraljevo	6.00	2
20140026	Zorica	Miladinovic	Vranje	6.66	1
20130027	Milena	Stankovic	NULL	8.00	1
20130024	Nikola	Vukovic	NULL	7.66	2

ROW_NUMBER

Određuje se redni broj reda u uređenju koje je definisano, počevši od 1.

■ **Primer 13.148** Prikazati brojeve redova onim redom kojim se generišu u rezultatu. U prikazu redovi mogu da budu promenjeni u zavisnosti od atributa za koga se zahteva uređenje.

```
with proseci(indeks, ime, prezime, prosek) as (  
    select  
        a.indeks,  
        ime,  
        prezime,  
        dec(avg(ocena * 1.0), 4, 2)  
    from  
        ispit a,  
        dosije b  
    where  
        ocena > 5  
        and a.indeks = b.indeks  
        and a.datum_ispita =(  
            select  
                max(datum_ispita)  
            from  
                ispit c  
            where  
                a.indeks = c.indeks  
                and a.id_predmeta = c.id_predmeta  
        )  
    group by  
        a.indeks,  
        ime,  
        prezime  
)  
select  
    indeks,  
    ime,  
    prezime,  
    prosek,  
    row_number() over(  
        order by  
            prosek desc  
    ) as "Redosled"  
from
```

```
proseci;
```

```
-- order by indeks;
```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	Redosled
20130023	Sanja	Terzic	8.66	1
20140021	Milos	Peric	8.25	2
20130027	Milena	Stankovic	8.00	3
20140022	Marijana	Savkovic	8.00	4
20130024	Nikola	Vukovic	7.66	5
20140026	Zorica	Miladinovic	6.66	6
20140025	Marijana	Savkovic	6.00	7

Za poređenje između row_number, rank i dense_rank izvršiti sledeći upit:

```
with proseci(indeks, ime, prezime, prosek) as (
    select
        a.indeks,
        ime,
        prezime,
        dec(avg(ocena * 1.0), 4, 2)
    from
        ispit a,
        dosije b
    where
        ocena > 5
        and a.indeks = b.indeks
        and a.datum_ispita =(
            select
                max(datum_ispita)
            from
                ispit c
            where
                a.indeks = c.indeks
                and a.id_predmeta = c.id_predmeta
        )
    group by
        a.indeks,
        ime,
        prezime
```

```

)
select
    indeks,
    ime,
    prezime,
    prosek,
    row_number() over(
        order by
            prosek desc
    ) as "Row number",
    rank() over(
        order by
            prosek desc
    ) as "Rank",
    dense_rank() over(
        order by
            prosek desc
    ) as "Dense rank"
from
    proseci

```

Rezultat:

INDEKS	IME	PREZIME	PROSEK	Row number	Rank	Danse rank
20130023	Sanja	Terzic	8.66	1	1	1
20140021	Milos	Peric	8.25	2	2	2
20130027	Milena	Stankovic	8.00	3	3	3
20140022	Marijana	Savkovic	8.00	4	3	3
20130024	Nikola	Vukovic	7.66	5	5	4
20140026	Zorica	Miladinovic	6.66	6	6	5
20140025	Marijana	Savkovic	6.00	7	7	6

■ **Primer 13.149** Izbrisati dvostruke redove iz tabele.

```

select
    *
from
    dosije_sa_duplikatima;

delete from
    (

```

```

select
    rownumber() over (partition by indeks) as rn
from
    dosije_sa_duplikatima
) as a
where
    rn > 1;

select
    *
from
    dosije_sa_duplikatima;

```

■

13.7.4 Rekurzivni SQL

Rekurzivni izraz je izraz koji sadrži referencu na samu sebe iz FROM klauze. Izraz se može sastojati od select-a ili select-a kombinovanim sa skupovnim operacijama, tzv. fullselect.

Rekurzivni izrazi imaju sledeća pravila:

- Prvi fullselect prve unije ne sme da sadrži referencu na pomoćnu tabelu (tabelu definisanu sa WITH)
- Za svaki fullselect koji je deo rekurzivnog ciklusa mora da važi sledeće:
 - počinje sa SELECT. SELECT DISTINCT nije dozvoljen.
 - sadrži samo jednu referencu na pomoćnu tabelu koja je deo rekurzivnog ciklusa u svojoj FROM klauzi
 - ne sadrži agregatne funkcije, GROUP BY ili HAVING klauze
- Imena kolona moraju da se navedu nakon imena pomoćne tabele
- Tip podataka i dužina svake kolone iz pomoćne tabele moraju da odgovaraju tipu i dužini svih odgovarajućih kolona iterativnog fullselect-a
- Ukoliko se koristi unija navesti UNION ALL umesto UNION
- Ne može se koristiti INTERSECT ili EXCEPT
- Spoljna spajanja ne mogu da budu deo rekurzivnog ciklusa
- Podupiti ne mogu da budu deo rekurzivnog ciklusa

Rekurzivne upite treba pisati vrlo pažljivo da se ne bi napravila beskonačna petlja.

■ **Primer 13.150** Napisati rekurzivan SQL upit koji izdvaja predmete koji su direktan preduslov za polaganje ispita iz predmeta 'Relacione baze podataka', kao i sve njihove direktne preduslove, rekurzivno.

Rešenje prikazuje zavisnosti tipa 'predmet A' → 'predmet B'. Pri tome nije obezbeđeno da 'predmet A' uvek bude željeni predmet, već se prikazuju svi predmeti u hijerarhiji za koje

postoji zavisnost a do kojih se dolazi polazeći od željenog predmeta. Proizvodi se izveštaj tipa:

Predmet	Uslovni predmet
-----	-----
Relacione baze podataka	Diskretne strukture 1
Relacione baze podataka	Diskretne strukture 2
Relacione baze podataka	Programiranje 1
Relacione baze podataka	Programiranje 2
Programiranje 2	Programiranje 1

SQL koji formira ovaj izveštaj je:

```
with preduslovi (predmet, uslovni_predmet) as (
    -- Inicijalni SELECT
    select
        koren.idpredmeta,
        koren.iduslovnogpredmeta
    from
        da.uslovnipredmet koren
    where
        koren.idpredmeta in (
            select
                id
            from
                da.predmet
            where
                naziv = 'Relacione baze podataka'
        )
    union
    all -- Iterativni SELECT
    select
        dete.idpredmeta,
        dete.iduslovnogpredmeta
    from
        preduslovi roditelj,
        da.uslovnipredmet dete
    where
        roditelj.uslovni_predmet = dete.idpredmeta
) -- Glavni SELECT
select
    distinct (
        select
```



```

                                char(naziv, 25)
                                from
                                da.predmet b
                                where
                                b.id = a.predmet
        ) as "Predmet",
        (
            select
                char(naziv, 25)
            from
                da.predmet b
            where
                b.id = a.uslovni_predmet
        ) as "Uslovni predmet"
from
    preduslovi a
order by
    1 desc;

```

■

■ **Primer 13.151** Napisati rekurzivan SQL upit koji izdvaja predmete koji su preduslov za polaganje ispita iz predmeta 'Relacione baze podataka', uključujući i one koji nisu direktan preduzlov.

Izveštaj koji se dobija ima sledeći oblik:

Predmet	Uslovni predmet
Relacione baze podataka	Diskretne strukture 1
Relacione baze podataka	Diskretne strukture 2
Relacione baze podataka	Programiranje 1
Relacione baze podataka	Programiranje 2

SQL koji formira ovaj izveštaj je:

```

with preduslovi (predmet, uslovni_predmet) as (
    select
        koren.idpredmeta,
        koren.iduslovnogpredmeta
    from
        da.uslovnipredmet koren
    where
        koren.idpredmeta in (

```

```

        select
            id
        from
            da.predmet
        where
            naziv = 'Relacione baze podataka'
    )
union
all
select
    roditelj.predmet,
    dete.iduslovnogpredmeta
from
    preduslovi roditelj,
    da.uslovnipredmet dete
where
    roditelj.uslovni_predmet = dete.idpredmeta
)
select
    distinct (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.predmet
    ) as "Predmet",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.uslovni_predmet
    ) as "Uslovni predmet"
from
    preduslovi a
order by
    1 desc;

```

■

■ **Primer 13.152** Napisati rekurzivan SQL upit koji izdvaja predmete koji su preduslov za polaganje ispita iz predmeta 'Relacione baze podataka', uključujući i one koji nisu direktan

preduzlov. Dodati i kolonu koja predstavlja naziv predmeta preko kog je predmet preduslov, ili null ukoliko je direktan preduslov. Dodati i polje sa nivoom rekurzije.

Uvođenje nivoa i predmeta preko koga je neki predmet preduslov (ako postoji tranzivnost).

Nivo	Predmet	Predmet preko koga	Uslovni predmet
0	Relacione baze podataka	NULL	Diskretne strukture 1
0	Relacione baze podataka	NULL	Diskretne strukture 2
0	Relacione baze podataka	NULL	Programiranje 1
0	Relacione baze podataka	NULL	Programiranje 2
1	Relacione baze podataka	Programiranje 2	Programiranje 1

SQL koji formira ovaj izveštaj je:

```
with preduslovi (nivo, predmet, preko, uslovni_predmet) as (
    -- Inicijalni SELECT
    select
        0,
        idpredmeta,
        null,
        iduslovnogpredmeta
    from
        da.uslovnipredmet koren
    where
        koren.idpredmeta in (
            select
                id
            from
                da.predmet
            where
                naziv = 'Relacione baze podataka'
        )
    union
    all -- Iterativni SELECT
    select
        roditelj.nivo + 1,
        roditelj.predmet,
        dete.idpredmeta,
        dete.iduslovnogpredmeta
    from
        preduslovi roditelj,
```

```

        da.uslovnipredmet dete
    where
        roditelj.uslovni_predmet = dete.idpredmeta
) -- Glavni SELECT
select
    distinct nivo as "Nivo",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.predmet
    ) as "Predmet",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.preko
    ) as "Predmet preko koga se uslovni predmet slusa",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.uslovni_predmet
    ) as "Uslovni predmet"
from
    preduslovi a
order by
    1;

```

■ **Primer 13.153** Napisati rekurzivan SQL upit koji izdvaja studijski program, nazive, semestar u kome se slušaju i broj bodova svih predmeta koji su preduslov za polaganje ispita iz predmeta 'Relacione baze podataka'. Navesti nivo i predmet preko koga je predmet implicitno zavisian.

Rezultat bi izgledao ovako:

Studijski program

Nivo Predmet

Astronomija i astrofizika-A	0	Relacione baze podataka
Astronomija i astrofizika-A	1	Relacione baze podataka
Astronomija i astrofizika-A	0	Relacione baze podataka
Astronomija i astrofizika-A	0	Relacione baze podataka
Astronomija i astrofizika-A	0	Relacione baze podataka
Informatika-I	0	Relacione baze podataka
Informatika-I	1	Relacione baze podataka
Informatika-I	0	Relacione baze podataka
Informatika-I	0	Relacione baze podataka
Informatika-I	0	Relacione baze podataka

Predmet preko koga je preduslov	Uslovni predmet	ESPB	Semestar
NULL	Diskretne strukture 1	6	1
Programiranje 2	Programiranje 1	8	1
NULL	Programiranje 1	8	1
NULL	Diskretne strukture 2	6	2
NULL	Programiranje 2	8	2
NULL	Diskretne strukture 1	6	1
Programiranje 2	Programiranje 1	8	1
NULL	Programiranje 1	8	1
NULL	Diskretne strukture 2	6	2
NULL	Programiranje 2	8	2

SQL koji formira ovaj izveštaj je:

```

with preduslovi (
    nivo,
    id_programa,
    predmet,
    preko,
    uslovni_predmet
) as (
    -- Inicijalni SELECT
    select
        0,
        idprograma,
        idpredmeta,
        null,
        iduslovnogpredmeta
    from

```

```

        da.uslovnipredmet koren
where
    koren.idpredmeta in (
        select
            id
        from
            da.predmet
        where
            naziv = 'Relacione baze podataka'
    )
union
all --      Iterativni SELECT
select
    roditelj.nivo + 1,
    roditelj.id_programa,
    roditelj.predmet,
    dete.idpredmeta,
    dete.iduslovnogpredmeta
from
    preduslovi roditelj,
    da.uslovnipredmet dete
where
    roditelj.uslovni_predmet = dete.idpredmeta
    and roditelj.id_programa = dete.idprograma
    and nivo <= 10 -- uklanja poruku da je moguća beskonačna petlja
) -- Glavni SELECT
select
    distinct (
        select
            char(naziv || '-' || oznaka, 30)
        from
            da.studijskiprogram b
        where
            b.id = a.id_programa
    ) as "Studijski program",
    substr(char(nivo), 1, 4) as "Nivo",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.predmet
    )

```

```

    ) as "Predmet",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.preko
    ) as "Predmet preko koga je preduslov",
    (
        select
            char(naziv, 25)
        from
            da.predmet b
        where
            b.id = a.uslovni_predmet
    ) as "Uslovni predmet",
    (
        select
            ESPB
        from
            da.predmet b
        where
            b.id = a.uslovni_predmet
    ) as "ESPB",
    (
        select
            semestar
        from
            da.predmetprograma b
        where
            b.idpredmeta = a.uslovni_predmet
            and b.idprograma = a.id_programa
    ) as "Semestar"
from
    preduslovi a
order by
    1;

```

■

■ **Primer 13.154** Kreirati tabelu o delovima i poddelovima.

```
create table listadelova (
```

```

    deo varchar(8) not null,
    poddeo varchar(8) not null,
    kolicina integer,
    primary key (deo, poddeo)
);

```

Popuniti tabelu podacima iz fajla `delovi.load`. Ispis komandi upisati u fajl `delovi.load.izv`.

```

load
from
    "delovi.load" of del modified by fastparse
    method p (1, 2, 3) messages "delovi.load.izv"
insert into
    listadelova (deo, poddeo, kolicina)
    nonrecoverable indexing mode autoselect;

```

Fajl `delovi.load`:

```

00,01,5
00,05,3
01,02,2
01,03,3
01,04,4
01,06,3
02,05,7
02,06,6
03,07,6
04,08,10
04,09,11
05,10,10
05,11,10
06,12,10
06,13,10
07,14,8
07,12,8

```

■

■ **Primer 13.155** Prikazati spisak svih delova i količine neposrednih sastavnih poddelova od kojih se ti delovi sastoje.

Rešenje:

```

with rekurzlistadelova (deo, poddeo, kolicina) as (
    --    inicijalni select

```



```

select
    koren.deo,
    koren.poddeo,
    koren.kolicina
from
    listadelova koren
where
    koren.deo = '01'
union
all --    iterativni select
select
    dete.deo,
    dete.poddeo,
    dete.kolicina
from
    rekurzlistadelova roditelj,
    listadelova dete
where
    roditelj.poddeo = dete.deo
) -- glavni select
select
    distinct deo,
    poddeo,
    sum(kolicina) as ukupno
from
    rekurzlistadelova
group by
    deo,
    poddeo
order by
    deo,
    poddeo;

```

Rezultat:

DEO	PODDEO	UKUPNO
01	02	2
01	03	3
01	04	4

SQL0347W The recursive common table expression "DB2INST1.REKURZLISTADELOVA" may contain an infinite loop. SQLSTATE=01605

01	06	3
02	05	7
02	06	6
03	07	6
04	08	10
04	09	11
05	10	10
05	11	10
06	12	20
06	13	20
07	12	8
07	14	8

■ **Primer 13.156** Prikazati spisak svih delova sa količinama koji su ugrađeni u proizvod (deo) sa šifrom 01.

Rešenje:

```
with rekurzlistadelova (deo, poddeo, kolicina) as (  
    select  
        koren.deo,  
        koren.poddeo,  
        koren.kolicina  
    from  
        listadelova koren  
    where  
        koren.deo = '01'  
    union  
    all  
    select  
        roditelj.deo,  
        dete.poddeo,  
        roditelj.kolicina * dete.kolicina  
    from  
        rekurzlistadelova roditelj,  
        listadelova dete  
    where  
        roditelj.poddeo = dete.deo  
)  
select  
    deo,  
    poddeo,  
    sum(kolicina) as "ukupna upotrebljena kolcina"
```

```

from
    rekurzlistadelova
group by
    deo,
    poddeo
order by
    deo,
    poddeo;

```

Rezultat:

DEO	PODDEO	Ukupna upotrebljena kolcina

SQL0347W The recursive common table expression "DB2INST1.REKURZLISTADELOVA" may contain an infinite loop. SQLSTATE=01605		
01	02	2
01	03	3
01	04	4
01	05	14
01	06	15
01	07	18
01	08	40
01	09	44
01	10	140
01	11	140
01	12	294
01	13	150
01	14	144

■ **Primer 13.157** Prikazati spisak delova koji se koriste pri pravljenju proizvoda sa šifrom 01. Spisak ograničiti na prva dva nivoa.

Rešenje:

```

with rekurzlistadelova (nivo, deo, poddeo, kolicina) as (
    select
        1,
        koren.deo,
        koren.poddeo,
        koren.kolicina
    from
        listadelova koren

```

```

        where
            koren.deo = '01'
        union
        all
        select
            roditelj.nivo + 1,
            dete.deo,
            dete.poddeo,
            dete.kolicina
        from
            rekurzlistadelova roditelj,
            listadelova dete
        where
            roditelj.poddeo = dete.deo
            and roditelj.nivo < 2
    )
select
    deo,
    nivo,
    poddeo,
    kolicina
from
    rekurzlistadelova;

```

Rezultat:

DEO	NIVO	PODDEO	KOLICINA
01		1 02	2
01		1 03	3
01		1 04	4
01		1 06	3
02		2 05	7
02		2 06	6
03		2 07	6
04		2 08	10
04		2 09	11
06		2 12	10
06		2 13	10

13.7.5 LOB

Termin LOB (eng. large object) se odnosi na tipove CLOB, DBLOB i BLOB:

- CLOB (eng. character large object) je tip stringa promenljive dužine do 2,147,483,647 bajtova (2 GB minus 1 bajt). Napravljen je za skladištenje velikih dokumenata, na primer, u kodiranju utf-8.
- DBLOB (eng. double-byte character large object) je graphic tip promenljive dužine do 1,073,741,823 karaktera od po dva bajta. Napravljen je za skladištenje velikih dokumenata, na primer, u kodiranju utf-16.
- BLOB (eng. binary large object) je tip promenljive dužine do 2,147,483,647 bajtova (2 GB minus 1 bajt). Napravljen je za skladištenje fajlova poput slika, audio ili multimedijalnog materijala.

■ **Primer 13.158** Kreirati tabelu `velika_tabela` za smeštanje LOB objekata. Popuniti tabelu podacima.

Rešenje:

```
drop table velika_tabela
create table velika_tabela (
    redni_broj integer      not null,
    tip_teksta character(20) not null,
    tekst      clob(3M)      ,
    slika      blob(3M)      ,
    xmltekst   xml          ,
    primary key(redni_broj)
);

load from punjenje.load of del lobs from . modified by fastparse
lobsinfile method p (1, 2, 3, 4, 5) messages punjenje.msg
replace into velika_tabela (redni_broj, tip_teksta, tekst, slika, xmltekst)
nonrecoverable indexing mode autoselect;
```

Sadržaj fajla `punjenje.load` je:

```
1,tip1,rekurzivnisql.sql,1.Uvod.pdf,ulaz.slog1.xml
2,tip2,rbp.html,2.arhitektura.pdf,ulaz.slog2.xml
3,duze od 32K,1RU3.pdb,1ru3_bio_r_500.jpg,ulaz.slog3.xml
4,datoteka od 2M,1GXH.pdb,1gxh_asym_r_500.jpg,ulaz.slog4.xml
```

Proverimo veličine unetih fajlova:

```
select
    redni_broj,
    tip_teksta,
```

```

length(tekst) Tekst,
length(slika) Slika,
length(xmlserialize(xmltekst as clob(20m))) XML
from
    velika_tabela;

```

Rezultat:

REDNI_BROJ	TIP TEKSTA	TEKST	SLIKA	XML
1	tip1	2275	110630	101
2	tip2	5626	682247	28
3	duze od 32K	557280	49506	108
4	datoteka od 2M	2162538	36756	849

■

13.7.6 Materijalizovani upiti

Materijalizovani upiti (eng. materialized query table) su definisani rezultatom upita u cilju poboljšanja performansi. Ove tabele bi trebalo da se definišu u prostoru tabela koji je označen sa NOT LOGGED da bi se izbegao gubitak performansi usled logovanja izmena nad podacima.

Da bi se definisala MQT potrebno je:

- Napisati CREATE TABLE iskaz na osnovu fullselect-a.
- Navesti kako se popunjava podacima
 - DATA INITIALLY DEFERRED - Db2 ne popunjava MQT prilikom kreiranja. Da bi se popunila tabela potrebno je eksplicitno navesti:
 - * REFRESH TABLE - za MQT koje održava sistem
 - * LOAD, INSERT ili REFRESH TABLE - za MQT koje održava korisnik
 - REFRESH DEFERRED - Db2 ne osvežava odmah podatke u MQT kada su promenjeni podaci u baznim tabelama. Može da se koristi REFRESH TABLE za osvežavanje podataka tako da odgovaraju podacima u tabelama nad kojima su definisani.
- Navesti ko održava MQT
 - MAINTAINED BY SYSTEM - održava je sistem. Ne može se menjati od strane korisnika pomoću LOAD, ili izkazima INSERT, UPDATE, MERGE, TRUNCATE, DELETE. Može se ažurirati samo pomoću REFRESH TABLE iskaza. Ako se ne navede, BY SYSTEM je podrazumevano ponašanje.
 - MAINTAINED BY USER - održava je korisnik. Može se menjati od strane korisnika pomoću LOAD, INSERT, UPDATE, MERGE, TRUNCATE, DELETE ili REFRESH TABLE.
- Označiti da li se omogućava optimizacija

- ENABLE QUERY OPTIMIZATION - ukoliko se uključi ova opcija, Db2 je selektivniji šta može da se uključi u fullselect izkazu, zbog mogućnosti korišćenja za automatsko prepisivanje upita. Kada se tek pravi MQT koji održava korisnik na početku isključiti ovu opciju za optimizaciju upita, da se ne bi prepisali upiti na osnovu prazne MQT, jer je na početku MQT prazna.
- DISABLE QUERY OPTIMIZATION - označiti da Db2 ne može koristiti ovu tabelu za automatsko prepisivanje upita.

■ **Primer 13.159** Napraviti MQT koji za svaki predmet sadrži koliko je u kojim rokovima prosek ocena, koliko je studenata palo, koliko je položilo i broj takvih polaganja.

```
drop table prosekocenapoispitnimrokovima;

create table prosekocenapoispitnimrokovima(
    predmet,
    godina_roka,
    oznaka_roka,
    prosek,
    položio_pao,
    broj
) as (
    with položio(predmet, godina_roka, oznaka_roka, prosek, broj) as (
        select
            b.naziv,
            a.godina_roka,
            a.oznaka_roka,
            dec(avg(ocena * 1.0), 4, 2),
            count(*)
        from
            ispit a,
            predmet b
        where
            a.id_predmeta = b.id_predmeta
            and a.ocena > 5
        group by
            b.naziv,
            a.godina_roka,
            a.oznaka_roka
    ),
    pao(predmet, godina_roka, oznaka_roka, prosek, broj) as (
        select
            b.naziv,
            a.godina_roka,
            a.oznaka_roka,
```

```

        dec(avg(ocena * 1.0), 4, 2),
        count(*)
    from
        ispit a,
        predmet b
    where
        a.id_predmeta = b.id_predmeta
        and a.ocena = 5
    group by
        b.naziv,
        a.godina_roka,
        a.oznaka_roka
)
select
    predmet,
    godina_roka,
    oznaka_roka,
    prosek,
    'polozilo',
    broj
from
    polozio
union
all
select
    predmet,
    godina_roka,
    oznaka_roka,
    prosek,
    'palo',
    broj
from
    pao
)
data initially deferred
refresh deferred
disable query optimization;

```

Za inicijalno punjenje i/ili osvežavanje sadržaja MQT tabele izvršiti:

```

refresh table prosekocenapoispitnimrokovima
allow no access;

```


Izlistavamo sadržaj materijalizovanog upita:

```
select * from prosekocenapoispitnimrokovima;
```

Rezultat:

PREDMET	GODINA_ROKA	OZNAKA_ROKA	PROSEK	POLOZIO_PAO	BROJ
Analiza 1	2015	jan	5.00	palo	1
Engleski jezik 1	2015	jan	5.00	palo	1
Geometrija	2015	apr	5.00	palo	1
Geometrija	2015	feb	5.00	palo	1
Programiranje 1	2015	jan	5.00	palo	1
Analiza 1	2015	feb	7.00	polozilo	1
Analiza 1	2015	jan	8.16	polozilo	6
Engleski jezik 1	2015	jan	6.50	polozilo	4
Geometrija	2015	apr	7.80	polozilo	5
Geometrija	2015	feb	7.00	polozilo	1
Geometrija	2015	jan	7.00	polozilo	2
Programiranje 1	2015	feb	6.50	polozilo	2
Programiranje 1	2015	jan	8.50	polozilo	4

Ne prolazi:

```
update prosekocenapoispitnimrokovima
set oznaka_roka='jan2' where predmet = 'Programiranje 1';
```

jer nije dozvoljeno ažuriranje MQT.

■

13.8 Predavanje 10

13.8.1 Katalog

Informacije o objektima u bazi čuvaju se u katalogu (repozitorijumu, rečniku podataka). Katalog čine tabele i pogledi. Objekte u katalogu ažurira RSUBP, dok korisnici mogu da ih čitaju.

U Db2 postoje sheme SYSIBM sa tabelama, SYSCAT i SYSSTAT sa pogledima.

Neke tabele i pogledi koji prikazuju postojeće objekte i njihove karakteristike su:

- tabele - sysibm.systables, syscat.tables
- atributi - sysibm.syscolumns, syscat.columns

- autorizacija baze - sysibm.sysdbauth, syscat.dbauth
- autorizacija tabela - sysibm.systabauth, syscat.tabauth
- autorizacija atributa - sysibm.syscolauth, syscat.colauth
- ograničenja - sysibm.syschecks, syscat.checks
- indeksi - sysibm.sysindexes, syscat.indexes
- tipovi podataka - sysibm.sysdatatypes, syscat.datatypes
- okidači - sysibm.systriggers, syscat.triggers
- ...

Katalog se koristi prilikom izvršavanja upita za proveru da li objekat postoji, da li postoji odgovarajuća autorizacija. Koristi se za kontrolu stanja u bazi od strane administratora. Takođe, koristi se i za optimizaciju upita, ali to zahteva da statistika bude ažurna.

■ **Primer 13.160** Prikazati veličinu svih ne-sistemskih tabela u bazi u KB, MB i GB.

Rešenje:

```
SELECT
    SUBSTR(TABSCHEMA, 1, 18) TABSCHEMA,
    SUBSTR(TABNAME, 1, 30) TABNAME,
    (
        DATA_OBJECT_P_SIZE + INDEX_OBJECT_P_SIZE +
        LONG_OBJECT_P_SIZE + LOB_OBJECT_P_SIZE +
        XML_OBJECT_P_SIZE
    ) AS TOTAL_SIZE_IN_KB,
    (
        DATA_OBJECT_P_SIZE + INDEX_OBJECT_P_SIZE +
        LONG_OBJECT_P_SIZE + LOB_OBJECT_P_SIZE +
        XML_OBJECT_P_SIZE
    ) / 1024 AS TOTAL_SIZE_IN_MB,
    (
        DATA_OBJECT_P_SIZE + INDEX_OBJECT_P_SIZE +
        LONG_OBJECT_P_SIZE + LOB_OBJECT_P_SIZE +
        XML_OBJECT_P_SIZE
    ) / (1024 * 1024) AS TOTAL_SIZE_IN_GB
FROM
    SYSIBMADM.ADMINTABINFO
WHERE
    TABSCHEMA NOT LIKE 'SYS%';
```

Rezultat:

TABSCHEMA	TABNAME	KB	MB	GB
DA	UPISANKURS	8320	8	0

DA	ISPITNIROK	512	0	0
DA	ISPIT	19712	19	0
DB	STUDIJSKIPROGRAM	768	0	0
DB	PREDMETPROGRAMA	512	0	0
DB	USLOVNIPREDMET	512	0	0
...				

■ **Primer 13.161** Prikazati sve korisnike koji imaju autorizaciju za čitanje ili ažuriranje podataka u tabeli DOSIJE u SHEMI DB, da li je ta autorizacija dalje prenosiva, kao i identifikaciju nosioca koji im je dodelio tu autorizaciju.

Rešenje:

```
select
    GRANTOR as "Vlasnik autorizacije",
    GRANTEE as "Id kome je data autorizacija",
    case
        selectauth
        when 'Y' then 'Ima neprenosivu autorizaciju za čitanje'
        when 'G' then 'Ima autorizaciju za čitanje koju može da prenese'
        else 'Nema autorizaciju za čitanje'
    end,
    case
        updateauth
        when 'Y' then 'Ima neprenosivu autorizaciju za ažuriranje'
        when 'G' then 'Ima autorizaciju za ažuriranje koju može da prenese'
        else 'Nema autorizaciju za ažuriranje'
    end
from
    syscat.tabauth
where
    TABSCHEMA = 'DB'
    and TABNAME = 'DOSIJE'
    and (
        selectauth in ('Y', 'G')
        or updateauth in ('Y', 'G')
    );
```

Rezultat:

Vlasnik autorizacije	Id kome je data autorizacija
-----	-----
SYSIBM	DB2INST1

DB2INST1

PUBLIC

3

 Ima autorizaciju za čitanje koju može da prenese
 Ima neprenosivu autorizaciju za čitanje

4

 Ima autorizaciju za ažuriranje koju može da prenese
 Ima neprenosivu autorizaciju za ažuriranje

■ **Primer 13.162** Napisati upit koji prebrojava koliko puta se u tabeli db.ispit javljaju različite vrednosti svakog atributa. Da ne bi navodili ručno svaki od atributa tabele, može da se napiše upit koji formira upit sa željenim karakteristikama.

Rešenje:

```
with x(atribut_broj_razlicitih,rbrkolone)
as
(
select
  '
    select '''||colname||''',
      count(distinct '''||colname||') as '''||colname||'
    from db.ispit
    union all
  ',
  colno
from   syscat.columns
where  tabname='ISPIT'
and    tabschema='DB'
)

select listagg(atribut_broj_razlicitih,' ')
       within group (order by rbrkolone desc)
from   x;
```

Rezultat:

```
select 'OCENA',
       count(distinct OCENA) as "OCENA"
from db.ispit
```

```

union all

select 'POENI',
       count(distinct POENI) as "POENI"
from db.ispit
union all

select 'DATPOLAGANJA',
       count(distinct DATPOLAGANJA) as "DATPOLAGANJA"
from db.ispit
union all

select 'STATUS',
       count(distinct STATUS) as "STATUS"
from db.ispit
union all

select 'IDPREDMETA',
       count(distinct IDPREDMETA) as "IDPREDMETA"
from db.ispit
union all

select 'INDEKS',
       count(distinct INDEKS) as "INDEKS"
from db.ispit
union all

select 'OZNAKAROKA',
       count(distinct OZNAKAROKA) as "OZNAKAROKA"
from db.ispit
union all

select 'SKGODINA',
       count(distinct SKGODINA) as "SKGODINA"
from db.ispit
union all

```

Kada se izvrši dobijeni upit bez poslednjeg union all, rezultat je:

1	2
OCENA	6
POENI	101

DATPOLAGANJA	813
STATUS	6
IDPREDMETA	476
INDEKS	2760
OZNAKAROKA	10
SKGODINA	5

■ **Primer 13.163** Izlistati iz kataloga sve definisane check-ove za tabele iz sheme DA.

Rešenje: Tražimo spisak kolona iz tabele syscat.checks:

```
describe table syscat.checks;
```

Rezultat:

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
CONSTNAME	SYSIBM	VARCHAR	128	0	No
OWNER	SYSIBM	VARCHAR	128	0	No
OWNERTYPE	SYSIBM	CHARACTER	1	0	No
TABSCHEMA	SYSIBM	VARCHAR	128	0	No
TABNAME	SYSIBM	VARCHAR	128	0	No
CREATE_TIME	SYSIBM	TIMESTAMP	10	6	No
QUALIFIER	SYSIBM	VARCHAR	128	0	No
TYPE	SYSIBM	CHARACTER	1	0	No
FUNC_PATH	SYSIBM	CLOB	2048	0	No
TEXT	SYSIBM	CLOB	2097152	0	No
PERCENTVALID	SYSIBM	SMALLINT	2	0	No
COLLATIONSCHEMA	SYSIBM	VARCHAR	128	0	No
COLLATIONNAME	SYSIBM	VARCHAR	128	0	Yes
COLLATIONSCHEMA_ORDERBY	SYSIBM	VARCHAR	128	0	No
COLLATIONNAME_ORDERBY	SYSIBM	VARCHAR	128	0	Yes
DEFINER	SYSIBM	VARCHAR	128	0	No
ENVSTRINGUNITS	SYSIBM	VARCHAR	11	0	No

Zatim, tražimo željene kolone iz te tabele:

```
select
    char(constname, 20),
    char(tabname, 20),
    text
```

```

from
    syscat.checks
where
    tabschema = 'DA'
order by
    tabname;

```

Rezultat:

1	2	3
CHK_OCENA	ISPIT	(STATUS in ('p','n') and POEN...
CHK_STATUS	ISPIT	STATUS in ('p','n','o','d','x...
CHK_ESPB	PREDMET	ESPB between 1 and 50
CHK_SEMESTAR	PREDMETPROGRAMA	SEMESTAR between 1 and 10
CHK_VRSTA	PREDMETPROGRAMA	VRSTA in ('obavezan','izborni...
CHK_ESPB	PRIZNATISPIT	ESPB between 1 and 50
CHK_OCENA	PRIZNATISPIT	OCENA between 6 and 10 or OCE...
CHK_SEMESTAR	SEMESTAR	SEMESTAR in (1,2)
CHK_SKGODINA	SKOLSKAGODINA	SKGODINA between 2000 and 210...
CHK_STUDIRA	STUDENTSKISTATUS	STUDIRA in (0,1)
CHK_OBIMESPB	STUDIJSKI PROGRAM	OBIMESPB between 30 and 300

■

13.9 Predavanje 11

13.9.1 Transakcije

SQL standard ne opisuje eksplicitne metode zaključavanja. Uopšte i ne pominje zaključavanje kao takvo. Zahteva da se promene koje izvrši transakcija ne vide od strane bilo koji druge transakcije sve dok prva transakcija ne izvrši commit. To podrazumeva da sve transakcije rade u nivou izolovanosti READ COMMITTED, REPEATABLE READ ili SERIALIZABLE.

U SQL-u nivo izolacije se navodi prilikom BEGIN TRANSACTION i može biti:

- SERIALIZABLE
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

SERIALIZABLE je podrazumevana, dok ukoliko je bilo koja druga navedena implementacija može pridružiti bilo koju sa većim nivoom izolacije, pri čemu su nivoi u redosledu

SERIALIZABLE > REPEATABLE READ > READ COMMITTED > READ UNCOMMITTED.

Ukoliko sve transakcije rade u novou izolacije SERIALIZABLE tada se garantuje da će izvršavanje biti serijabilno. Sa druge strane, ukoliko se koristi manji nivo izolacije, moguće je poremetiti serijabilnost. Standard definiše tri vrste prekršaja: prljavo čitanje, neponovljivo čitanje i fantome, kao u tabeli 13.14.

DB2	ANSI	Prljavo pisanje	Prljavo čitanje	Neuzastopno čitanje	Fantomi
UR	READ UNCOMMITTED	Ne	Da	Da	Da
CS	READ COMMITTED	Ne	Ne	Da	Da
RS	REPEATABLE READ	Ne	Ne	Ne	Da
RR	SERIALIZABLE	Ne	Ne	Ne	Ne

Tabela 13.14: SQL nivoi izolacije

Napomena. REPEATABLE READ u SQL standardu nije isto što i repeatable read u DB2. Repeatable read u DB2 je isto što i SERIALIZABLE u SQL standardu.

Većina SQL naredbi se izvršava atomično osim naredbi CALL i RETURN. SQL uvodi START TRANSACTION, COMMIT WORK i ROLLBACK WORK za BEGIN TRANSACTION, COMMIT i ROLLBACK.

```
START TRANSACTION <option commalist>;
```

Pri čemu <option commalist> sadrži način pristupa, nivo izolacije ili oba.

Načini pristupa mogu biti READ ONLY ili READ WRITE. Ukoliko se ni jedan ne navede READ WRITE se podrazumeva, osim ako je naveden nivo izolacije READ UNCOMMITTED, u kom slučaju se READ ONLY podrazumeva. Ukoliko je READ WRITE naveden, nivo izolacije ne sme biti READ UNCOMMITTED.

Nivo izolacije se navodi u obliku:

```
ISOLATION LEVEL <isolation>
```

Pri čemu <isolation> može biti READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ ili SERIALIZABLE.

Sintaksa za COMMIT i ROLLBACK je:

```
COMMIT [WORK] [AND [NO] CHAIN]
```

```
ROLLBACK [WORK] [AND [NO] CHAIN]
```


WORK je reč koja se može izostaviti. AND CHAIN prouzrokuje da se START TRANSACTION sa istim <option commalist> izvrši automatski nakon COMMIT naredbe. AND NO CHAIN je podrazumevano ponašanje.

Nakon COMMIT i ROLLBACK kursori se zatvaraju automatski, osim pri COMMIT kod kursora koji su otvoreni sa WITH HOLD opcijom. WITH HOLD opcija označava da se kursor ne zatvori automatski nakon COMMIT već da čuva pokazivač tako da sledeći FETCH može da se pomeri na sledeći slog u toj sekvenci. Ovime se izbegava moguće složeno pozicioniranje prilikom sledeće OPEN operacije.

SQL podržava tačke pamćenja koje se kreiraju na sledeći način:

```
SAVEPOINT <savepoint name>
```

Može se navesti ime tačke pamćenja koje je lokalno za transakciju.

Naredba kojom se poništavaju radnje do neke tačke pamćenja je:

```
ROLLBACK TO <savepoint name>
```

Naredba kojom se briše tačka pamćenja je:

```
RELEASE <savepoint name>
```

13.10 Predavanje 12

13.10.1 Optimizacija

U Data Studiju se preko Visual Explain mogu videti planovi izvršavanja upita. Može se koristiti i Query Tuning za dobijanje informacija o mogućim poboljšanjima kako bi izvršenje upita bilo efikasnije.

Pri formulisanju select naredbe važe opšta pravila:

- Navesti samo attribute koji su neophodni, izbegavati select * ako nema potrebe.
- Koristiti predikate koji prave restrikciju samo na one attribute koji su potrebni
- Ako je potrebno značajno manji broj slogova od broja postojećih u tabeli koristiti OPTIMIZE FOR klauzu
- Koristiti FOR READ ONLY/FOR FETCH ONLY klauze
- Isključiti DISTINCT/ORDER BY gde nisu neophodni. Koristiti GROUP BY za uklanjanje duplikata umesto DISTINCT
- Koristiti UNION ALL umesto UNION gde je to moguće

- Izbegavati konverziju numeričkih tipova
- Atributi koji se porede treba da budu istog tipa
- Ako je moguće, koristiti sledeće tipove podataka
 - CHAR umesto VARCHAR za kraće attribute
 - Integer umesto FLOAT, DECIMAL ili DECFLOAT
 - DECFLOAT umesto DECIMAL
 - Datumsko-vremenski tip umesto karaktera
 - Brojčane vrednosti umesto karaktera
- Osim u slučaju malih tabela izbegavati `SELECT count(*) from <tabela>` za proveru da li je tabela prazna
- Koristiti IN listu ako se isti atribut javlja u više predikata
- Ako je moguće, izbeći korišćenje OR predikata pri spajanju tabela
- ...

Izbegavati, ukoliko je moguće, korišćenje skalarnih funkcija nad atributima u predikatu.

Umesto

```
select ime, prezime
from   dosije
where  year(datum_rodjenja) = 2002
```

efikasniji zapis je

```
select ime, prezime
from   dosije
where  datum_rodjenja between '2002-01-01'
      and '2002-12-31'
```

Isključiti, ukoliko je moguće, primenu matematičkih funkcija nad atributima u predikatu.

Umesto

```
select indeks,id_predmeta,ocena
from   ispit
where  godina_roka + 5 > 2010
```

efikasniji zapis je

```
select indeks,id_predmeta,ocena
from   ispit
where  godina_roka > 2010 - 5
```

Umesto

```
select distinct id_predmeta, a.godina_roka
from   ispit a, ispitni_rok b
where  a.oznaka_roka = b.oznaka_roka
```

efikasniji zapis je

```
select id_predmeta, a.godina_roka
from   ispit a, ispitni_rok b
where  a.oznaka_roka = b.oznaka_roka
group by id_predmeta, a.godina_roka
```

Umesto

```
select id_predmeta, a.godina_roka
from   ispit a
where  a.oznaka_roka in (select oznaka_roka
                        from ispitni_rok)
```

efikasniji zapis je

```
select id_predmeta, a.godina_roka
from   ispit a
where  exists (select 1
              from   ispitni_rok b
              where  b.oznaka_roka = a.oznaka_roka
              )
```

Ne tražiti podatke koji su već poznati.

Umesto

```
select indeks, id_predmeta, godina_roka, ocena
from   ispit
where  godina_roka = 2020
```

efikasniji zapis je

```
select indeks, id_predmeta, ocena
from   ispit
where  godina_roka = 2020
```

Koristiti CASE umesto UNION, ako je moguće.

Umesto

```
select creator, name, 'Tabela'
from sysibm.systables
where type = 'T'
UNION
select creator, name, 'Pogled'
from sysibm.systables
where type = 'V'
UNION
select creator, name, 'Alias'
from sysibm.systables
where type = 'A'
UNION
select creator, name, 'MQT'
from sysibm.systables
where type = 'S'
order by creator, name
```

Efikasniji zapis je

```
select creator, name, case type
                        when 'T' then 'Tabela'
                        when 'V' then 'Pogled'
                        when 'A' then 'Alias'
                        when 'S' then 'MQT'
                        end
from sysibm.systables
order by creator, name
```

13.11 Predavanje 13

13.11.1 XML

XML dokument se može predstaviti hijerarhijskim modelom - drvetom čvorova koji predstavljaju xml elemente i attribute. Svaki element je sekvenca koja može da se sastoji od nula, jednog ili više objekata. Objekti mogu biti atomične vrednosti ili drugi čvorovi.

Ovde je prikazan samo osnovni skup karakteristika, za više detalja pogledati DB2 XML Guide.

Da bi se napravila tabela koja sadrži xml kolonu, potrebno je da se za tip kolone navede xml:

```
create table dosije_xml (
    indeks            integer    not null,
    podatak           xml        not null,
    primary key       (indeks)
);
```

Vrednost koja može da se unese u xml kolonu je u xml formatu i unos se može izvršiti na sledeći način:

```
insert into
    dosije_xml (indeks, podatak)
VALUES
    (
        20140021,
        '<?xml version="1.0" encoding="UTF-8" ?>
        <student broj_indeksa="20140021">
        <ime>Pera</ime>
        <prezime>Perić</prezime>
        <adresa drzava="Srbija">
            <grad>Beograd</grad>
            <ulica>Bulevar Kralja Aleksandra 123</ulica>
        /adresa>
        <datum_upisa>"06.07.2014"</datum_upisa>
        <datum_rodjenja>"20.01.1995"</datum_rodjenja>
        <telefon tip="mobilni">064-123-456</telefon>
        <telefon tip="fiksni">011-123-456</telefon>
        <prosek>9.45</prosek>
        </student>'
    );
```

13.11.2 XQUERY

XQUERY je funkcionalni programski jezik za čitanje, pretraživanje i menjanje podataka u xml formatu.

Za čitanje xml kolone se može koristiti funkcija `db2-fn:xmlcolumn`.

■ **Primer 13.164** Izdvojiti imena svih studenata.

Rešenje:

Traže se svi elementi ime koji se nalaze u elementima student atributa podatak tabele dosije_xml.

```
xquery
```

```
db2-fn:xmlcolumn('dosije_xml.PODATAK')/student/ime;
```

Rezultat:

```
1
-----
<ime>Miloš</ime>
<ime>Marijana</ime>
<ime>Sanja</ime>
<ime>Nikola</ime>
<ime>Marijana</ime>
<ime>Zorica</ime>
<ime>Zorica</ime>
```

■

Za čitanje xml kolone se može koristiti i db2-fn:sqlquery funkcija u kojoj se navodi select naredba.

■ **Primer 13.165** Izdvojiti prezimena svih studenata.

Rešenje:

Traže se svi elementi prezime koji se nalaze u elementima student atributa podatak tabele dosije_xml.

```
xquery
```

```
db2-fn:sqlquery('select PODATAK from dosije_xml')/student/prezime;
```

Rezultat:

```
1
-----
<prezime>Perić</prezime>
<prezime>Savković</prezime>
<prezime>Terzić</prezime>
<prezime>Vuković</prezime>
<prezime>Savković</prezime>
<prezime>Miladinović</prezime>
<prezime>Miladinović</prezime>
```

■

U okviru xml podatka može se zadati restrikcija na elemente koji sadrže određene attribute i može se zadati željena vrednosti atributa. Na atribut se referiše znakom @.

■ **Primer 13.166** Izdvojiti ime studenta sa indeksom 21/2014.

Rešenje:

Traže se svi elementi ime koji se nalaze u elementima student koji imaju atribut broj indeksa sa vrednošću 20140021, atributa podatak tabele dosije_xml. To će biti jedan element ukoliko je broj indeksa jedinstven.

xquery

```
db2-fn:xmlcolumn('dosije_xml.PODATAK')/student[@broj_indeksa="20140021"]/ime;
```

Rezultat:

1

<ime>Miloš</ime>

■

■ **Primer 13.167** Izdvojiti sve gradove iz kojih su studenti.

Rešenje:

Traže se svi elementi grad koji se nalaze u elementima adresa elemenata student.

xquery

```
db2-fn:xmlcolumn('dosije_xml.PODATAK')/student/adresa/grad;
```

Rezultat:

1

<grad>Beograd</grad>
<grad>Kraljevo</grad>
<grad>Beograd</grad>
<grad/>
<grad>Kraljevo</grad>
<grad>Vranje</grad>
<grad/>

■

13.11.3 XQUERY FLWOR

FLWOR se odnosi na for, let, where, order by, return:

- For i let vezuju jednu ili više promenljivih koje se koriste u ostalim delovima xquery izraza.
- Where filtrira torke, odnosno pravi restrikciju po navedenom uslovu.
- Order by sortira torke po navedenom uslovu.
- Return generiše rezultat flwor izraza.

■ **Primer 13.168** Generisati parove gradova tako da prvi grad pripada jednom skupu, a drugi grad drugom skupu.

Rešenje:

```
xquery
for $i in ("Beograd", "Novi Sad", "Niš")
for $j in ("Subotica", "Preševo")
return <output>{$i, "-", $j}</output>;
```

Rezultat:

```
1
-----
<output>Beograd - Subotica</output>
<output>Beograd - Preševo</output>
<output>Novi Sad - Subotica</output>
<output>Novi Sad - Preševo</output>
<output>Niš - Subotica</output>
<output>Niš - Preševo</output>
```

■ **Primer 13.169** Pronaći imena svih studenata

Rešenje:

Ovaj zadatak se može rešiti na više načina:

```
xquery
for $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")/student
return $i/ime;
```

```
xquery
for $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")/student/ime
return $i;
```



```
xquery
for $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")
return $i/student/ime;

xquery db2-fn:xmlcolumn("dosije_xml.PODATAK")/student/ime;
```

Rezultat:

```
1
-----
<ime>Miloš</ime>
<ime>Marijana</ime>
<ime>Sanja</ime>
<ime>Nikola</ime>
<ime>Marijana</ime>
<ime>Zorica</ime>
<ime>Zorica</ime>
```

- **Primer 13.170** Izdvojiti ime studenta sa indeksom 21/2014.

Rešenje:

```
xquery
for $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")/student
where $i/@broj_indeksa=20140021
return $i/ime;
```

Rezultat:

```
1
-----
<ime>Miloš</ime>
```

- **Primer 13.171** Za studente iz Beograda izdvojiti njihova imena, a rezultat urediti u opadajućem redosledu po indeksu studenata.

Rešenje:

```
xquery
for $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")/student
```

```

let      $j := $i/ime
where    $i/adresa/grad = "Beograd"
order by $i/@broj_indeksa descending
return  $j;

```

Rezultat:

```

1
-----
<ime>Miloš</ime>
<ime>Sanja</ime>

```

■

13.11.4 Formiranje nove XML strukture pomoću konstruktora

■ **Primer 13.172** Za studente iz Beograda izdvojiti njihova imena i gradove. Za svakog studenta formirati xml strukturu:

```

<student>
  <ime_studenta>...</ime_studenta>
  <grad_u_kome_je_rođen>...</grad_u_kome_je_rođen>
</student>

```

Rešenje:

```

xquery
for      $i in db2-fn:xmlcolumn("dosije_xml.PODATAK")/student
where    $i/adresa/grad = "Beograd"
return
  <student>
    <ime_studenta>{$i/ime/text()}</ime_studenta>
    <grad_u_kome_je_rodjen>{$i/adresa/grad/text()}</grad_u_kome_je_rodjen>
  </student>;

```

Rezultat:

```

1
-----
<student>
  <ime_studenta>Miloš</ime_studenta>
  <grad_u_kome_je_rodjen>Beograd</grad_u_kome_je_rodjen>

```

```

</student>
<student>
  <ime_studenta>Sanja</ime_studenta>
  <grad_u_kome_je_rodjen>Beograd</grad_u_kome_je_rodjen>
</student>

```

■

Uporediti prethodni upit sa upitom bez text().

13.11.5 Operatori

■ **Primer 13.173** Vratiti strukturu oblika sa svim gradovima i ulicama:

```

<gradovi_i_ulice>
  <grad>...</grad>
  <ulica>...</ulica>
</gradovi_i_ulice>

```

Rešenje:

```

xquery
let $gradstanovanja :=
  db2-fn:xmlcolumn('dosije_xml.PODATAK')/student/adresa/grad
let $ulicastanovanja :=
  db2-fn:xmlcolumn('dosije_xml.PODATAK')/student/adresa/ulica
return
  <gradovi_i_ulice>
    { $gradstanovanja union $ulicastanovanja }
  </gradovi_i_ulice>;

```

Rezultat:

```

1
-----
<gradovi_i_ulice>
  <grad>Beograd</grad>
  <ulica>Bulevar Kralja Aleksandra 123</ulica>
  <grad>Kraljevo</grad>
  <ulica>Kraljevačkog bataljona 46B</ulica>
  <grad>Beograd</grad>
  <ulica>Studentski trg 22/IV</ulica>

```

```

    <grad/><ulica/>
    <grad>Kraljevo</grad>
    <ulica>Žička 17</ulica>
    <grad>Vranje</grad>
    <ulica>Bore Stankovića 12</ulica>
    <grad/><ulica/>
  </gradovi_i_ulice>

```

■

13.11.6 Korišćenje agregatnih funkcija

■ **Primer 13.174** Za sve studente izračunati koliko ukupno ima različitih brojeva telefona, i koji je prosek svih proseka studenata.

Rešenje:

```

xquery
let $studenti := db2-fn:xmlcolumn('dosije_xml.PODATAK')/student
return
  <info>
    {'Ukupno različitih brojeva telefona='}
    {count($studenti//telefon)}
    {' , a prosek ocena studenata je '}
    {avg ($studenti//prosek)}
  </info>;

```

Rezultat:

```

1
-----
<info>
  Ukupno različitih brojeva telefona=16, a prosek ocena studenata je 7.88125
</info>

```

■

13.11.7 SQL/XML

XML se može pretraživati i putem select naredbe.

■ **Primer 13.175** Izdvojiti elemente prezime svih studenata.

Rešenje:

```
select xmlquery('$PODATAK/student/prezime')
from   dosije_xml;

select xmlquery('$i/student/prezime' passing podatak as "i")
from   dosije_xml;
```

Rezultat:

```
1
-----
<prezime>Perić</prezime>
<prezime>Savković</prezime>
<prezime>Terzić</prezime>
<prezime>Vuković</prezime>
<prezime>Savković</prezime>
<prezime>Miladinović</prezime>
<prezime>Miladinović</prezime>
```

- **Primer 13.176** Izdvojiti samo tekstualne vrednosti elemenata prezime svih studenata.

Rešenje:

```
select xmlquery('$PODATAK/student/prezime/text()')
from   dosije_xml;
```

Rezultat:

```
1
-----
Perić
Savković
Terzić
Vuković
Savković
Miladinović
Miladinović
```

Uklanjanje xml oznaka pomoću xmlcast.

```
select xmlcast(xmlquery('$PODATAK/student/prezime') as character(15))
from   dosije_xml;
```

Rezultat:

```

1
-----
Perić
Savković
Terzić
Vuković
Savković
Miladinović
Miladinović

```

- **Primer 13.177** Izdvojiti imena, prezimena i datume upisa svih studenata.

Rešenje:

```

select indeks as "Indeks",
       xmlquery('$PODATAK/student/prezime/text()') as "Prezime",
       xmlquery('$PODATAK/student/ime/text()') as "Ime",
       xmlquery('$PODATAK/student/datum_upisa/text()') as "Datum_upisa"
from   dosije_xml;

```

Rezultat:

Indeks	Prezime	Ime	Datum_upisa
-----	-----	-----	-----
20140021	Perić	Miloš	"06.07.2014"
20140022	Savković	Marijana	"05.07.2014"
20130023	Terzić	Sanja	"04.07.2013"
20130024	Vuković	Nikola	"04.07.2013"
20140025	Savković	Marijana	"06.07.2014"
20140026	Miladinović	Zorica	"06.07.2014"
20130027	Miladinović	Zorica	"03.09.2013"

- **Primer 13.178** Izdvojiti adrese koje su iz Srbije.

```

select xmlquery('$i/student/adresa[@drzava="Srbija"]' passing podatak as "i")
from   dosije_xml;

```

Rezultat:

1

```

-----
<adresa drzava="Srbija">
  <grad>Beograd</grad>
  <ulica>Bulevar Kralja Aleksandra 123</ulica>
</adresa>
<adresa drzava="Srbija">
  <grad>Kraljevo</grad>
  <ulica>Kraljevačkog bataljona 46B</ulica>
</adresa>
<adresa drzava="Srbija">
  <grad>Beograd</grad>
  <ulica>Studentski trg 22/IV</ulica>
</adresa>
<adresa drzava="Srbija">
  <grad/><ulica/>
</adresa>
<adresa drzava="Srbija">
  <grad>Kraljevo</grad>
  <ulica>Žička 17</ulica>
</adresa>
<adresa drzava="Srbija">
  <grad>Vranje</grad>
  <ulica>Bore Stankovića 12</ulica>
</adresa>

```

■

■ **Primer 13.179** Ispitati da li je adresa iz Srbije.

```

select xmlquery('$i/student/adresa/@drzava="Srbija"' passing podatak as "i")
from   dosije_xml;

```

Rezultat:

1

```

-----
true
true
true
true
true
true

```

```
false
false
```

■

13.11.8 Upotreba XMLTABLE

- **Primer 13.180** Izdvojiti imena, prezimena, ulice i gradove svih studenata.

Rešenje:

```
select dosije.*
from   dosije_xml,
       xmltable('$PODATAK/student'
               columns
                 broj_indeksa integer PATH '@broj_indeksa',
                 ime varchar(15) PATH 'ime',
                 prezime varchar(15) PATH 'prezime',
                 ulica varchar(20) PATH 'adresa/ulica',
                 grad varchar(15) PATH 'adresa/grad')
AS dosije;
```

Rezultat:

BROJ_INDEKSA	IME	PREZIME	ULICA	GRAD
20140021	Miloš	Perić	Bulevar Kralja Aleks	Beograd
20140022	Marijana	Savković	Kraljevačkog batalj	Kraljevo
20130023	Sanja	Terzić	Studentski trg 22/IV	Beograd
20130024	Nikola	Vuković		
20140025	Marijana	Savković	Žička 17	Kraljevo
20140026	Zorica	Miladinović	Bore Stankovića 12	Vranje
20130027	Zorica	Miladinović		

■

- **Primer 13.181** Izdvojiti informacije o studentima.

Rešenje:

```
select dosije.*
from   dosije_xml,
       xmltable('$PODATAK/student' columns
                 broj_indeksa integer path '@broj_indeksa',
```



```

ime          varchar(15) path 'ime',
prezime      varchar(15) path 'prezime',
telefon1     varchar(15) path 'telefon[1]',
telefon2     varchar(15) path 'telefon[2]'
) as dosije
order by broj_indeksa desc;

```

Rezultat:

BROJ_INDEKSA	IME	PREZIME	TELEFON1	TELEFON2
20140026	Zorica	Miladinović	064-776-4332	017-654-334
20140025	Marijana	Savković	063-654-332	036-323-444
20140022	Marijana	Savković	065-654-321	036-23-456
20140021	Miloš	Perić	064-123-456	011-123-456
20130027	Zorica	Miladinović	060-889-4332	
20130024	Nikola	Vuković	063-343-545	
20130023	Sanja	Terzić	066-316-321	011-323-456

■

13.11.9 Upotreba XMLEXISTS

■ **Primer 13.182** Izdvojiti imena, prezimena, datume upisa studenata iz Beograda.

Rešenje:

```

select indeks as "Indeks",
       xmlquery('$PODATAK/student/prezime/text()') as "Prezime",
       xmlquery('$PODATAK/student/ime/text()') as "Ime",
       xmlquery('$PODATAK/student/datum_upisa/text()') as "Datum_upisa"
from   dosije_xml
where  xmlexists('$PODATAK/student[adresa/grad = "Beograd"]');

```

Rezultat:

Indeks	Prezime	Ime	Datum_upisa
20140021	Perić	Miloš	"06.07.2014"
20130023	Terzić	Sanja	"04.07.2013"

■

Bibliografija

- [1] C.J.Date. *An Introduction to Database Systems*. 8. izdanje. Addison Wesley Inc, 2004.
- [2] C.J.Date. *Database Design and Relational Theory: Normal Forms and All That Jazz*. O'Reilly Media Inc, 2019.
- [3] C.J.Date. *SQL and Relational Theory: How to Write Accurate SQL Code*. O'Reilly Media Inc, 2015.
- [4] Saša Malkov. *Skripta za predmet Relacione baze podataka*.
- [5] Nenad Mitić. *Materijali za kurs Uvod u relacione baze podataka*. URL: <http://poincare.matf.bg.ac.rs/~nenad.mitic/rbp.html>.
- [6] Gordana Pavlović-Lažetić. *Osnove relacionih baza podataka*. 2. izdanje. Matematički fakultet, 1999.