Uvod u relacione baze podataka

Skripta za šk. 2022/2023. godinu

Mirjana Maljković

Sadržaj

1. Uvod

Rukopis sadrži materijal koji prati izlaganja na predavanjima od prve do šeste nedelje u okviru kursa Uvod u relacione baze podataka na Matematičkom fakultetu za školsku 2022/2023. godinu. U rukopisu je sažeto predstavljen materijal iz glavne literature za predmet - materijali profesora dr Nenada Mitića i knjiga *An Introduction to Database Systems* 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 podatka, 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 [4].

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 [3]. 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]

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 redudantnosti (ponavljanja). Sistemi koji koriste bazu podatka 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 korsnik 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 podatka. 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

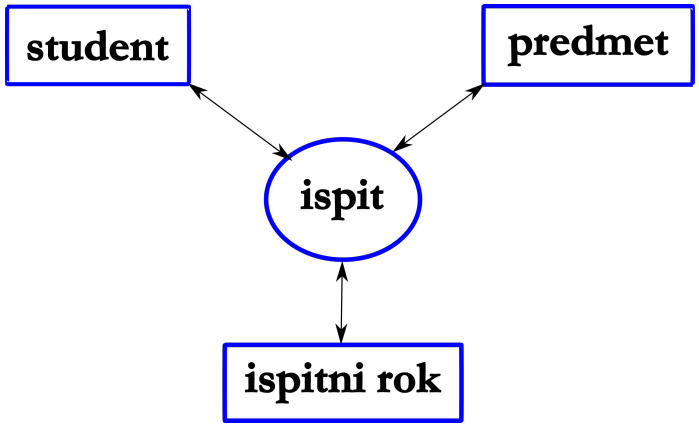
lice, već pripada upravljačkim strukturama. Administrator baze podataka (eng. database administrator, DBA) je tehničko lice koje pravi bazu i implementira kontrolne strukture. DBA je odgovoran za implementaciju odluka DA kao i za rad sistema, performanse, ...

Baza podataka je skup postojanih 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 2.1. 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

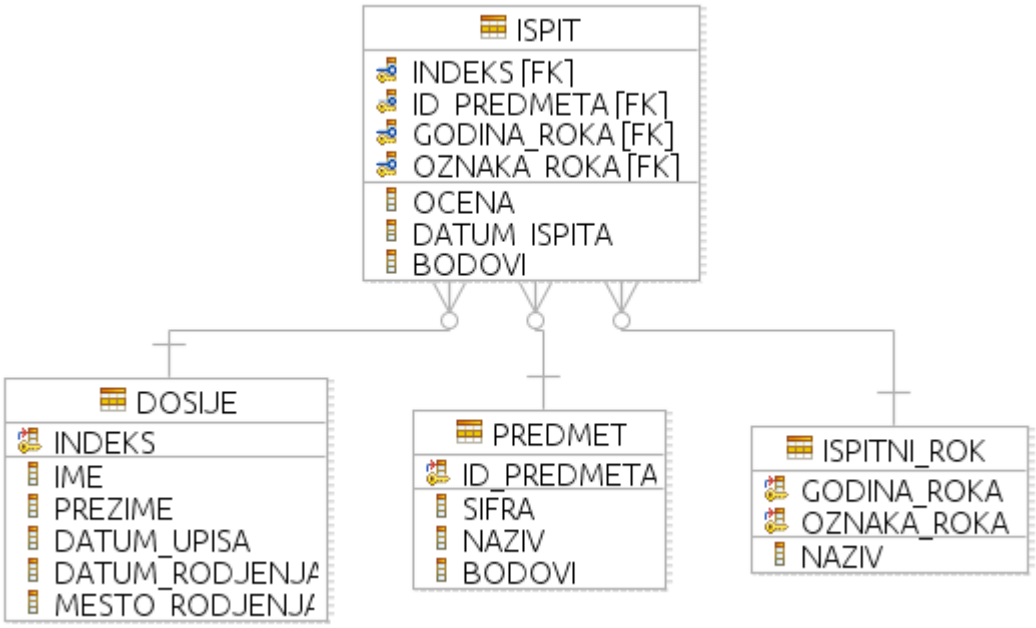
2.4 Informacije o studentima

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. Svki 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 redudantnosti podataka eliminiše se i nekonzistetntnost, 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 prebacijvanje 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 razčilite 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 struktuirani 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 obezbeditit 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 brojčanih podataka - npr. da li se koristi zapis u fiksnom ili pokretnom zarezu, koja je preciznost (broj cifara), ...
- reprezentacija znakovnih podataka

- jedinice za brojčane 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 ($i = 1, 2, \ldots, n$), 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 $A_i : T_i$ $(i = 1, 2, ..., n)$ 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;
- torka je svaki red u tabeli;
- atributi su kolone Indeks, Ime, Prezime, Datum_rodenja, Mesto_rodenja, 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_RODJENJA	MESTO_RODJENJA	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 atribute 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 ugneždeni 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

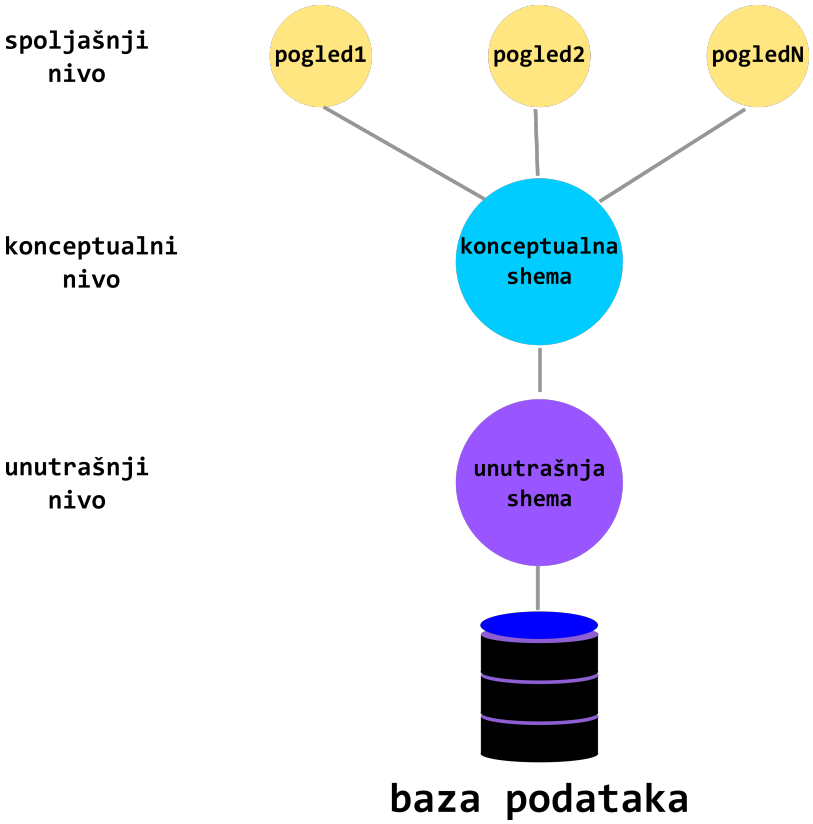
Arhitektura baza podataka koja je predložena od strane ANSI[1]/SPARC[2] studijske grupe za sisteme za upravljanje podacima se sastoji od tri nivoa[4]:

- unutrašnji nivo (interni nivo, fizički nivo) koji se odnosi na način kako se podaci čuvaju u sistemu;
- konceptualni nivo (zajednicki 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

[1] ANSI - American National Standards Institute
[2] ANSI/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:

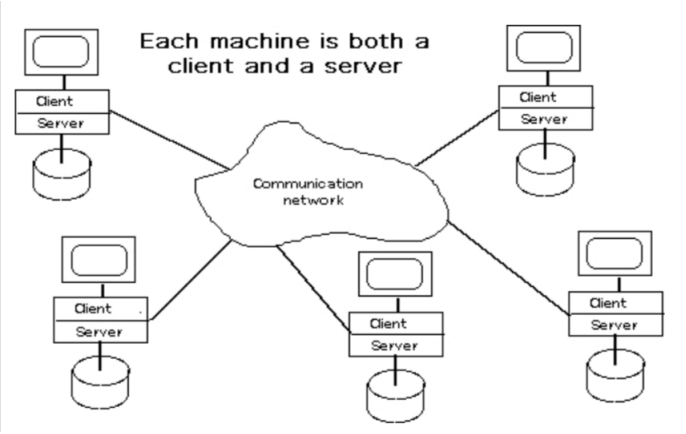
- LOAD rutina koja obezbeđuje pravljenje inicijalne verzije baze podataka iz regularnih datoteka
- UNLOAD/RELOAD rutine koje obezbeđuju čuvanje cele ili dela baze podataka i ponovno učitavanje podataka iz rezervne kopije
- programi za računanje raznih statistika
- programi za analizu statistika

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 kombiunuju 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

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[4].

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 pirmenjuju 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 odogovarajuće telo.

4.4 Semantika originalne algebre

4.4.1 Restrikcija

Neka relacija R ima atribute $X, Y, \ldots Z$ i neka je u istinitosna funkcija čiji su parametri neki podskup od $X, Y, \ldots Z$. 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

Neka relacija R ima bar atribute $X, Y, ...Z$. Projekcija relacije R na $X, Y, ...Z$ je relacija čije

- zaglavlje je izvedeno iz relacije R uklanjanjem svih atributa koji se ne nalaze u skupu $\{X, Y, ..., Z\}$;
- telo se sastoji od svih torki $\{X : x, Y : y, ..., Z : z\}$ 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 atribute relacije R bez atributa $X, Y, ..., Z$.

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

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

■

4.4.3 Dekartov proizvod

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
a	b
c	d

times

Y_1	Y_2
1	2
3	4

=

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

Neka relacije R_1 i R_2 imaju sledeća zaglavlja

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

,

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

Neka su atributi $\{X_1, X_2, \ldots, X_m\}$ označeni sa X, atributi $\{Y_1, Y_2, \ldots, Y_n\}$ sa Y, a atributi $\{Z_1, Z_2, \ldots, Z_p\}$ 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
a	b
c	d

join

Y_1	X_2
1	b
3	f

=

X_1	X_2	Y_1
a	b	1

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 Θ = '=' 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 pojaljuju u relaciji R_1 ili R_2.

Sintaksa operatora za uniju je:

```
R1 union R2
```

Ilustracija primene unije nad relacijama je prikazana u tabeli 4.5.

X_1	X_2
a	b
c	d

union

X_1	X_2
c	d
e	f

=

X_1	X_2
a	b
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 pojaljuju i u relaciji R_1 i u relaciji R_2.

Sintaksa operatora za presek je:

```
R1 intersect R2
```

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
a	b
c	d

intersect

X_1	X_2
c	d
e	f

=

X_1	X_2
c	d

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 pojaljuju 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
a	b
c	d

minus

X_1	X_2
c	d
e	f

=

X_1	X_2
a	b

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 where u_1 or u_2 = (R where u_1) union (R where u_2)

- R where u_1 and u_2 = (R where u_1) intersect (R where u_2)

- R where not (u) = R minus (R 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
a	1
a	2
b	1
c	1
c	2
d	2

divideby

X_2
1
2

=

X_1
a
c

Tabela 4.8: Ilustracija primene deljenja nad relacijama

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

divideby

X_3	X_4
1	2
3	4

=

X_1	X_2
a	b
g	h

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;
- definisnje 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 INERSECT 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 INERSECT 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 deklarisana. 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 promeljiva 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 deklarisana. 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 atribute 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 promenjive važi zadati uslov, a netačno inače.

Uslovni izraz

```
forall promenljiva(uslov)
```

izračunava tačno ako za sve torke promenjive 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 opertatora `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 promenjivih 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

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($X_1, X_2, ..., X_n$) predstavlja podskup atributa X te relacije, ako važi:

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

- 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

$$T_n \to T_{n-1} \to T_{n-2} \to \ldots \to T_1 \to T_n$$

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)`.

7. Relacioni upitni jezik SQL

7.1 Predavanja 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 relacioni 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 [4]. 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 7.1** Napisati naredbu za pravljenje baze podataka stud2020

```
create database stud2020
```

■

Naredba za brisanje baze podataka ima sintaksu

```
drop database ime-baze-podataka
```

■ **Primer 7.2** 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-kluč]?
[, def-strani-kluč]*)
```

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 nedeostajuće 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.

7.1.1 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 7.3** Prikazati kompletan sadržaj tabele dosije.

```
select *
from dosije
```

■

■ **Primer 7.4** Prikazati broj indeksa, ime i prezime studenta.

```
select indeks, ime, prezime
from dosije
```

■

■ **Primer 7.5** Prikazati informacije o studentima koji su rođeni u Beogradu.

```
select *
from dosije
where mesto_rodjenja = 'Beograd'
```

■

■ **Primer 7.6** Prikazati informacije o studentima koji su nisu rođeni u Beogradu.

```
select *
from dosije
where mesto_rodjenja <> 'Beograd'
```

■

■ **Primer 7.7** 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 7.8** 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 7.9** Prikazati imena i prezimena studenata za koje nije poznato mesto rođenja.

```
select ime, prezime, mesto_rodjenja
from dosije
where mesto_rodjenja is null
```

■

■ **Primer 7.10** Prikazati imena i prezimena studenata za koje je poznato mesto rođenja.

```
select ime, prezime, mesto_rodjenja
from dosije
where mesto_rodjenja is not null
```

■

7.2 Predavanje 2

7.2.1 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 7.11** 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 7.12** 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'
```

■

7.2.2 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

```
izaz 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

```
izaz like maska escape prekidacki-simbol
```

■ **Primer 7.13** 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 7.14** 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 7.15** 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%'
```

■

7.2.3 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 7.16** 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 7.17** 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 7.18** 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 7.19** 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 ispisano *Dvostruka vrednost broja bodova je* =.

```
select id_predmeta, sifra,
       'Dvostruka vrednost broja bodova je =' as "Opis",
       bodovi* 2 as "Dvostruko"
from predmet
```

■

■ **Primer 7.20** 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
```

■

7.2.4 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 restućem poretku 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 7.21** 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 7.22** 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
```

Primetiti da dodeljeno ime koloni u klauzuli `select` može da se koristi u klauzuli `order by` jer se klauzula `order by` izvršava nakon klauzule `select`.

■

7.2.5 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 7.23** Prikazati sve moguće kombinacije studenata i predmeta koje oni mogu da odaberu.

```
select *
from  dosije, predmet
```

ili

```
select *
from  dosije cross join predmet
```

■

7.2.6 Spajanje tabela

Unutrašnje spajanje tabela

Dve tabela se mogu spojiti primenom proizvoda nad tim tabelama, a zatim restrikcije radi navođenja uslova spajanja.

■ **Primer 7.24** 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 mu 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 7.1.

a	c
1	m
2	n
3	l

(a) Tabela A

b	c
4	m
5	n
6	r

(b) Tabela B

Tabela 7.1: 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 7.2. Kako bi izgledao rezultat da je u klauzuli `select` navedena `*`?

a	**c**	b
1	m	4
2	n	5

Tabela 7.2: Rezultat primera unutrašnjeg spajanja tabela A i B

■ **Primer 7.25** 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 7.1 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 7.3.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
3	l	null	null

Tabela 7.3: Rezultat primera levo spošašnjeg 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 7.1 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 7.4.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
null	null	r	6

Tabela 7.4: Rezultat primera desno spošašnjeg 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 7.1 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 7.5.

a	tabelaA.c	tabelaB.c	b
1	m	m	4
2	n	n	5
3	l	null	null
null	null	r	6

Tabela 7.5: Rezultat primera potpuno spoljašnjeg spajanja tabela A i B

■ **Primer 7.26** 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
```

■

7.3 Predavanje 3

7.3.1 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 7.27** 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 7.28** 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'
```

■

7.3.2 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 7.6 i 7.7.

X
1
2
2
3

union

X
1
2
3
4
4

=

X
1
2
3
4

Tabela 7.6: Primer primene operatora za uniju

■ **Primer 7.29** Prikazati broj bodova i nazive svih predmeta ako predmeti pripadaju grupi Analiza ili su polagani u januarskom ispitnom roku 2015. godine.

X
1
2
2
3

union all

X
1
2
3
4
4

=

X
1
1
2
2
2
3
3
4
4

Tabela 7.7: Primer primene operatora za uniju

```
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 7.30** 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 7.31** 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 7.32** 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
```

■

7.3.3 Podupiti

U okviru upita mogu se pisati i podupiti. Na primer, u okviru klauzule `where` kao deo operatora `in` može se navesti podupit.

■ **Primer 7.33** 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 7.34** 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 7.35** 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.

7.4 Predavanje 4

7.4.1 Kvantifikovana poređenja

Kvantifikovanjem poređenja moguće je porediti skalarnu vrednost sa tabelom skalarnih vrednosti[2]. 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 7.36** 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 7.37** Pronaći ispit koji nije položen sa najvećim brojem bodova.

```
select *
from ispit
where bodovi < any (select bodovi
                    from ispit
                    where bodovi is not null)
```

■

7.4.2 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 7.38** 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 7.39** 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 7.40** 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
                             )
                  )
```

■

7.4.3 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 7.41** Prikazati tabelu sa sledećim sadržajem

1
a
b
c

```
values ('a'), ('b'), ('c')
```

■

■ **Primer 7.42** Prikazati tabelu sa sledećim sadržajem

1	2
a	1
b	2
c	3

```
values ('a', 1), ('b', 2), ('c', 3)
```

■

7.4.4 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 7.43** 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)
```

■

7.4.5 Skalarne funkcije

Skalarne funkcije su one funkcije koje kao argumente dobijaju skalarne vrednosti (jedan argument je jedna vrednost), i kao rezultat vraćaju skalarnu vrednost.

U tabeli 7.8 je dat opis nekih specifičnih skalarnih funkcija, a u tabelama 7.9 i 7.10 je dat opis nekih skalarnih funkcija za rad sa niskama.

Funkcija	**Opis**
coalesce(arg1, arg2, ...)	vraća prvi argument iz liste argumenata čija vrednost nije null
value(arg1, arg2, ...)	isto kao coalesce(arg1, arg2, ...)

Tabela 7.8: Specifične skalarne funkcije

■ **Primer 7.44** 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 7.45** 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
```

Funkcija	**Opis**
substr(niska_arg, početak_arg, dužina_arg)	vraća podnisku iz *niska_arg* dužine *dužina_arg* koja počinje od pozicije *početak_arg*
concat(arg1, arg2) ili sa korišćenjem operatora arg1 || arg2	spajanje niski
space(arg1)	vraća *arg1* praznih mesta
posstr(arg1, arg2)	traži gde počinje niska *arg2* u *arg1*
repeat(arg1, arg2)	ponavlja nisku *arg1* *arg2*. puta
replace(arg1, arg2, arg3)	zamenjuje sva pojavljivanja niske *arg2* u *arg1* sa niskom *arg3*
length(arg1)	vraća dužinu niske *arg1*
lcase(arg1) ili lower(arg1)	vraća nisku *arg1* u kojoj su velika slova zamenjena malim slovima
ucase(arg1) ili upper(arg1)	vraća nisku *arg1* u kojoj su mala slova zamenjena velikim slovima
left(arg1, arg2)	vraća prefiks niske *arg1* dužine *arg2*
right(arg1, arg2)	vraća sufiks niske *arg1* dužine *arg2*
rtrim(arg1)	briše beline sa kraja niske *arg1* i vraća nisku koju dobije kao rezultat
ltrim(arg1)	briše beline sa početka niske *arg1* i vraća nisku koju dobije kao rezultat
trim(arg1)	briše beline sa početka i kraja niske *arg1* i vraća nisku koju dobije kao rezultat

Tabela 7.9: Opis nekih skalarnih funkcija za rad sa niskama

ili

```
select substr(naziv,2,2)  || sifra
from predmet
```

Primer rezultata: naM113

■

■ **Primer 7.46** Ispisati nisku koja reč *Tekst* sadrži 3 puta.

```
values (repeat('Tekst',3))
```

ili

```
select repeat('Tekst',3)
from sysibm.sysdummy1
```

Funkcija	Opis
char(arg1)	pretvara vrednosti različitih tipova u nisku koju vraća kao rezultat. Niska će imati dužinu koja zavisi od tipa argumenta
char(niska_arg1, dužina_arg2)	vraća prvih *dužina_arg2* karaktera niske *niska_arg1*
insert(niska_arg, početak_arg, dužina_arg, zamena_arg)	podnisku dužine *dužina_arg* koja počinje na poziciji *početak_arg* u okviru niske *niska_arg* zamenjuje sa *zamena_arg*
locate_in_string(niska_arg1, podniska_arg2, počevšiod_arg3, brpojavljivanja_arg4)	vraća poziciju *brpojavljivanja_arg4*. pojavljivanja niske *podniska_arg2* u okviru niske *niska_arg1* počevši od pozicije *počevšiod_arg3*
locate(podniska_arg1, niska_arg2, pozicija_arg3)	vraća poziciju prvog pojavljivanja niske *podniska_arg1* u okviru niske *niska_arg2* počevši od pozicije *pozicija_arg3*
translate(niska_arg1, zamenisa_arg2, zameni_arg3, podrazumevano_arg4)	vraća nisku *niska_arg1* u kojoj su karakteri navedeni u *zameni_arg3* zamenjeni sa *zamenisa_arg2*. Karakter koji se menja i njegova zamena se uparuju prema poziciji. Karakterom *podrazumevano_arg4* se zamenjuju karakteri u *zameni_arg3* koji nemaju svog para u *zamenisa_arg2*.

Tabela 7.10: Opis nekih skalarnih funkcija za rad sa niskama

Rezultat: TekstTekstTekst

■

■ **Primer 7.47** Ispisati mesta rođenja studenata i umesto *Beograd* ispisati *Bg*.

```
select replace(mesto_rodjenja,'Beograd','Bg')
from dosije
```

■

■ **Primer 7.48** Ispisati nazive predmeta samo sa malim slovima, a zatim samo sa velikim slovima.

```
select lcase(naziv), lower(naziv), ucase(naziv), upper(naziv)
from predmet
```

■

■ **Primer 7.49** 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 7.50** 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 7.51** 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 7.52** 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 7.53** 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 7.54** Odrediti poziciju trećeg pojavljivanja niske *ra* u reči *Abrakadabrara*.

```
values locate_in_string('Abrakadabrara','ra',1,3)
```

Rezultat: 12

■

■ **Primer 7.55** 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-

■

7.5 Predavanje 5

U tabeli 7.11 je dat opis nekih skalaranih funkcija za rad sa datumima, a u tabeli 7.12 za rad sa vremenom.

■ **Primer 7.56** Prikazati današnji datum u različitim formatima.

```
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 7.57** Zapisati datum 5.5.2023.

```
values date('5.5.2023')
```

ili

```
values cast('5.5.2023' as date)
```

Primer rezultata: 2023-05-05

■

■ **Primer 7.58** 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 7.59** 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
```

Funkcija	**Opis**
date(arg1)	vraća vrednost tipa date za *arg1*, a datum se zadaje kao niska
cast(arg1 as arg2)	vraća vrednost *arg1* tipa *arg2*
timestamp(arg1, arg2)	vraća vrednost tipa timestamp definisanu argumentima od kojih *arg1* predstavlja datum, a *arg2* vreme
char(datetime_arg, format_arg)	vraća datum ili vreme koji su zadatih sa *datetime_arg* u željenom formatu koji se zadaje sa *format_arg*
year(datum_arg1)	vraća godinu iz *datum_arg1*, rezultat je ceo broj
month(datum_arg1)	vraća mesec iz *datum_arg1*, rezultat je ceo broj
day(datum_arg1)	vraća dan iz *datum_arg1*, rezultat je ceo broj
week(datum_arg1)	vraća sedmicu u godini za *datum_arg1* (rezultat je u intervalu 1- 54, sedmica se broji od nedelje)
dayofyear(datum_arg1)	vraća dan u godini za *datum_arg1*
dayofweek(datum_arg1)	vraća redni broj dana u sedmici za *datum_arg1* (rezultat u intervalu od 1 do 7, gde 1 označava nedelju)
dayname(datum_arg1)	vraća naziv dana u sedmici za *datum_arg1*
monthname(datum_arg1)	vraća naziv meseca za *datum_arg1*
monthname(datum_arg1, zapis_arg2)	vraća naziv meseca za *datum_arg1* u zapisu *zapis_arg2* videti cldr.unicode.org
last_day(datum_arg1)	vraća datum poslednjeg dana meseca u kome je *datum_arg1*
next_day(datum_arg1, dan_arg2)	vraća datum koji odgovara prvom datumu koji je posle *datum_arg1* i čije je ime dana zadato sa *dan_arg2*, koji može biti MON, TUE, WED ...
days(datum_arg1)	vraća celobrojnu reprezentaciju datuma

Tabela 7.11: Opis funkcija za rad sa datumima

■

■ **Primer 7.60** 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 7.61** 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 7.62** 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 7.63** 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 *ymmdd*, 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 7.64** 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 7.65** Koliko vremena je prošlo od 5.5.2020?

```
values current date - date('5.5.2020')
```

Primer rezultata:20525 ■

■ **Primer 7.66** 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 *arg1*
minute(arg1)	izdvaja minute iz vremena zadatog sa *arg1*
second(arg1)	izdvaja sekunde iz vremena zadatog sa *arg1*
microsecond(arg1)	izdvaja mikrosekunde iz vremena zadatog sa *arg1*

Tabela 7.12: Skalarne funkcije za rad sa vremenom

■ **Primer 7.67** 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 7.13 je dat opis nekih funkcija za konverziju i rad sa brojevima.

■ **Primer 7.68** Prikazati broj 1234.5656 u formatu 8.2.

```
select   decimal('1234.5656',8,2)
from sysibm.sysdummy1
```

Rezultat: 1234.56

■

■ **Primer 7.69** 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 7.70** Prikazati rezultat primene funkcija floor i ceil na broj 65.6.

Funkcija	Opis
integer(arg1)	konvertuje vrednost *arg1* (broj ili niska) u tip integer
decimal(arg1, arg2, arg3)	vraća decimalnu reprezentaciju *arg1* (broj ili niska) pri čemu *arg2* predstavlja ukupan broj cifara, a *arg3* broj mesta iza decimalne tačke
digits(arg1)	vraća *arg1* kao nisku koja sadrži vrednost *arg1* 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 *arg1* (broj ili niska) u tip double (zapis u pokretnom zarezu dvostruke tačnosti)
real(arg1)	konvertuje vrednost *arg1* (broj ili niska) u tip real (zapis u pokretnom zarezu jednostruke tačnosti)
round(arg1, arg2)	vraća vrednost *arg1* zaokruženu na *arg2* broj decimala ako je *arg2* pozitivan broj, a ako je negativan vraća zaokružen broj *arg1* na *arg2* broj mesta u levo od tačke
ceil(arg1)	vraća najmanji ceo broj koji je jednak ili veći od *arg1*
floor(arg1)	vraća najveći ceo broj koji je jednak ili manji od *arg1*

Tabela 7.13: Funkcije za konverziju i rad sa brojevima

```
select floor(65.6), ceil(65.6)
from sysibm.sysdummy1
```

Rezultata: 65 66 ■

■ **Primer 7.71** 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 ■

7.5.1 Agregatne funkcije

Agregatne funkcije se koriste kada je potrebno izvršiti neke operacije nad svim entitetima koji ulaze u rezultat upita[2]. Osnove agregatne fukcije 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 funkacija, 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 7.72** 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
```

■

7.5.2 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 7.73** 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 7.74** 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 7.75** 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 7.76** 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 prosecna decimalna
                                        vrednost ocene" ,
       dec(round(avg(dec(ocena,7,4)),2),7,2) as "Zaokruzena prosecna
                                                 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
```

■

7.5.3 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 7.77** 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 7.78** 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 7.79** Prikazati uređene četrvorke (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 na-

ziv_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 poretkom)]?
```

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 poretkom)` se zadaje redosled po kome će se vršiti spajanje niski.

■ **Primer 7.80** Prikazati nisku koja sadrži imena studenata koja su uređena prema godini njihovog rođenja.

```
select listagg(ime,' ,')
        within group (order by datum_rodjenja)
from   dosije a
```

■

7.5.4 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 7.81** 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
```

■

7.5.5 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 7.82** 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
```

■

7.5.6 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 7.83** 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
```

■

7.5.7 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 7.84** 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 7.85** 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

■

7.5.8 Dodavanje entiteta

Za dodavane novih entiteta u tabelu koristi se naredba `insert` koja se upotreblajva 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 7.86** 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 7.87** 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 7.88** 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'
```

■

7.5.9 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 promenjenji svi entiteti tabele.

■ **Primer 7.89** 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 7.90** 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 posledjnjeg 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)
```

■

7.6 Predavanje 6

7.6.1 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 postojći redovi iz tabele će biti obrisani. Za postavljanje složenijih uslova koriste se podupiti u okviru klauzule `where`.

■ **Primer 7.91** Obrisati sve ispite na kojima je polagan predmet Razvoj softvera (id 4005).

```
delete from ispit
where id_predmeta=4005
```

■

■ **Primer 7.92** 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)
```

■

7.6.2 Menjanje entiteta jedne tabele na osnovu sadržaja druge tabele/tabela

Naredba `merge` menja entitete u ciljnoj tabeli koristeći podatke iz druge tabele ili rezultata upita. Sintaksa narebe `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 7.93** 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 7.94** U tabelu dosije_prosek uneti indekse studenata iz Beograda.

```
insert into dosije_prosek (indeks)
select indeks
from dosije
where mesto_rodjenja='Beograd'
```

■

■ **Primer 7.95** 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)
```

■

7.6.3 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 7.96** 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 7.97** Urediti studente po indeksu i prikazati 5 studenata, ali izostaviti prva 3 studenta.

```
select *
from dosije
order by indeks
limit 5 offset 2
```

■

7.6.4 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 korisi 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

- relaciona baza podataka (`database`)
- šema (`schema`);
- tabela (`table`);
- pogled (`view`);
- ...

7.6.5 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 7.98** Napraviti bazu podataka stud.

```
create db stud
   using codeset utf-8 territory sp
   pagesize 32768
```

■

■ **Primer 7.99** Obrisati bazu podataka stud.

```
drop db stud
```

■

7.6.6 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 7.100** Napraviti šemu stud.

```
create schema stud
```

■

■ **Primer 7.101** Obrisati šemu stud.

```
drop schema stud restrict
```

■

■ **Primer 7.102** 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 )
```

■

7.6.7 Pravljenje tabele

Za pravljenje tabela se koristi naredba

```
create table [shema]?.ime-tabele  (
 def-kolone  [, def-kolone]*
 [, def-primarni-ključ]?
 [, def-strani-ključ]*
 [, def-uslov-ogranicenja]* )
```

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 jednobajtne 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 promenjive dužine do *n* jedinica *c*, gde *c* može biti:
    * K
    * M
    * G (n može biti 1-2)
- za dvobajtne 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 defeniciji 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[2]:

- `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 razikuju 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 roiteljskoj 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)
```

7.6.8 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 7.103** 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 7.104** Napraviti korisnički definisan tip država čiji je izvorni tip varchar(30).

```
create distinct type
                drzava as varchar(30)
                with comparisons
```

■

■ **Primer 7.105** Napraviti korisnički definisan tip char10 ciji je izvorni tip varchar(10).

```
create distinct type char10 as char(10)
```

■

■ **Primer 7.106** 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)
)
```

■

7.6.9 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 menajnje 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 korsiti 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 7.107** 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
```

■

7.6.10 Indeksi

Indeks je mehanizam koji ubrzava pristup vrstama tabele i može se posmatrati kao uređeni skup pokazivača na vrste bazne tabele[4]. 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 7.108** Napraviti indeks koji obezbeđuje da ne mogu da postoje dva studenta sa istim imenom.

```
create unique index studime on stud.student(ime)
```

■

7.6.11 Dodatni primeri

■ **Primer 7.109** Napraviti korisnički definisane tipove datum i vreme.

```
create distinct type datum as date with comparisons;
create distinct type vreme as time with comparisons;
```

■

■ **Primer 7.110** 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 7.111** 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 7.112** 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 7.113** 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 7.114** Napraviti tabelu stipendije.

```
create table stipendije  (
 indeks       integer  not null ,
 jmbg         char(15) not null ,
 ovajmesec    stipendija        ,
 proslimesec  stipendija        ,
 narednimesec stipendija        ,
 primary key  (indeks)
)
```

Pošto je prethodno definisan operator za sabiranje (prethodni primer), moguće je izvršiti upit:

```
select indeks, ovajmesec + narednimesec+ proslimesec
from   stipendije
```

■

7.6.12 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 za automatsku izmenu drugih tabela nakon izmene sadržaja ciljne tabele.

Naredba za pravljenje okidača se koristi u obliku

```
create trigger ime-okidaca
vreme-izvrsavanja-akcija dogadjaj 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;
- `dogadjaj` 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 7.115** 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 7.116** 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 sifratriger
before update of sifra on predmet
referencing new as nova
for each row
when (nova.sifra is null)
      set nova.naziv = null;

create trigger sifratriger_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;
```

■

Bibliografija

[1] C.J.Date. *An Introduction to Database Systems*. 8. izdanje. Addison Wesley Inc, 2004.

[2] Saša Malkov. *Skripta za predmet Relacione baze podataka*.

[3] Nenad Mitić. *Materijali za kurs Uvod u relacione baze podataka*. URL: `http://poincare.matf.bg.ac.rs/~nenad/rbp.html`.

[4] Gordana Pavlović-Lažetić. *Osnove relacionih baza podataka*. 2. izdanje. Matematički fakultet, 1999.