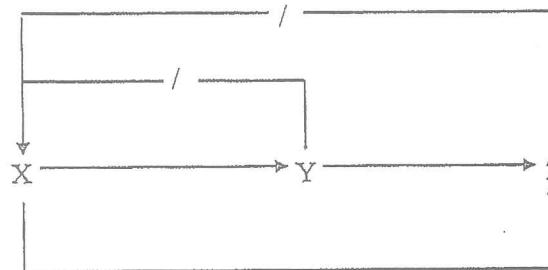


Prethodnog sledi $X \rightarrow Z$, $Z \rightarrow X$, dok $Z \rightarrow Y$ nije niti uslov niti je zabranjeno. Ove zavisnosti se mogu ilustrovati grafički sledećim dijagramom:



Parcijalna FZ je poseban slučaj tranzitivne zavisnosti. Neka je $R(X, A_1, A_2, \dots, A_n)$ relaciona šema. Neka je X' podskup obeležja u R za koji vredi $X' \subset X$, i neka vredi $X \rightarrow A_1, \dots, A_n$ i $X' \rightarrow A_m$. Zbog refleksivnosti (jedan od aksioma za FZ: svaki skup obeležja funkcionalno određuje svaki svoj podskup) vredi $X \rightarrow X'$, a iz zavisnosti $X \rightarrow X'$ i $X' \rightarrow A_m$ proizilazi da je zavisnost $X \rightarrow A_m$ tranzitivna funkcionalna zavisnost.

Relaciona šema R nalazi se u 3NF ako je u 1NF i ako ni jedno neključno obeležje u R nije tranzitivno zavisno od ključa od R.

3NF podrazumeva ispunjenost 1NF i 2NF. Prethodna definicija 3NF eksplicitno zahteva da se relacija nalazi samo u 1NF. S obzirom da je parcijalna funkcionalna zavisnost poseban slučaj tranzitivne zavisnosti, iz uslova da neključna obeležja relacione šeme u 3NF ne smeju biti tranzitivno zavisna od ključa, proizilazi da se relaciona šema u 3NF mora nalaziti i u 2NF.

Neka relaciona šema $R(A_1, A_2, \dots, A_n)$ nije u 3NF. Postoji takva dekompozicija relacione šeme R u skup relacionih šema koje su u 3NF.

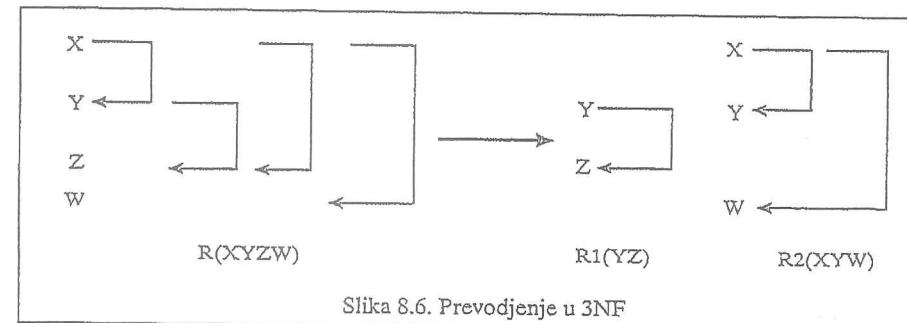
Za relacionu šemu $R(A_1, A_2, \dots, A_n)$ koja nije u 3NF postoje podskupovi X, Y i Z skupa obeležja $\{A_1, A_2, \dots, A_n\}$ takvi da:

- Y i Z nisu ključna obeležja,
- X je kandidat za ključ i vredi $X \rightarrow Y$ i $Y \rightarrow X$,
- $Y \rightarrow Z$ je potpuna FZ.

Neka je W skup svih obeležja šeme relacije R koji nisu ni u X ni u Y ni u Z.

Šemu relacije $R(A_1, A_2, \dots, A_n) = R(XYZW)$ dekomponujemo (zamenjujemo) na šeme relacija $R1(YZ)$ i $R2(XYW)$.

Prevodjenje šeme relacije $R(XYZW)$ u 3NF možemo ilustrovati grafički slikom 8.6.



Slika 8.6. Prevodjenje u 3NF

Relacione šeme $R1(YZ)$ i $R2(XYW)$ ispunjavaju uslove dekompozicije bez gubitka informacija.

Ukoliko relaciona šema $R1(YZ)$ ili $R2(XYW)$ ili obe ne ispunjavaju uslove 3NF postupak normalizacije se na njima nastavlja.

Primer: Šemu relacije $NFM(S_NAS, PREZIME_IME, FAKULTET, MESTO)$ u kojoj vrede sledeće funkcionalne zavisnosti:

$$\begin{array}{lcl} S_NAS & \rightarrow & FAKULTET, \\ FAKULTET & \rightarrow & MESTO, \\ MESTO & \rightarrow & FAKULTET, \\ S_NAS & \rightarrow & MESTO \text{ tranzitivna FZ.} \end{array}$$

prevesti na oblik koji zadovoljava uslove 3NF.

Šema relacije $NFM(S_NAS, PREZIME_IME, FAKULTET, MESTO)$ ispunjava uslove 1NF i 2NF, ali ne ispunjava uslove 3NF.

Na osnovu datog načina za dekompoziciju šeme relacije koja nije u 3NF, šemu $NFM(S_NAS, PREZIME_IME, FAKULTET, MESTO)$ dekomponujemo na relacione šeme $NF(S_NAS, PREZIME_IME, FAKULTET)$ i $FM(FAKULTET, MESTO)$. Šeme relacija $NF(S_NAS, PREZIME_IME,$

FAKULTET) i FM(FAKULTET, MESTO) ispunjavaju uslove 3NF čime je postupak svodjenja na 3NF završen.

Primer: Prevesti na oblik koji zadovoljava uslove 3NF šemu relacije

$$R(JMBG_DETETA, IME_DETETA, JMBG_MAJKE, IME_MAJKE, JMBG_OCA, IME_OCA, MESTO_BORAVKA_OCA, POSTA_BROJ).$$

u kojoj pored zavisnosti od ključa JMBG_DETETA vrede i sledeće funkcionalne zavisnosti:

$$\begin{aligned} JMBG_OCA &\rightarrow IME_OCA \\ JMBG_MAJKE &\rightarrow IME_MAJKE \end{aligned}$$

Obeležja IME_OCA, MESTO_BORAVKA_OCA, POSTA_BROJ, i IME_MAJKE su tranzitivno zavisna od ključa te prema tome relacija R ne ispunjava uslov 3NF.

Postoje sledeće dve tranzitivne zavisnosti:

$$\begin{aligned} JMBG_DETETA &\rightarrow JMBG_MAJKE \rightarrow IME_MAJKE, \\ JMBG_DETETA &\rightarrow JMBG_OCA \rightarrow IME_OCA \\ &\quad MESTO_BORAVKA_OCA \\ &\quad POSTA_BROJ. \end{aligned}$$

Ako dekompoziciju relacione šeme R izvršimo uvažavajući drugu tranzitivnu zavisnost dobijamo sledeće šeme relacija:

$$\begin{aligned} R1(JMBG_OCA, IME_OCA, MESTO_BORAVKA_OCA, POSTA_BROJ), \\ R2(JMBG_DETETA, IME_DETETA, JMBG_OCA, JMBG_MAJKE, IME_MAJKE). \end{aligned}$$

Relaciona šema R1 nije u 3NF jer poseduje sledeću tranzitivnu zavisnost:
 $JMBG_OCA \rightarrow POSTA_BROJ \rightarrow MESTO_BORAVKA_OCA$.

Relacionu šemu R1 dekomponujemo na sledeće dve:

$$\begin{aligned} R11(POSTA_BROJ, MESTO_BORAVKA_OCA), \\ R12(JMBG_OCA, IME_OCA, POSTA_BROJ), \end{aligned}$$

Glava 8. Relacioni model podataka

Relacione šeme R11 i R12 ispunjavaju uslove 3NF.

Relaciona šema R2 nije u 3NF jer poseduje sledeću tranzitivnu zavisnost:

$$JMBG_DETETA \rightarrow JMBG_MAJKE \rightarrow IME_MAJKE.$$

Relacionu šemu R2 dekomponujemo na sledeće dve:

$$R21(JMBG_MAJKE, IME_MAJKE).$$

$$R22(JMBG_DETETA, IME_DETETA, JMBG_OCA, JMBG_MAJKE),$$

Relacione šeme R21 i R22 ispunjavaju uslove 3NF.

U ovom primeru polazeći od relacione šeme R i prve tranzitivne zavisnosti ($JMBG_DETETA \rightarrow JMBG_MAJKE \rightarrow IME_MAJKE$) postupkom normalizacije došli bi do istog rezultata.

Rezime postupka normalizacije

Relacija čije sve domene sadrže samo atomarne vrednosti kažemo da je normalizovana, odnosno da se nalazi u 1NF. Prevodjenje relacione šeme u 2NF znači eliminisanje neključnih obeležja koja su funkcionalno zavisna od dela primarnog ključa. Konačno, prevodjenjem relacione šeme u 3NF iz šeme su izbačena sva neključna obeležja koja su bila tranzitivno zavisna od ključa.

Neključna obeležja relacione šeme koja je u 3NF ispunjavaju sledeće uslove:

- funkcionalno su zavisna od primarnog ključa šeme,
- nisu funkcionalno zavisna od podskupa obeležja šeme koji nije kandidat za ključ odnosno nisu tranzitivno zavisni od primarnog ključa.

Neka je R relaciona šema, i neka su X i Y podskupovi obeležja u R s tim da je X ključ od R, a Y bilo koji podskup obeležja. Neka je A bilo koje neključno obeležje. S obzirom na to da je X ključ u R mora vredeti:

$$X \rightarrow Y$$

$$X \rightarrow A.$$

R se nalazi u 3NF ako vredi:

- a) $Y \rightarrow A$, ili
 b) $Y \rightarrow X$.

U slučaju da vredi a) niti jedno neključno obeležje nije funkcionalno zavisno o bilo kom podskupu obeležja Y , koji nije ključ relacione šeme, i šema se nalazi u 3NF.

Ako vredi b), podskup obeležja Y je kandidat za ključ relacione šeme, i šema se nalazi u 3NF.

Ako Y nije kandidat za ključ od R i u R vredi $Y \rightarrow A$, R nije u 3NF. Pri tome se javljaju sledeće dve mogućnosti:

- 1) $Y \subset X$, ili
- 2) $Y \cap X = \emptyset$.

U slučaju 1) obeležje A je parcijalno funkcionalno zavisno od ključa relacione šeme R . U slučaju 2) obeležje A je tranzitivno zavisno od ključa relacione šeme R .

Transformacija relacione šeme u 3NF ima veliki značaj za praksu, relativno je jednostavna i uvek izvodiva uz postizanje većine ciljeva normalizacije.

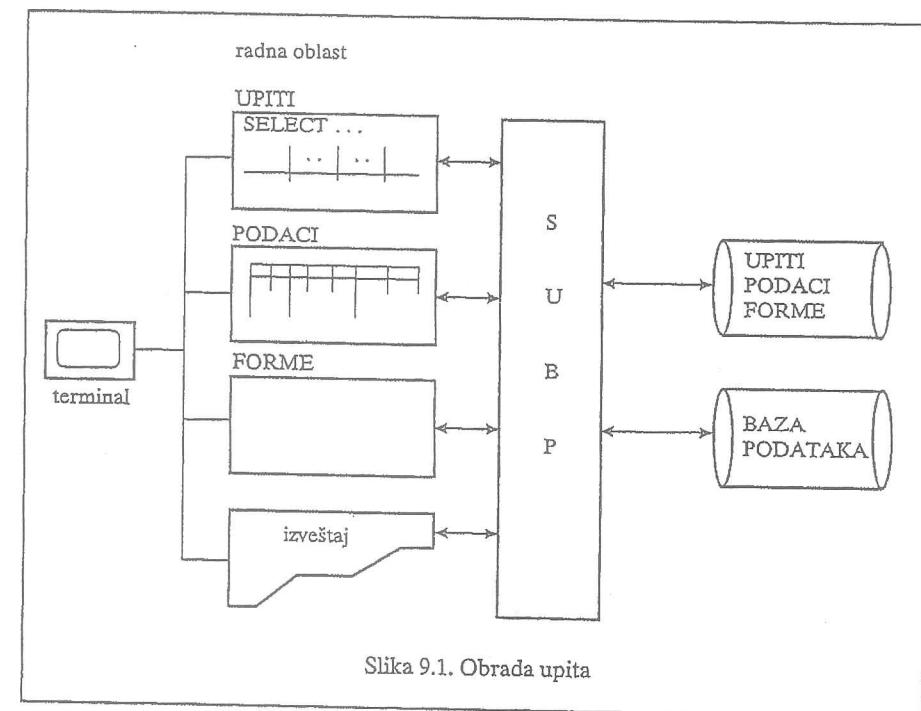
Kao nedostatak postupka vertikalne normalizacije dekompozicijom možemo navesti to da broj dobijenih šema relacija koje zadovoljavaju uslove 3NF ne mora biti najmanje moguć za datu nenormalizovanu šemu relacije. Za dobijeno rešenje ne znamo da li je optimalno ili postoji rešenje sa manjim brojem relacionih šema u 3NF. Ukupan broj relacionih šema u 3NF zavisi o prepostavljenom vlasniku nenormalizovane šeme relacije i izboru ključa. Kada dobijeni broj šema relacija nije optimalan to se može negativno odraziti na troškove održavanja i korišćenja baze podataka.

Kao drugi problem pomenimo da kod vertikalne normalizacije dekompozicijom može doći do gubitka zavisnosti što se može negativno odraziti na održavanje baze podataka.

9. JEZICI ZA MANIPULISANJE PODACIMA U RELACIONOM MODELU

9.1. Obrada upita

Kako bi korisnici mogli brzo i efikasno da koriste bazu podataka u SUBP se obično nalazi jedan skup programa namenjen obradi upita. Tipičan interaktivni način rada za obradu upita ilustrovan je na slici 9.1, a sastoji se iz sledećih koraka:



1. Koristeći neki od jezika za definisanje upita korisnik definiše upit. Takav upit memoriše se u privremenoj radnoj oblasti nazvanoj UPITI.
2. Zahtevanje izvršavanja upita iz oblasti UPITI. Rezultati upita memorišu se u drugoj privremenoj radnoj oblasti nazvanoj PODACI.
3. SUBP formira takozvanu *sistemsku formu* koju memoriše u privremenoj radnoj oblasti FORME i koristi za prikaz podataka iz oblasti PODACI.
4. Posle pregledanja rezultata iz prethodnog koraka korisnik može na ekran pozvati formu i izvršiti njenu redakciju.
5. Korisnik zahteva prikaz podataka iz oblasti PODACI prema formi definisanoj u oblasti FORME pri čemu nije neophodno ponovo izvršavanje upita.
6. Koraci 4 i 5 ponavljaju se sve dotele dok se ne dobije zadovoljavajući rezultat.
7. Zahtev za štampanjem rezultata.

UPITI, PODACI i FORME iz privremenih radnih memorijskih zona odgovarajućim naredbama mogu biti i trajno memorisani i ponovo naknadno korišćeni.

Upitni jezici za relacioni model podataka dele se u sledeće dve klase:

1. *Algebarski jezici* (zasnovani na relacionoj algebri) kod kojih se upiti izražavaju primenom posebnih operatora nad relacijama.

Primeri algebarskih upitnih jezika su:

ISBL,
ASTRID.

2. *Jezici predikatskog računa* kod kojih se upitima opisuje skup n-torki rezultata navodeći predikate koje n-torce moraju da zadovolje.

Primeri jezika predikatskog računa:

QUEL (koristi se u SUBP INGRES),
SQL (koristi se u više SUBP, npr. ORACLE, DB2),
QBE (koristi se u SUBP DB2).

9.2. Standardni upitni jezik - SQL

SQL (Structured Query Language) je standardni relacioni upitni jezik. Njegov tvorac je Chamberlin, a nastao je 1974 godine u IBM-ovoj istraživačkoj laboratoriji (IBM Research Laboratory) u San Jose, Kalifornija na istom mestu gde je E.F. Codd 1970 godine definisao osnovne koncepte relacionog modela podataka.

SQL nije samo upitni jezik već predstavlja kompletan jezik podataka koji sadrži jezike i za: definiciju podataka, ažuriranje, kontrolu, konzistentnost, konkurentni rad i jezik za održavanje rečnika podataka. Ugradjen je u većinu komercijalno raspoloživih SUBP za personalne računare do velikih računara (npr. DB2, SQL/DS, ORACLE, INGRES, PROGRES). Na žalost, i pored toga što SQL u svom nazivu sadrži reč standardni postoje odredjene razlike u njegovoj realizaciji. No ipak znajući SQL moguće je na gotovo identičan način raditi sa bilo kojim od SUBP. Proizvodjači relacionih SUBP sa drugim upitnim jezicima prinudjeni su zbog svog opstanka na tržištu da u svojim SUBP omoguće i korišćenje SQL. Zbog navedene činjenice o velikim mogućnostima korišćenja u daljem tekstu ovog poglavlja biće opisana struktura i semantika većeg broja naredbi SQL sa napomenom da to ni u kom slučaju ne može u potpunosti da zameni odgovarajuću dokumentaciju konkretnog SUBP koja se odnosi na SQL.

SQL je korisnički orijentisan jezik. Uči se lako i brzo, a prethodno iskustvo u automatskoj obradi podataka nije neophodno. Osnovne karakteristike SQL su:

1. Jednostavnost i jednoobraznost pri korišćenju

Tabela se kreira jednom izvršnom naredbom i odmah po kreiranju spremna je za korišćenje (upisivanje, promena, ukidanje i pretraživanje podataka). Podaci se prikazuju u obliku tabele. Termin *tabela* u SQL-u predstavlja sinonimom za pojam *relacija* pa će se tako i ovde koristiti.

2. Mogućnos interaktivnog i klasičnog programiranja

Koristeći SQL dobijaju se odgovori na trenutno postavljene zahteve ili se SQL blokovi ugradjuju u neki viši programski jezik (FORTRAN, COBOL, PL/I, C, ...).

3. Neproceduralnost (tj. proceduralnost u minimalnom stepenu)

Ni za jedan jezik se ne može reći da je potpuno neproceduralan, već da je proceduralan u većoj ili manjoj meri. SQL-je u velikoj meri neproceduralan jer se njime definiše ŠTA se želi dobiti a ne KAKO se do rezultata dolazi: koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez specifikacije procedure za dobijanje podataka. Pravi smisao mere neproceduralnosti SQL čitalac će shvatiti posle upoznavanja samog jezika.

Za opis naredbi SQL koristimo sintaksnu notaciju sa zagradama sa sledećim elementima:

Ekskluzivna disjunkcija - Od više objekata unutar zagrada, medjusobno razdvojenih uspravnom crtom (| - uobičajeni znak eksluzivne disjunkcije) može se odabratи samo jedan.

Na primer, iskaz alfa ili beta ili gama definišemo na sledeći način:

{alfa | beta | gama}.

Opcioni elementi - su elementi koji se po želji mogu izostaviti tj. njihov broj pojavljivanja je 0 ili 1 puta. Označavaju se navodnjem u uglatim zagradama.

Na primer, definicija:

[alfa | beta | gama]

ukazuje da se na tom mestu može koristiti prazno, a ako se koristi nešto drugo onda to može i mora biti alfa ili beta ili gama.

Ključne reči - su terminalni simboli jezika i imaju ulogu reči separatora u naredbama. U definicijama naredbi napisane su velikim slovima, a pri upotrebi trebaju biti navedene u istom obliku pri čemu pisanje velikim ili malim slovima nema značaja. Ukoliko su navedene u uglatim zagradama mogu se izostaviti. Objekti navedeni velikim slovima sa podvlačenjem označavaju prepostavljenе vrednosti tj. i bez navodjenja odredjeno je navedeno značenje. Reči separatore predstavljaju i '(', ')', ';', ',' i '|'. Uočavamo da znakovi '[', ']', '{', '}' i '[' ne predstavljaju separator te da se nikada ne navode u naredbama.

Na primer, definicija:

```
CREATE TABLE naziv_tabele
  (naziv_obl, tip_podataka [NULL | NOT NULL]
  [,naziv_obl, tip_podataka [NULL | NOT NULL]]...)
  [drugi_parametri];
```

može biti zamenjena naredbom

```
CREATE TABLE NASTAVNIK
  (S_NAS          NUMBER(3)      NOT NULL,
   PREZIME_IME    CHAR(10)       NOT NULL,
   ZVANJE         CHAR(6),
   S_DIR          NUMBER(3),
   DATZAP        DATE,
   PLATA          NUMBER(8,2)     NOT NULL,
   DODATAK        NUMBER(8,2));
```

u kojoj su ključne reči: CREATE, TABLE, NULL, NOT NULL, (,) i ;.

Od konstrukcije [NULL | NOT NULL] može biti navedeno NULL ili NOT NULL, pri čemu ako se ništa ne navede (to je moguće zbog uglatih zagrada) smatra se kao da je navedeno NULL (prepostavljena vrednost označena podvlačenjem).

Minimalno u datom primeru naredba mora glasiti (naredba sa obaveznim elementima):

```
CREATE TABLE naziv_tabele
  (naziv_obl tip_podataka);
```

Promenljive reči - Objekti navedeni malim slovima trebaju biti zamenjeni konkretnim značenjima odabranim od strane korisnika. Skup objekata koji mogu biti imenovani od strane korisnika i pravila za izbor imena su specifičnosti SUBP. Kao što je ranije navedeno preporuka pri izboru imena je da ona podesećaju (asociraju) na značenja iz realnog sveta, čime se olakšava komunikacija i pisanje aplikacije.

Na primer, u definiciji iz prethodnog primera promenljiva naziv-tabele zamenjena je sa NASTAVNIK, obli sa S_NAS, tip_podataka sa NUMBER(13) itd.

Ponavljanje prethodnog objekta nula, jednom ili više puta označava se tako da se iza navedenog objekta napišu tri tačke (...).

Na primer, deo definicije iz prethodnog primera

[,naziv_obl, tip_podataka [NULL | NOT NULL]]...

zamenjen je sa:

PREZIME_IME	CHAR(10)	NOT NULL,
ZVANJE	CHAR(6),	
S_DIR	NUMBER(3),	
DATZAP	DATE,	
PLATA	NUMBER(8,2)	NOT NULL,
DODATAK	NUMBER(8,2)	

U SQL sledeći objekti mogu biti imenovani: tabele, pogledi, sinonimi, kolone, indeksi i korisničke promenljive.

Pravila za izbor imena su:

- dužina imena može biti od 1 do 30 simbola,
- prvi simbol u imenu mora biti slovo,
- ime promenljive sadrži A-Z, a-z, 0-9, _, \$ i #.
- ime promenljive ne sme biti rezervisana (službena) reč u SUBP.

Primeri ispravnih imena:

STUDENT, PREDMET, ul._plata.

Primeri neispravnih imena:

25maj, "zvezada", ULAZ-IZLAZ.

Uobičajeno je da ime promenljive sadrži više od jednog simbola. Pogodno je imena birati tako da ona podsećaju na značenje iz realnog sveta, čime se olakšava pisanje aplikacije.

9.2.1. Naredbe za definisanje podataka

9.2.1.1. Kreiranje nove tabele - CREATE TABLE

Naredba CREATE TABLE se koristi za kreiranje nove tabele (bazna tabela, bazna relacija). Potrebno je navesti naziv tabele i za svako obeležje (kolona tabele) naziv, tip, dužinu i da li su NULL vrednosti dozvoljene.

Opšti oblik naredbe je:

```
CREATE TABLE naziv_tabele
  (naziv_obl, tip_podataka [NULL | NOT NULL]
   [,naziv_obl, tip_podataka [NULL | NOT NULL]]...)
   [drugi_parametri];
```

gde je: naziv_obl - naziv i-tog obeležja tabele,
tip_podataka - tip i dužina podatka obeležja koje prethodi,
drugi_parametri - opis memorijskog prostora, klastera ili upita na osnovu kojeg može biti definisana tabela i preneti podaci.

Pri kreiranju tabele ili izmeni definicije tabele mora se definisati tip podatka svakog obeležja. Osnovni tipovi podataka u SQL su:

karakter CHAR(*dužina*) - alfanumerički podaci (velika ili mala slova, decimalne cifre 0-9 i specijalni znaci kao npr. +, -, \$ itd.). Broj znakova određen je parametrom *dužina*.

datum DATE - za datum postoji više različitih formi pri čemu je prepostavljena (standardna) forma DD-MON-YY.

numeric NUMBER (*dužina*) - za numeričke podatke dužine *dužina*.
NUMBER (*dužina.dec*) - za numeričke podatke dužine *dužina* sa *dec* cifara posle decimalne tačke.

Primer: Kreirati tabele sledećih šema relacija:

NASTAVNIK (S_NAS, PREZIME_IME, ZVANJE, S_DIR,
DATZAP, PLATA, DODATAK),
PREDMET (S_PRED, NAZIV),
PREDAJE (S_NAS, S_PRED, BR_GR).

Rešenje:

```

CREATE TABLE NASTAVNIK
  (S_NAS          NUMBER(3)      NOT NULL,
   PREZIME_IME    CHAR(10)       NOT NULL,
   ZVANJE         CHAR(6),
   S_DIR          NUMBER(3),
   DATZAP        DATE,
   PLATA          NUMBER(8,2)     NOT NULL,
   DODATAK        NUMBER(8,2));
CREATE TABLE PREDMET
  (S_PRED          NUMBER(3)      NOT NULL,
   NAZIV           CHAR(30)       NOT NULL);
CREATE TABLE PREDAJE
  (S_NAS          NUMBER(3)      NOT NULL,
   S_PRED          NUMBER(3)      NOT NULL,
   BR_GR          NUMBER(2)      NOT NULL);

```

Zadata relaciona šema nije optimalna sa stanovišta ažuriranja i izveštavanja jer tabela NASTAVNIK može sadržavati NULL vrednosti nekih obeležja što kao što smo rekli i kao što ćemo videti komplikuje operacijsku komponentu. Bolje rešenje za prethodni primer je da na primer, obeležje DODATAK ima opciju NOT NULL.

S obzirom da će navedena relaciona šema poslužiti za kasniju ilustraciju i drugih naredbi, korišćenjem obeležja DODATAK u tabeli NASTAVNIK sa NULL vrednostima pokazaće nam do kakvih sve problema pri korišćenju dovodi postojanje takvog obeležja.

Za obeležja koja nemaju klauzule NULL unapred definisana vrednost je NULL što znači da pri dodavanju n-torce u tabelu ista ne moraju dobiti vrednost tj. imaju neodredjenu vrednost. Neodredjena vrednost koristi se za predstavljanje nepoznate vrednosti ili neprimenljivog svojstva neke n-torce. Obeležja definisana kao NOT NULL prilikom dodavanja n-torki u tabelu moraju dobiti vrednost.

Može se uočiti veliki nedostatak naredbe CREATE TABLE koji se odnosi na nemogućnost definisanja primarnog ključa tabele. Klauzula NOT NULL ne obezbeđuje jedinstvenost vrednosti nekog obeležja ili grupe obeležja pa ne može zameniti ulogu ključa. U SQL-u uloga ključa se može nadomestiti

formiranjem jedinstvenog indeksa nad kombinacijom obeležja koje čine ključ. Nad jednom tabelom može se definisati više ključeva.

9.2.1.2. Kreiranje pogleda - CREATE VIEW

Pogled je jedan "prozor" kroz koji se vide podaci baze podataka. Pogled nema svoje posebne podatke pa je zato virtualna tabela sa kojom se radi gotovo kao i sa baznom tabelom. Pogled se koristi kao bilo koja druga tabela pri izveštavanju dok pri ažuriranju baze podataka preko pogleda postoje niz ograničenja.

Razlozi za korišćenje pogleda su:

- pojednostavljuje korišćenje baze podataka,
- tajnost - realizuje kontrolu pristupa podacima,
- performanse - čuva se u kompajliranom obliku,
- nezavisnost podataka - menjaju se definicije pogleda, a ne aplikacioni programi koji koriste podatke baze podataka preko pogleda.

Opšti oblik naredbe za kreiranje pogleda je:

```

CREATE VIEW naziv_pogleda
  [(naziv_obl, [ , naziv_obl ]...)]
  AS podupit
  [WITH CHECK OPTION];

```

gde je: naziv_obl - naziv i-tog obeležja (kolone) pogleda,
 podupit - ispravna SELECT naredba bez operatora UNION, bez
 klauzule ORDER BY.

Primer: Kreirati pogled

NASTAVNIK_PL (PREZIME_IME, PLATA)

"kroz koji se vide" podaci o nastavnicima koji imaju zvanje
 DOCENT.

Rešenje:

```
CREATE VIEW NASTAVNIK_PL (PREZIME_IME, PLATA)
AS SELECT PREZIME_IME, PLATA
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

U pogledu NASTAVNIK_PL dva obeležja PREZIME_IME i PLATA odgovaraju obeležjima PREZIME_IME i PLATA bazne tabele NASTAVNIK. Ukoliko imena obeležja pogleda u naredbi CREATE VIEW nisu navedena, tada pogled dobija ista obeležja iz podupita navedenog iza SELECT. U našem primeru to bi bila obeležja: PREZIME_IME i PLATA. Sva imena obeležja pogleda trebaju biti specificirana u sledećim slučajevima:

- a) kada se neko obeležje pogleda dobija primenom standardnih funkcija, aritmetičkog izraza ili konstante, pa prema tome nema obeležja bazne tabele od kojeg bi moglo biti nasledjeno.
- b) kada bi dva ili više obeležja imala isto ime.

Primer: Kreirati pogled R_P (S_PRED, UK_GRUPA) u kojem je obeležje S_PRED, šifra predmeta, a UK_GRUPA predstavlja ukupan broj grupa predavanja.

Rešenje:

```
CREATE VIEW R_P (S_PRED, UK_GRUPA)
AS SELECT S_PRED SUM(BR_GR)
FROM PREDAJE
GROUP BY S_PRED;
```

U navedenom primeru nema takvog obeležja koje bi se moglo naslediti za drugo obeležje pogleda R_P imenovano UK_GRUPA, te su zbog toga imena obeležja pogleda morala biti navedena. Može se uočiti da pogled R_P ne predstavlja prost podskup n-torki tabele PREDAJE već jednu izvedenu statističku tabelu.

S obzirom da se pogled može definisati pomoću bilo kog dozvoljenog podupita i da se upiti mogu definisati nad pogledima, to se i pogled može definisati nad pogledom.

Klauzula WITH CHECK OPTION (sa proverom) ukazuje da se operacije UPDATE i INSERT nad tim pogledom trebaju izvršavati sa proverom uslova definisanog u SELECT klauzuli.

Brisanje definicije pogleda ima sledeći opšti oblik:

```
DROP VIEW naziv_pogleda;
```

Izvršavanjem naredbe DROP VIEW iz baze podataka se briše definicija navedenog pogleda kao i svih pogleda koji su na osnovu navedenog definisani.

U jeziku SQL nema naredbe ALTER VIEW. Izvršavanje naredbi ALTER TABLE kojom se dodaju obeležja bazne tabele nemaju uticaja na do tada definisane poglede nad tom tabelom osim u slučaju da su kolone pogleda dobijene iz svih kolona podupita (SELECT * FROM...) kada pogled više neće funkcionisati korektno.

9.2.1.3. Promena strukture tabele nakon što je kreirana - ALTER TABLE

Naredba ALTER TABLE omogućava da:

- 1) Dodamo s desna nova obeležja postojećoj tabeli;
- 2) Promenimo opis obeležja koje već postoji u tabeli.

Dodavanje s desna novih obeležja (kolona) postojećoj tabeli izvodi se naredbom sledećeg opštег oblika:

```
ALTER TABLE naziv_tabele
ADD (naziv_obi; tip_podatka [NULL | NOT NULL]
[,naziv_obii+1; tip_podatka [NULL | NOT NULL]]...);
```

Primer: U tabelu PREDMET dodati obeležje SEMESTAR.

Rešenje:

```
ALTER TABLE PREDMET
ADD (SEMESTAR NUMBER (2));
```

Sve postojeće n-torce tabelle PREDMET proširuju se sa četvrtim poljem koje ima neodredjenu vrednost (NULL). U naredbi ALTER TABLE specifikacija NOT NULL dozvoljena je samo ako tabela nema nijedne n-torce (prazna tabela). Ako se u tabelu koja sadrži n-torce treba dodati obeležje koje treba biti opisano kao NOT NULL moraju se izvršiti sledeće operacije:

1. Dodati novo obeležje u postojeću tabelu sa klauzulom NULL.
2. Za sve n-torce tabele upisati konkretne (definisane) vrednosti novog obeležja.
3. Izvršiti izmenu definicije sada postojećeg obeležja navodjenjem klauzule NOT NULL.

Izmena definicije postojećeg obeležja tabelle (tip, dužina, NULL, NOT NULL) može se izvršiti naredbom sledećeg opštег oblika:

```
ALTER TABLE naziv_tabele
    MODIFY (naziv_obi, tip_podataka [NULL | NOT NULL]
            [,naziv_obi, tip_podataka [NULL | NOT NULL]]...);
```

Primer: U tabeli NASTAVNIK povećati broj znakova obeležja PREZIME_IME na 35 znakova.

Rešenje:

```
ALTER TABLE NASTAVNIK
    MODIFY (PREZIME_IME CHAR (35));
```

9.2.1.4. Brisanje tabelle iz baze podataka - **DROP TABLE**

Brisanje definicije tabelle, zajedno sa podacima koje sadrži može se uraditi naredbom DROP TABLE sledećeg oblika:

```
DROP TABLE naziv_tabele;
```

Brišanjem definicije tabelle automatski se brišu definicije indeksa definisanih nad tom tabelom, dok se definicije pogleda ne brišu ali postaju neupotrebljive. Definicije pogleda možemo takodje brisati ili ih redefinisati.

Primer: Brisati sledeće definicije tabela i same tabele iz baze podataka: NASTAVNIK, PREDMET i PREDAJE.

Rešenje:

```
DROP TABLE NASTAVNIK;
DROP TABLE PREDMET;
DROP TABLE PREDAJE;
```

9.2.1.5. Indeksi

Indeksi se koriste iz sledećih razloga:

- Obezbedjivanje bržeg pristupa po kolonama koje se indeksiraju bez čitanja cele tabelle,
- Obezbedjivanje jedinstvenosti vrednosti kolona koje čine indeks (kolone koje čine indeks mogu imati ulogu primarnog ključa).

Opšti oblik naredbe za kreiranje indeksa je:

```
CREATE [UNIQUE] INDEX naziv_indeksa
    ON naziv_tabele (naziv_obi, [ASC | DESC]
                      [,naziv_obi, [ASC | DESC]]...)
    [drugi_parametri];
```

Po jednoj tabeli može se kreirati više indeksa. Svaki indeks povećava vreme potrebno za održavanje tabelle. Svaki indeks mora imati jedinstveno ime. Klauzule ASC | DESC određuju rastući ili opadajući redosled uredjenja indeksa. Klauzula UNIQUE određuje da tabela ne može sadržavati dve n-torce sa istim sadržajem obeležja koja čine indeks. Ovim mehanizmom se može obezrediti uloga primarnog ključa. Ako klauzula UNIQUE nije navedena mogu postojati dve ili više n-torki sa istim vrednostima indeksa.

Posle izdavanja naredbe za kreiranje indeksa on se automatski održava. Zahtev za kreiranja UNIQUE indeksa nad tabelom koja sadrži podatke koji ne ispunjavaju uslov jedinstvenosti neće biti izvršen.

Indeks se briše naredbom:

```
DROP INDEX naziv_indeksa;
```

Primer: Kreirati indeks nad tabelom NASTAVNIK po obeležju PREZIME_IME.

Rešenje: CREATE INDEX ABC_IME
ON NASTAVNIK (PREZIME_IME);

Indeks ABC_IME omogućava pristup podacima tabele NASTAVNIK po abecedi obeležja PREZIME_IME. Mogu postojati više nastavnika sa istim prezimenom i imenom.

Primer: Kreirati indeks tabele NASTAVNIK koji može imati ulogu primarnog ključa.

Rešenje: CREATE UNIQUE INDEX NASTAVNIK_KEY
ON NASTAVNIK (S_NAS);

9.2.2. Naredbe za manipulisanje podacima

9.2.2.1. Dodavanje novih n-torki u tabelu - *INSERT*

Opšti oblik INSERT naredbe je:

```
INSERT INTO naziv_tabele [(naziv_obl, [, naziv_obl2]...)]  
{VALUES (vrednost_obl, [,vrednost_obl2]...) | podupit};
```

gde se podupitom selektuju n-torki koje će biti upisane u tabelu.

Razmotrimo sledeće tipove insert naredbi:

1. Unos vrednosti za *sva* obeležja jedne n-torke,
2. Unos vrednosti *nekih* obeležja jedne n-torke,
3. Prepisivanje podataka iz jedne tabele u drugu.

1. Naredba INSERT za unos vrednosti *svih* obeležja jedne n-torke tabele ima sledeći opšti oblik:

```
INSERT INTO naziv_tabele  
VALUES (vrednost_obl, [,vrednost_obl2]...);
```

Za svako obeležje tabele definisano naredbom CREATE ili ALTER mora biti upisana vrednost, pri čemu je NULL dozvoljena opcija za svako obeležje koje nije NOT NULL. U listi vrednosti obeležja, vrednost_obi odgovara i-tom obeležju u listi obeležja tabele.

Primer: Potrebno je dodati podatke u tabelu NASTAVNIK za PETRA PETROVIĆA sa šifrom 002, sa zvanjem DOCENTA koji radi od 1 februara 1982, prima platu od 11500 dinara, nema dodatka i kome je rukovodilac nastavnik sa šifrom 001.

Rešenje:
INSERT INTO NASTAVNIK VALUES (002,
'PETROVIĆ PETAR', 'DOCENT', 001,'01-FEB-82', 11500,NULL);

Primer: Dodati u tabelu NASTAVNIK i sledeće podatke.

```
INSERT INTO NASTAVNIK VALUES (003, 'PEIĆ PETAR',  
'DOCENT',005,'01-MAR-83',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (004,'SIMIĆ SIMA',  
'DOCENT',005,'01-APR-84',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (005,'ILIĆ JOVAN',  
'DOCENT',005,'01-MAY-85',11500,NULL);  
INSERT INTO NASTAVNIK VALUES (006,'SAVIĆ ILIJA',  
'V PROF',001,'01-JUN-75',12500,1000);  
INSERT INTO NASTAVNIK VALUES (007,'TOT ANA',  
'V PROF',001,'01-AUG-75',12500,1000);  
INSERT INTO NASTAVNIK VALUES (008,'SAVIĆ MILAN',  
'R PROF',001,'01-SEP-65',13500,2000);  
INSERT INTO NASTAVNIK VALUES (001,'RADOVIĆ NIKOLA',  
'R PROF',NULL,'01-OCT-60',14500,4000);
```

Primer: Dodati u tabelu PREDMET sledeće podatke.

S_PRED	NAZIV	SEMESTAR
001	INFORMACIONI SISTEMI	08
002	STRUKTURE I BP	03
003	OSNOVE RAČUNARSTVA	01
004	TEHNIKE PROGRAMIRANJA	08
005	PROGRAMIRANJE RS	02

Rešenje:

```
INSERT INTO PREDMET
    VALUES (001, 'INFORMACIONI SISTEMI', 08);
INSERT INTO PREDMET
    VALUES (002, 'STRUKTURE I BP', 03);
INSERT INTO PREDMET
    VALUES (003, 'OSNOVE RAČUNARSTVA', 01);
INSERT INTO PREDMET
    VALUES (004, 'TEHNIKE PROGRAMIRANJA', 08);
INSERT INTO PREDMET
    VALUES (005, 'PROGRAMIRANJE RS', 02);
```

Primer: Dodati u tabelu PREDAJE podatke koji odgovaraju sledećem stanju:

PREZIME_IME	NAZIV	BR_GR
RADOVIĆ NIKOLA	Osnove računarstva	2
RADOVIĆ NILOLA	Programiranje RS	1
PETROVIĆ PETAR	Informacioni sistemi	1
PEIĆ PETAR	Programiranje RS	1
ILIĆ JOVAN	Strukture i BP	1

Rešenje:

```
INSERT INTO PREDAJE VALUES (001,003,2);
INSERT INTO PREDAJE VALUES (001,005,1);
INSERT INTO PREDAJE VALUES (002,001,1);
INSERT INTO PREDAJE VALUES (003,005,1);
INSERT INTO PREDAJE VALUES (005,002,1);
```

Ukoliko želimo da unesemo vrednosti za samo neka obeležja jedne n-torke tabele koristi se naredba INSERT sledećeg opštег oblika:

```
INSERT INTO naziv_tabele (naziv_obl1, [naziv_obl2]...)
    VALUES (vrednost_obl1, [vrednost_obl2]...);
```

U listi vrednosti obeležja, vrednost_obl_i odgovara i-tom obeležju iz liste naziva obeležja. Uočimo da se sva NOT NULL obeležja moraju nalaziti u listi

naziva obeležja, što znači da se u listi vrednosti moraju naći vrednosti svih NOT NULL obeležja. Redosled obeležja s leva na desno ne mora odgovarati redosledu obeležja navedenih u naredbi CREATE (ili ALTER).

Primer: Potrebno je dodati u tabelu NASTAVNIK podatke za PETRIĆ JANKA sa šifrom 009, sa zvanjem redovni profesor koji radi od 1 novembra 1961 godine, ima platu od 13500 dinara, dodatak i njegov rukovodilac još nisu određeni.

Rešenje:

```
INSERT INTO NASTAVNIK
    (S_NAS, PREZIME_IME, DATZAP, ZVANJE, PLATA)
    VALUES (009, 'PETRIĆ JANKO', '01-NOV-61', 'R PROF', 13500);
```

Prepisivanje podataka iz jedne tabele u drugu izvodi se naredbom sledećeg opštег oblika:

```
INSERT INTO naziv_tabele [(naziv_obl1, naziv_obl2...)]
    podupit;
```

gde podupit definiše n-torke koje se selektiraju i dodaju u tabelu (naredba SELECT);

9.2.2.2. Pretraživanje relacione baze podataka - SELECT

Opšti oblik naredbe SELECT u SQL je:

```
SELECT [ALL | DISTINCT] { * | {naziv_tabele.*} | element_selekcijske } [alias]
    [{,naziv_tabele.*} | element_selekcijske ] [alias] ... }
    FROM naziv_tabele [alias] [, naziv_tabele [alias]] ...
    [WHERE uslov]
    [CONNECT BY uslov [START WITH uslov]]
    [GROUP BY naziv_obl1, [naziv_obl2]... [HAVING uslov]]
    [{UNION | INTERSECT | MINUS} SELECT ...]
    [ORDER BY { naziv_obl1, pozicija} [ASC|DESC]
        [, {naziv_obl2} pozicija] [ASC|DESC]]...]
    [FOR UPDATE OF kolona, [,kolona2] ... [NOWAIT]];
```

gde je:	naziv_tabele	- naziv tabele ili naziv pogleda,
	element_selekcijske	- obeležje, funkcija, izraz,
	alias	- privremeno ime (sinonim, drugo ime),
	naziv_obeležja	- obeležje tabele,
	uslov	- validan uslov,
	pozicija	- relativna pozicija kolone u SELECT listi,
	kolona	- kolona tabele iz FROM klauzule.

Prepostavljena vrednost ALL će kao rezultat dati sve n-torce koje su rezultat pretraživanja. Klauzulom DISTINCT iz prikaza će biti isključene identične n-torce.

Svaki *element-selekcijske* je ime jedne kolone u prikazu rezultata, dok *tabela.** postaje skup kolona (po jedna kolona za svaku kolonu iz *tabele*) u redosledu koji je definisan pri kreiranju te tabele.

Karakter * koji se može pojaviti samostalno je ekvivalentan sa *tabela.** za svaku tabelu koja se pojavljuje iza FROM u redosledu navodjenja.

FROM *naziv_tabele* definiše tabelu ili pogled iz koje se pretražuju podaci. Ako je navedeno više od jednog naziva tabele mora se izvršiti operacija spoj.

WHERE definiše *uslov* za pretraživanje n-torki i/ili spoj tabela.

CONNECT BY se koristi za prikaz rezultata kada su n-torce tabele strukturirane u stablo (kada jedna n-torka referencira drugu n-torku u istoj tabeli).

GROUP BY i HAVING se koristi za prikaz sumarnih rezultata grupe n-torki koje imaju istu vrednost u jednom ili više obeležja.

Operatori {UNION | INTERSECT | MINUS} SELECT ... se koristi za kombinovanje rezultata dve SELECT naredbe u jednu tabelu.

ORDER BY definiše redosled n-torki rezultata.

FOR UPDATE OF se koristi za zaključavanje selektiranih n-torki tabele. Naredbu SELECT ... FOR UPDATE OF normalno sledi jedna ili više naredbi UPDATE, INSERT ili DELETE. Sve do naredbe COMMIT ili ROLLBACK ostali korisnici nemaju pristupa zaključanim n-torkama tabele.

Obavezni elementi SELECT naredbe su:

```
SELECT {*} | {naziv_tabele.* | element_selekcijske}
      FROM naziv_tabele;
```

Rezultat operacije SELECT je nova tabela koja se na neki način formira iz zadatih tabela baze podataka, ali koja nema svog imena. S obzirom na to moguće je primeniti drugu operaciju SELECT na rezultat jedne operacije SELECT. To opet znači da operacije SELECT mogu biti uložene (ugradjene) jedna u drugu.

9.2.2.2.1. Pretraživanje jedne tabele uz prikaz neizmenjenog sadržaja

Prikaz kompletног sadržaja tabele - SELECT *

Primer: Prikazati kompletan sadržaj tabele: PREDMET, NASTAVNIK, i PREDAJE.

Rešenje:

```
SELECT *
      FROM PREDMET;
```

PREDMET.NAZIV	SEMESTAR
1 INFORMACIONI SISTEMI	8
2 STRUKTURE I BP	3
3 OSNOVE RACUNARSTVA	1
4 TEHNIKE PROGRAMIRANJA	8
5 PROGRAMIRANJERS	2

```
SELECT *
      FROM NASTAVNIK;
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
2	PETROVIĆ PETAR	DOCENT	6	01-FEB-82	11500	
3	PEIĆ PETAR	DOCENT	6	01-MAR-83	11500	
4	SIMIĆ SIMA	DOCENT	6	01-APR-84	11500	
5	Ilić Jovan	DOCENT	6	01-MAJ-85	11500	
6	SAVIĆ ILIJA	V PROF	1	01-JUN-75	12500	1000
7	TOT ANA	V PROF	1	01-AUG-75	12500	1000
8	SAVIĆ MILAN	R PROF	1	01-SEP-65	13500	2000
1	RADOVIĆ NIKOLA	R PROF		01-NOV-61	14500	4000
9	PETRIĆ JANKO	R PROF		01-OCT-60	13500	

SELECT * FROM PREDJAE;

S_NAS	S_PRED	BR_GR
1	3	2
1	5	1
2	1	1
3	5	1
5	2	1

Selekcija kolona

Primer: Prikazati S_NAS i PREZIME_IME n-torki iz tabele NASTAVNIK.

Rešenje: SELECT S_NAS, PREZIME_IME FROM NASTAVNIK;

S_NAS	PREZIME_IME
2	PETROVIĆ PETAR
3	PEIĆ PETAR
4	SIMIĆ SIMA
5	Ilić Jovan
6	SAVIĆ ILIJA
7	TOT ANA
8	SAVIĆ MILAN
9	PETRIĆ JANKO
1	RADOVIĆ NIKOLA

Kolone rezultata pojavljuju se u redosledu definisanom u SELECT naredbi. Redosled n-torki rezultata određuje SUBP.

Selekcija n-torki - klauzula WHERE

U prethodnom primeru smo videli kako vršimo selekciju kolona tabele. Za selekciju n-torki tabele potrebno je uključiti WHERE klauzulu.

Korišćena u SELECT naredbi WHERE klauzula omogućava:

1. Selekciju odredjenih n-torki tabele,
2. Selekciju n-torki koje zadovoljavaju višestruke uslove,
3. Selekciju n-torki koje zadovoljavaju bar jedan od više uslova,
4. Selekciju n-torki koje ne zadovoljavaju odredjene uslove,
5. Selekciju n-torki korišćenjem AND i OR logičkih operatora,
6. Selekciju n-torki unutar definisanog raspona,
7. Selekciju n-torki koje zadovoljavaju vrednosti u listi vrednosti,
8. Selekciju n-torki koje sadrže određenu kombinaciju karaktera.

Za izražavanje potrebnih uslova selekcije n-torki u WHERE klauzuli mogu se koristiti sledeći operatori:

- | | |
|----------|---|
| = | jednako, |
| != | nije jednako, |
| >>= < <= | veće, veće ili jednako, manje, manje ili jednako, |
| NOT IN | nije jednako nijednoj vrednosti što je ekvivalentno sa ' != ALL ', |
| IN | jednako sa nekom vrednošću što je ekvivalentno sa ' = ANY ', |
| ANY | poredjenje vrednosti sa svakom dobijenom vrednošću rezultata, |
| ALL | poredjenje vrednosti sa svim dobijenim vrednostima rezultata, |

[NOT] BETWEEN x AND y [Not] veće ili jednako od x i manje ili jednako od y.

EXISTS istinito ako se kao rezultat dobije bar jedna n-torka,
[NOT] LIKE slaganje ili ne slaganje sekvence karaktera koja sledi.
'%' slaganje bilo koje sekvene karaktera,
'_' slaganje jednog karaktera,

IS [NOT] NULL je [nije] nul vrednost,
NOT negacija logičkog rezultata,
AND logičko I,
OR logičko ILI,

Primer: Prikazati sve predmete koji se predaju u 8 semestru:

Rešenje:

```
SELECT *
  FROM PREDMET
 WHERE SEMESTAR = 8;
```

S_PRED_NAZIV	SEMESTAR
1 INFORMACIONI SISTEMI	8
4 TEHNIKE PROGRAMIRANJA	8

Izražavanje uslova pomoću raspona podataka - BETWEEN

Operator BETWEEN omogućava selekciju n-torki sa sadržajem koji pripada definisanom rasponu podataka.

Primer: Prikazati sve nastavnike koji imaju platu izmedju 11000 i 12500.

Rešenje:

```
SELECT PREZIME_IME, PLATA
  FROM NASTAVNIK
 WHERE PLATA BETWEEN 11000 AND 12500;
```

PREZIME_IME	PLATA
PETROVIĆ PETAR	11500
PEIĆ PETAR	11500
SIMIĆ SIMA	11500
ILIĆ JOVAN	11500
SAVIĆ ILIJA	12500
TOT ANA	12500

Prethodni upit realizuje i sledeća naredba:

```
SELECT PREZIME_IME, PLATA
  FROM NASTAVNIK
 WHERE PLATA >= 11000 AND PLATA <= 12500;
```

Pretraživanje za vrednosti u listi - IN

Operator IN omogućava izbor n-torki koje sadrže vrednost koja je jednaka jednoj od vrednosti navedenih u listi.

Primer: Prikazati sve predmete čija je šifra 2 ili 3.

Rešenje:

```
SELECT *
  FROM PREDMET
 WHERE S_PRED IN (2, 3);
```

S_PRED_NAZIV	SEMESTAR
2 STRUKTURE I BP	3
3 OSNOVE RACUNARSTVA	1

Prethodni upit realizuje i sledeća naredba:

```
SELECT *
  FROM PREDMET
 WHERE S_PRED = 2 OR S_PRED = 3;
```

Operator LIKE

Operator LIKE omogućava izbor n-torki koje imaju parcijalno definisan sadržaj određenog obeležja (odredjenu kombinaciju karaktera).

Primer: Prikazati prezime i ime svih nastavnika koji kao treće slovo u prezimenu imaju "I".

Rešenje:

```
SELECT PREZIME_IME
  FROM NASTAVNIK
 WHERE PREZIME_IME LIKE '_ _ I %';
```

PREZIME_IME
PEIĆ PETAR
ILIĆ JOVAN

Karakter _ označava jedan karakter (bilo koji), dok % označava bilo koji niz karaktera.

Operatorima BETWEEN, IN i LIKE može prethoditi reč NOT (negacija). Isti mogu biti povezani sa AND i OR operatorima u izgradnji kompleksnih izraza za pretraživanje.

Definisanje redosleda n-torki rezultata pretraživanja - ORDER BY

U prethodnim primerima SELECT naredbe redosled n-torki u rezultatu određen je od strane SUBP. Korišćenjem klauzule ORDER BY na kraju SELECT naredbe možemo kontrolisati redosled n-torki rezultata.

Primer: Prikazati platu, prezime i ime i zvanje nastavnika prema opadajućem iznosu plate.

Rešenje:

```
SELECT PLATA, PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     ORDER BY PLATA DESC;
```

PLATA	S_NAS	PREZIME_IME	ZVANJE
14500	1	RADOVIĆ NIKOLA	R PROF
13500	9	PETRIĆ JANKO	R PROF
13500	8	SAVIC MILAN	R PROF
12500	6	SAVIC ILIJA	V PROF
12500	7	TOT ANA	V PROF
11500	2	PETROVIĆ PETAR	DOCENT
11500	3	PEIĆ PETAR	DOCENT
11500	4	SIMIĆ SIMA	DOCENT
11500	5	ILIĆ JOVAN	DOCENT

Eliminisanje jednakih n-torki u rezultatu - klauzula DISTINCT

Prepostavimo da želimo da prikažemo listu svih zvanja iz tabele NASTAVNIK. Odgovarajuća naredba glasi:

```
SELECT ZVANJE
      FROM NASTAVNIK;
```

ZVANJE

DOCENT

DOCENT

DOCENT

DOCENT

V PROF

V PROF

R PROF

R PROF

R PROF

R PROF

Vidimo u rezultatu više n-torki sa istim sadržajem jer postoji više nastavnika sa istim zvanjem. Ako želimo eliminisati iste n-torke u rezultatu koristimo klauzulu DISTINCT koja znači: *prikaži samo različite n-torke*.

Za prethodni primer:

```
SELECT DISTINCT ZVANJE
      FROM NASTAVNIK;
```

ZVANJE

DOCENT

V PROF

R PROF

9.2.2.2.2. Pretraživanje jedne tabele uz oblikovanje dobijenih podataka

Izrazi i funkcije

Pored prikazivanja vrednosti memorisanih u tabelama (proste vrednosti), SQL ima više funkcija koje mogu biti korišćene za dobijanje sumarnih informacija, kao i mogućnost primene aritmetičkih funkcija i funkcija nad nizovima karaktera (stringovi).

Funkcije za dobijanje sumarnih informacija nad numeričkim kolonama su:

AVG ([ALL | DISTINCT] *obeležje*) - izračunava srednju vrednost,
 SUM ([ALL | DISTINCT] *obeležje*) - izračunava ukupnu vrednost,
 MIN ([ALL | DISTINCT] *izraz*) - nalazi minimalnu vrednost od *izraz*,
 MAX ([ALL | DISTINCT] *izraz*) - nalazi maksimalnu vrednost od *izraz*.

Funkcija COUNT definisana je nad kolonama bilo kog tipa. Ona ima sledeći opšti oblik:

COUNT ({*} | [ALL | DISTINCT] naziv_obeležja).

Iz ovog opšteg oblika mogu nastati sledeći oblici:

COUNT (*) - nalazi broj n-torki,
 COUNT (*izraz*) - nalazi broj n-torki sa NOT NULL vrednostima od *izraz*,
 COUNT (DISTINCT *izraz*) - nalazi broj različitih n-torki sa NOT NULL vrednostima *izraz*,
 COUNT (ALL *izraz*) - nalazi broj svih n-torki sa NOT NULL vrednostima od *izraz*,

Primer: Naći minimalnu, srednju i maksimalnu platu svih nastavnika, kao i ukupan broj nastavnika.

Rešenje:

```
SELECT MIN(PLATA), AVG(PLATA), MAX(PLATA), COUNT(*)
FROM NASTAVNIK;
```

MIN(PLATA)	AVG(PLATA)	MAX(PLATA)	COUNT(*)
11500	12500	14500	9

Primer: Napisati odgovarajuću SQL naredbu za prikaz ukupne plate i dodatka za sve DOCENTE.

Rešenje:

```
SELECT SUM(PLATA), SUM(DODATAK)
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

SUM(PLATA) SUM(DODATAK)

46000

Sumiranje informacija - klauzula GROUP BY

Upotreba klauzule GROUP BY omogućava dobijanje informacija za svaku različitu vrednost obeležja po kojem se vrši grupisanje.

Razmotrimo potrebu dobijanja informacija o srednjoj plati i broju nastavnika svakog pojedinog zvanja. Na osnovu do sada rečenog ovaj upit bi se mogao realizovati izvršavanjem onoliko SQL SELECT naredbi koliko ima različitih zvanja. Pri tome se SELECT i FROM klauzule ne bi menjale, dok bi se uslov u WHERE klauzuli menjao u zavisnosti od zvanja.

Jedna od takvih SQL naredbi čiji je rezultat jedna n-torka, glasi:

```
SELECT ZVANJE, AVG(PLATA), COUNT(*)
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

ZVANJE	AVG(PLATA)	COUNT(*)
DOCENT	11500	4

Informaciju o svim različitim zvanjima dobili bi upitom (jedan od ranijih primera):

```
SELECT DISTINCT ZVANJE
FROM NASTAVNIK;
```

ZVANJE
DOCENT
V PROF
R PROF

Kao rezultat ponavljanja SELECT naredbe za sva zvanja dobili bi ukupno tri n-torce.

Identičan rezultat možemo dobiti samo jednom SQL naredbom (umesto četiri) sledećeg oblika:

```
SELECT ZVANJE, AVG(PLATA), COUNT(*)
  FROM NASTAVNIK
 GROUP BY ZVANJE;
```

ZVANJE	AVG(PLATA)	COUNT(*)
DOCENT	11500	4
V PROF	12500	2
R PROF	13500	3

Dejstvo GROUP BY klauzule je identično ponavljanju SELECT naredbe sa različitim WHERE uslovom.

U SELECT listi za prikaz kolona rezultata pored funkcija može doći samo obeležje koje se nalazi u GROUP BY klauzuli.

Grupisanje n-torki može se vršiti po više obeležja koja se navode u GROUP BY klauzuli.

Uslovi pretraživanja za grupu - klauzula HAVING

Pošto su grupe već formirane sa GROUP BY klauzulom korišćenjem HAVING klauzule definiše se kriterij za selekciju grupa.

Primer: Prikazati u kom zvanju ima više od dva nastavnika.

Rešenje:

```
SELECT ZVANJE, COUNT(*)
  FROM NASTAVNIK
 GROUP BY ZVANJE HAVING COUNT(*) > 2;
```

ZVANJE	COUNT(*)
DOCENT	4
R PROF	3

WHERE i GROUP BY klauzula se mogu koristiti u istoj SELECT naredbi pri čemu WHERE klauzula uvek ide pre GROUP BY klauzule. Sa WHERE klauzulom se prvo definiše uslov selekcije n-torki, zatim se selektovane n-torce grupišu GROUP BY klauzulom, pa se eventualno izvrši selekcija formiranih grupa sa HAVING klauzulom.

SQL naredbe mogu sadržavati aritmetičke izraze sastavljene od imena kolona i konstanti povezanih aritmetičkim operatorima (+, *, -, /). Korišćenjem zagrade u izrazima može se definisati redosled sračunavanja izraza.

Primer: Koliko je srednje godišnje primanje redovnih profesora.

Rešenje:

```
SELECT ZVANJE, AVG(PLATA + DODATAKA) * 12
  FROM NASTAVNIK
 WHERE ZVANJE = 'R PROF'
 GROUP BY ZVANJE;
```

ZVANJE	AVG(PLATA+DODATAK) * 12
R PROF	186000

Aritmetičke funkcije

SQL podržava sledeće aritmetičke funkcije:

ABS(<i>broj</i>)	- apsolutna vrednost od <i>broj</i> ,
MOD(<i>broj₁</i> , <i>broj₂</i>)	- izračunava <i>broj₁</i> moduo <i>broj₂</i> ,
POWER(<i>broj</i> , <i>e</i>)	- diže <i>broj</i> na <i>e</i> -ti stepen,
ROUND(<i>broj</i> [,d])	- zaokružuje <i>broj</i> na d decimalna,
SIGN(<i>broj</i>)	- daje +1 ako je <i>broj</i> > 0, 0 ako je <i>broj</i> = 0, -1 ako je <i>broj</i> < 0,
SQRT(<i>broj</i>)	- izračunava kvadratni koren od <i>broj</i> ,
TRUNC(<i>broj</i> [,d])	- u <i>broj</i> odbacuje ostatak posle d-tog decimalnog mesta,

Primer: Koji nastavnici zaradjuju više od 70 dinara po satu. Zaradu po satu izračunati tako da se ukupni iznos plate i dodatka podeli sa 176 (broj sati u mesecu). U prikazu zaradu po satu zaokruži na ceo broj.

Rešenje:

```
SELECT S_NAS, PREZIME_IME, ROUND((PLATA+DODATAK)/176)
      "ZARADA PO SATU"
  FROM NASTAVNIK
 WHERE (PLATA + DODATAK) / 176 > 70;
```

S_NAS	PREZIME_IME	ZARADA PO SATU
6	SAVIC ILIJA	77
7	TOT ANA	77
8	SAVIC MILAN	88
1	RADOVIĆ NIKOLA	105

Mogli smo do ovog primera zapaziti da se u rezultatu dobijenom SELECT naredbom kolone imenuju kao i navedeni *element selekcije*. Kada se u prikazu rezultata želi drugačiji naziv kolone od naziva elementa selekcije koristi se privremeno ime (sinonim) za prikaz elementa selekcije tako da se navede iza *elementa selekcije* između dva znaka ". Ukoliko se privremeno ime sastoji od jedne reči znakovi " se ne moraju navoditi. U prethodnom primeru može se uočiti da je to učinjeno za treću kolonu rezultata.

Pažljivom proverom tabele NASTAVNIK možemo se zapitati zašto u prethodnom rezultatu nema nastavnika PETRIĆ JANKA koji ima platu 13500, dodatak je NULL vrednost i za kojeg je zarada po satu $13500/176 > 70$. Razlog je u tome što se obeležja sa NULL vrednostima ne koriste pri izračunavanju izraza i funkcija. Da bi se izračunavanje ipak omogućilo možemo koristiti NVL funkciju koja privremeno menja NULL vrednost sa vrednošću za koju se odlučimo, tj. sa vrednošću koja odgovara željenoj operaciji odnosno rezultatu.

Funkcija NVL(*obeležje, broj*) izvršava se na sledeći način:

Ako je vrednost za *obeležje* NULL vrednost ista se privremeno zamenjuje sa *broj*.

Ako je vrednost za *obeležje* definisana tada se uzima ta vrednost.

Razmotrimo sada još jednom prethodni primer sa sledećom naredbom za pretraživanje:

```
SELECT S_NAS, PREZIME_IME, ROUND((PLATA +
          NVL(DODATAK,0)) / 176) "ZARADA PO SATU"
  FROM NASTAVNIK
 WHERE (PLATA + NVL(DODATAK, 0)) / 176 > 70;
```

S_NAS	PREZIME_IME	ZARADA PO SATU
6	SAVIC ILIJA	77
7	TOT ANA	77
8	SAVIC MILAN	88
1	RADOVIĆ NIKOLA	105
9	PETRIC JANKO	77

Uočavamo na prethodnom primeru da za posledicu korišćenja NULL vrednosti uvek treba voditi računa o tome šta predstavlja rezultat aritmetičkog izraza ili funkcije, odnosno koje n-torce su uključene u rezultat, a koje nisu. Zbog toga je i napomenuto pri razmatranju operacijske komponente relacionog modela podataka, da kad god je to moguće treba eliminisati kreiranje baze sa NULL vrednostima.

Funkcije nad nizom karaktera

U SQL je definisan veći broj funkcija nad podacima tipa karakter.

Slede neke od funkcija:

- niz*, || *niz* - spaja nizove karaktera,
- INSTR (*niz, podniz [,pos]*) - traži *podniz* u *niz* od pozicije *pos*. Ako je nadjen vraća njegovu poziciju, inače je 0.
- LENGTH (*niz*) - nalazi dužinu od *niz*,
- LOWER (*niz*) - konvertuje sva velika slova od *niz* u mala,
- LPAD (*niz₁, broj [,niz₂]*) - popunjava levu stranu od *niz₁* sa karakterom *niz₂* u dužini *broj*, ako je *niz₂* izostavljen sa blanko,
- RPAD (*niz₁, broj [,niz₂]*) - popunjava desnu stranu od *niz₁* sa karakterom *niz₂* u dužini *broj*, ako je *niz₂* izostavljen sa blanko,
- SUBSTR (*niz, pos [,duz]*) - daje podniz od *niz* počinjući od pozicije *pos*, dužine *duz*.
- TO_NUM (*niz*) - konvertuje *niz* karaktera (numeričkih) u broj,
- TO_CHAR (*broj*) - konvertuje *broj* u niz karaktera,

TO_CHAR (datum[format]) - konvertuje *datum* u karakter vrednost oblik *format*,

TO_DATE (niz, format) - konvertuje karakter vrednost datuma u datum sa formatom *format*.

UPPER (niz) - konvertuje sva mala slova od *niz* u velika.

Sve ove funkcije navode se iza SELECT naredbe.

Primer: Prikazati prezime ime nastavnika iza kojeg treba neposredno prikazati zvanje. Redosled n-torki rezultata treba da je u rastućem redosledu obeležja PREZIME_IME.

Rešenje:

```
SELECT PREZIME_IME || ',' || ZVANJE NASTAVNIK
      FROM NASTAVNIK
     ORDER BY PREZIME_IME;
```

NASTAVNIK
ILIĆ JOVAN, DOCENT
PEIĆ PETAR, DOCENT
PETRIĆ JANKO, R PROF
PETROVIĆ PETAR, DOCENT
RADOVIĆ NIKOLA, R PROF
SAVIC ILLIJA, V PROF
SAVIĆ MILAN, R PROF
SIMIĆ SIMA, DOCENT
TOT ANA, V PROF

Prethodno pretraživanje spaja tri niza (prvi niz je PREZIME_IME, drugi je zarez i jedno prazno mesto i treći niz je vrednost obeležja ZVANJE) koji se u prikazu rezultata imenuju sa NASTAVNIK.

Prikazivanje i rad sa datumima

Jedan od tipova podataka za definisanje obeležja relacione baze podataka je datum. Standardni oblik datuma izgleda na primer: 15-JAN-93. Ovaj oblik se piše 'DD-MON-YY'. Ako drugačije nije naglašeno prikaz i unos datuma je u ovom obliku.

Ranije pomenutom funkcijom TO_CHAR može se odabratи drugi oblik za prikaz datuma njegovim pretvaranjem u karakter vrednost.

Oblici koji se mogu koristiti za datum su:

OBLIK	PRIMER	OBJAŠNJENJE
(bez formata)	15-JAN-93	standardni format
MM/DD/YY	15/01/93	svi brojevi od dve brojke
DD/MM/YYYY	15.01.1993	"." kao separator, puna godina
Month DD, YYYY	January 15, 1993	mesec napisan slovima,
DY DD MON YY	WED 15 JAN 92	dan i mesec skraćeni i napisani velikim slovima
Day Mon DD	Wednesday Jan 15	dan napisan slovima, nema godine

Primer: Prikazati prezime ime, i datum zaposlenja nastavnika sa zvanjem vanrednog profesora. Datum zaposlenja prikazati u standardnom obliku i oblicima sa punom godinom.

Rešenje:

```
SELECT PREZIME_IME, DATZAP,
       TO_CHAR(DATZAP, 'DD.MM.YYYY') 'DD MM.YYYY',
       TO_CHAR(DATZAP, "Mesec" Month DD, YYYY)
                           'Month DD, YYYY'
    FROM NASTAVNIK
   WHERE ZVANJE = 'V PROF';
```

PREZIME_IME	DATZAP	DDMM.YYYY	Month DD, YYYY
SAVIC ILLIJA	01-JUN-75	01.06.1975	Mesec June 01, 1975
TOT ANA	01-AUG-75	01.08.1975	Mesec August 01, 1975

Sa podacima tipa datum mogu se izvoditi sledeće aritmetičke operacije:

datum + broj - na *datum* se dodaje *broj* dana,

datum - broj - od *datum* se oduzima *broj* dana,

datum₁ - datum₂ - oduzima *datum₂* od *datum₁*, i kao rezultat daje razliku iskazanu brojem dana.

Primer: Prikazati prezime ime i broj godina radnog staža svakog nastavnika.

Rešenje:

```
SELECT PREZIME_IME, (SYSDATE - DATZAP) / 365
      'GODINA STAŽA'
   FROM NASTAVNIK
  ORDER BY PREZIME_IME;
```

NASTAVNIK	GODINA STAŽA
ILIĆ JOVAN	8.39402
PEIĆ PETAR	10.5639
PETRIĆ JANKO	31.9064
PETROVIĆ PETAR	11.6406
RADOVIĆ NIKOLA	32.9913
SAVIĆ ILLJA	18.3173
SAVIĆ MILAN	28.0707
SIMIĆ SIMA	9.47622
TOT ANA	18.1502

U prethodnom primeru SYSDATE je sistemski tekući datum i vreme, a rezultati odgovaraju za 19.09.1993. Godine staža su izračunate za 365 dana u godini čime u razmatranje nisu uključene prestupne godine. Korišćenjem na primer, izraza MONTHS_BETWEEN (SYSDATE, DATZAP) / 12 dobili bi broj godina radnog staža nezavisno od prestupnih godina. Funkcija MONTHS_BETWEEN (SYSDATE, *datum*) daje broj meseci između datum i tekućeg datuma.

9.2.2.2.3. Ulaganje upita nad jednom tabelom u upit nad drugom tabelom

Jedan od načina povezivanja tabela u bazi podataka je ulaganjem upita nad jednom tabelom u upit nad drugom, odnosno dinamičkom zamenom rezultata jednog upita u WHERE klauzuli drugog upita.

Primer: Prikazati prezime ime i zvanje svih nastavnika koji imaju isto zvanje kao i RADOVIĆ NIKOLA.

Korak 1. Utvrdimo koje zvanje ima RADOVIĆ NIKOLA.

```
SELECT ZVANJE
      FROM NASTAVNIK
     WHERE PREZIME_IME = 'RADOVIĆ NIKOLA';
```

ZVANJE
R PROF

Korak 2. Prkažimo prezime ime i zvanje svih nastavnika koji imaju zvanje R PROF.

```
SELECT PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     WHERE ZVANJE = 'R PROF';
```

PREZIME_IME	ZVANJE
RADOVIĆ NIKOLA	R PROF
PETRIĆ JANKO	R PROF
SAVIĆ MILAN	R PROF

Povezivanje tabela dinamičkom zamenom rezultata jednog upita u WHERE klauzuli drugog sastoji se u tome da se prvi upit nadje u WHERE klauzuli drugog.

U prethodnom primeru to izgleda:

```
SELECT PREZIME_IME, ZVANJE
      FROM NASTAVNIK
     WHERE ZVANJE = (SELECT ZVANJE
                      FROM NASTAVNIK
                     WHERE PREZIME_IME = 'RADOVIĆ NIKOLA');
```

Dobijamo isti rezultat kao i posle koraka 2.

Upit iz koraka 1 ili prvi upit navodi se u zagradama drugog upita i naziva se *unutrašnji upit* (uloženi upit). Drugi upit naziva se *spoljašnji upit*.

Prilikom izvršavanja upita sa uloženim upitom prvo se izvršava unutrašnji upit a zatim spoljašnji upit.

Primer: Prikazati prezime ime, zvanje i platu nastavnika sa zvanjem DOCENT koji ne predaju ni jedan predmet.

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
  FROM NASTAVNIK
 WHERE ZVANJE = 'DOCENT' AND
       S_NAS NOT IN (SELECT DISTINCT S_NAS
                      FROM PREDAJE);
```

PREZIME_IME	ZVANJE	PLATA
SIMIC SIMA	DOCENT	11500

Primer: Koji nastavnik u svakom zvanju ima najveću platu ?

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
  FROM NASTAVNIK
 WHERE (ZVANJE, PLATA) IN (SELECT ZVANJE, MAX(PLATA)
                           FROM NASTAVNIK
                           GROUP BY ZVANJE)
 ORDER BY PLATA DESC;
```

PREZIME_IME	ZVANJE	PLATA
RADOVIĆ NIKOLA	R. PROF.	14500
SAVIC ILIJA	V. PROF.	12500
TOTANA	V. PROF.	12500
PETROVIĆ PETAR	DOCENT	11500
SIMIC SIMA	DOCENT	11500
ILIĆ JOVAN	DOCENT	11500

Primer: Prikazati prezime ime nastavnika koji imaju najmanje jednog nastavnika koji im je podredjen.

Rešenje:

```
SELECT PREZIME_IME
  FROM NASTAVNIK A
 WHERE EXISTS (SELECT *
                  FROM NSATAVNICI B
                 WHERE A.S_NAS = B.S_DIR)
 ORDER BY PREZIME_IME;
```

NASTAVNIK
RADOVIĆ NIKOLA SAVIC ILIJA

Prethodni primer pokazuje da spoljašnji i unutrašnji upit mogu biti povezani vrednostima više obeležja koja se u tom slučaju navode u zagradama.

Spoljašnji i unutrašnji upit mogu biti sastavljeni od dva ili više SELECT blokova medjusobno povezanih operatorima UNION, INTERSECT ili MINUS:

- UNION (unija) prikazuje sve rezultujuće n-torke prvog SELECT bloka i one rezultujuće n-torke drugog SELECT bloka koje nisu medju rezultujućim n-torkama prvog SELECT bloka.

- INTERSECT (presek) prikazuje sve rezultujuće n-torke prvog SELECT bloka koje su istovremeno i rezultujuće n-torke drugog SELECT bloka.

- MINUS prikazuje sve rezultujuće n-torke prvog SELECT bloka koje nisu istovremeno i rezultujuće n-torke drugog SELECT bloka.

Da bi navedene operacije mogle biti primenjene rezultati upita SELECT blokova koji učestvuju u operaciji moraju imati isti broj rezultujućih kolona i iste moraju odgovarati po tipu.

Primer: Prikazati prezime ime, zvanje i plate nastavnika sa zvanjem docenta i redovnog profesora.

Rešenje:

```
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'DOCENT'
UNION
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'R PROF';
```

PREZIME_IME	ZVANJE	PLATA
PETROVIC PETAR	DOCENT	11500
PEIĆ PETAR	DOCENT	11500
SIMIC SIMA	DOCENT	11500
ILIC JOVAN	DOCENT	11500
SAVIC MILAN	R PROF	13500
RADOVIC NIKOLA	R PROF	14500
PETRIC JANKO	R PROF	13500

Prethodno pretraživanje se može postići i sa:

```
SELECT PREZIME_IME, ZVANJE, PLATA
FROM NSTAVNIK
WHERE ZVANJE = 'DOCENT' OR ZVANJE = 'R PROF';
```

Kao što i prethodni primer pokazuje primena operacija UNION, INTERSECT i MINUS nemaju puno smisla nad jednom tabelom, jer se isti rezultati mogu dobiti korišćenjem logičkih operatora AND, OR i NOT.

9.2.2.2.4. Povezivanje više tabела

U svim prethodnim primerima SELECT naredbe rezultat pretraživanja formiran je iz jedne tabele. Kako doći do rezultata pretraživanja koji nije smešten u jednoj tabeli? Povezivanje dve ili više tabele omogućava kombinovanje podataka sa prikazom rezultata u jednoj tabeli.

Prepostavimo da želimo da znamo prezime i ime nastavnika koji predaje predmet sa šifrom 1. Posmatrajući naše tabele NASTAVNIK, PREDMET i PREDAJE vidimo da do željenog rezultata možemo doći u sledeća dva koraka:

Korak 1. Pretraživanjem tabele PREDAJE dobit ćemo šifru nastavnika koji predaje predmet sa šifrom 1.

```
SELECT S_PRED, S_NAS
FROM PREDAJE
WHERE S_PRED = 1;
```

S_PRED	S_NAS
1	2

Korak 2. Sada pretraživanjem tabele NASTAVNIK za šifru nastavnika 2 dobijemo prezime i ime nastavnika.

```
SELECT S_NAS, PREZIME_IME
FROM NASTAVNIK
WHERE S_NAS = 2;
```

S_NAS	PREZIME_IME
2	PETROVIC PETAR

Kao rezultat prethodna dva upita dobili smo odgovor da PETROVIĆ PETAR predaje predmet sa šifrom 1.

Do istog rezultata možemo doći koristeći umesto dva upita samo jedan. Uopšteno rečeno, da bi se izvršilo pretraživanje u kojem rezultat zavisi od sadržaja više tabele moraju se u FROM klauzuli navesti tabele koje treba pretražiti, u WHERE klauzuli obeležja i uslov za spajanje tabela. Pored uslova za spajanje u WHERE klauzuli mogu biti navedeni i uslovi selekcije n-torki. Uslov spajanja (odredjen operatorom izmedju unijiski kompatibilnih obeležja) može biti znak jednakosti, ali i bilo koji drugi smisleni odnos izmedju posmatranih obeležja (>, >=, <, <=, !=).

Ukoliko se izostavi uslov spajanja u WHERE klauzuli izvršava se Dekartov proizvod tabela iza FROM. Za dve tabele svaka n-torka prve spaja se sa svakom n-torkom druge.

Jedinstveni upit iz prethodnog primera glasi:

```
SELECT PREDAJE.S_PRED, S_NAS, PREZIME_IME
  FROM PREDAJE, NASTAVNIK
 WHERE NASTAVNIK.S_NAS = PREDAJE.S_NAS AND
       PREDAJE.S_PRED = 1;
```

S_PRED	S_NAS	PREZIME_IME
1	2	PETROVIC PETAR

Uslovi u WHERE klauzuli definišu uslove koji trebaju biti ispunjeni da bi se izvršilo spajanje n-torki tabela PREDAJE i NASTAVNIK.

Ukoliko se u prethodnom primeru u WHERE klauzuli izostavi uslov spajanja tabela kao rezultat bi dobili Dekartov proizvod (CARTESIAN JOIN) tabela PREDAJE i NASTAVNIK (ukupno 45 n-torki).

Nazivi tabela u WHERE klauzuli moraju se pisati samo ukoliko nazivi obeležja nisu jedinstveni u posmatranom skupu tabela. U prethodnom primeru u WHERE klauzuli umesto PREDAJE.S_PRED mogli smo pisati samo S_PRED jer je to jedinstveno obeležje u navedenom skupu tabela. Primećujemo i obavezu kvalifikacije obeležja iza SELECT.

Primer: Prikazati za svaku šifru nastavnika, odgovarajuće prezime i ime, šifru predmeta koji predaje i odgovarajući naziv predmeta.

Rešenje:

```
SELECT NASTAVNIK.S_NAS, PREZIME_IME, PREDMET.S_PRED, NAZIV
  FROM NASTAVNIK, PREDMET, PREDAJE
 WHERE NASTAVNIK.S_NAS = PREDAJE.S_NAS AND
       PREDAJE.S_PRED = PREDMET.S_PRED;
```

S_NAS	PREZIME_IME	S_PRED	NAZIV
2	PETROVIC PETAR	1	INFORMACIONI SISTEMI
5	ILIC JOVAN	2	STRUKTURE I BP
1	RADOVIC NIKOLA	3	OSNOVE RACUNARSTVA
1	RADOVIC NIKOLA	5	PROGRAMIRANJE RS
3	PEIC PETAR	5	PROGRAMIRANJE RS

Primer: Za svakog nastavnika prikazati prezime ime, platu kao i prezime ime i platu njegovog direktora. Redosled n-torki rezultata treba da bude po abecedi prezimena i imena nastavnika.

Rešenje:

```
SELECT A.PREZIME_IME, A.PLATA, B.PREZIME_IME
      "PREZIME_IME_DIR", B.PLATA PLATA_DIR
    FROM NASTAVNIK A, NASTAVNIK B
   WHERE A.S_DIR = B.S_NAS
 ORDER BY A.PREZIME_IME;
```

PREZIME_IME	PLATA	PREZIME_IME_DIR	PLATA_DIR
ILIC JOVAN	11500	SAVIC ILIJA	12500
PEIC PETAR	11500	SAVIC ILIJA	12500
PETROVIC PETAR	11500	SAVIC ILIJA	12500
SAVIC ILIJA	12500	RADOVIC NIKOILA	14500
SAVIC MILAN	13500	RADOVIC NIKOILA	14500
SIMIC SIMA	11500	SAVIC ILIJA	12500
TOTANA	12500	RADOVIC NIKOILA	14500

U ovom primeru rezultat je dobijen iz spoja tabele NASTAVNIK same sa sobom (SELF JOIN). Oznake A i B su upotrebljene kao dva privremena imena za tabelu NASTAVNIK. Spoj tabela A i B je izvršen za n-torce koje zadovoljavaju uslov jednakosti obeležja S_DIR iz prve sa S_NAS iz druge. Projekcija je izvršena po obeležjima navedenim iza SELECT. Klauzulom ORDER BY definisan je rastući redosled n-torki rezultata po obeležju PREZIM_IME iz tabele A.

Uočavamo da se u rezultatu ne dobijaju podaci o nastavnicima koji za obeležje S_DIR imaju NULL vrednost. Ako se žele prikazati n-torce koji za obeležje koje učestvuje u WHERE uslovu spajanja imaju NULL vrednost potrebno je izvršiti **spoljno spajanje** (OUTER JOIN). Spoljno spajanje dve tabele realizuje se tako da se označenoj tabeli (tabela koja ne sadrži NULL vrednosti) za potrebe spajanja doda još jedna n-torka koja sadrži null-vrednosti svih obeležja. Takva dodata n-torka spaja se sa onim n-torcama za koje se ne pronadje odgovarajuća stvarna n-torka. Spoljno spajanje se označava navodnjem u WHERE klauzuli oznake "(+)" iza *tabela.obeležje* kojoj treba dodati n-torku.

SQL izraz za prethodni primer glasi:

```
SELECT A.PREZIME_IME, A.PLATA, B.PREZIME_IME
      "PREZIME_IME_DIR", B.PLATA PLATA_DIR
   FROM NASTAVNIK A, NASTAVNIK B
  WHERE A.S_DIR = B.S_NAS (+)
 ORDER BY A.PREZIME_IME;
```

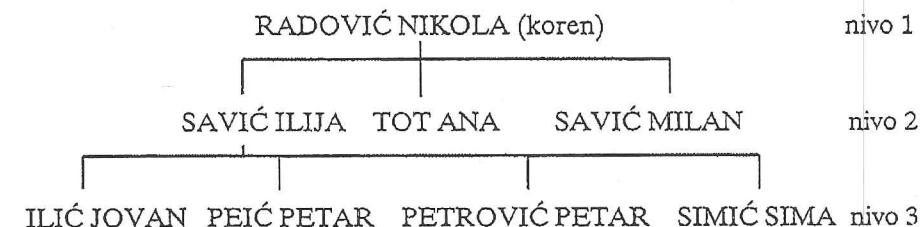
PREZIME_IME	PLATA	PREZIME_IME_DIR	PLATA_DIR
ILIĆ JOVAN	11500	SAVIĆ ILIJA	12500
PEIĆ PETAR	11500	SAVIĆ ILIJA	12500
PETRIĆ JANKO	13500		
PETROVIĆ PETAR	11500	SAVIĆ ILIJA	14500
RADOVIĆ NIKOLA	14500		
SAVIĆ ILIJA	12500	RADOVIĆ NIKOLA	14500
SAVIĆ MILAN	13500	RADOVIĆ NIKOLA	14500
SIMIĆ SIMA	11500	SAVIĆ ILIJA	12500
TOT ANA	12500	RADOVIĆ NIKOLA	14500

Treba uočiti da je u WHERE, uslov A.S_DIR = B.S_NAS identičan sa B.S_NAS = A.S_DIR, dok uslov A.S_DIR = B.S_NAS (+) nije identičan sa B.S_NAS = A.S_DIR (+).

Možemo se ovde još jednom podsetiti na neproceduralnost SQL tj. izražavanja ŠTA želimo dobiti, a ne i KAKO. Zamislimo šta bi sve morali da uključimo u programski kod ukoliko bi prethodni zahtev trebalo realizovati nekim od viših programskih jezika (npr. COBOL, PASCAL, FORTRAN).

9.2.2.2.5. Pretraživanje tabela koje u sebi sadrže strukturu tipa stabla

Kažemo da tabela sadrži strukturu tipa stabla kada sadrži n-torku koja referencira na jednu ili više drugih n-torki iste tabele. Primer takve tabele je tabela NASTAVNIK u kojoj se nalaze i obeležja S_NAS i S_DIR. Vrednost obeležje S_DIR u jednoj n-torci referencira na nadredjenog nastavnika za kojeg takodje postoji odgovarajuća n-torka. Koristeći obeležja S_NAS i S_DIR odnosi u tabeli NASTAVNIK mogu se prikazati na sledeći način:



Za prikaz informacija strukture tipa stabla koriste se klauzule:

CONNECT BY - definije strukturu stabla,
START WITH - definije koren stabla.

Primer: Prikazati šifru, prezime ime, zvanje i šifru prepostavljenog nastavnika u obliku stabla.

Rešenje:

```
SELECT S_NAS, PREZIME_IME, ZVANJE, S_DIR
  FROM NASTAVNIK
CONNECT BY PRIOR S_NAS = S_DIR
START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR
1	RADOVIĆ NIKOLA	R PROF	
6	SAVIC ILIJA	V PROF	1
2	PETROVIĆ PETAR	DOCENT	6
3	PEIĆ PETAR	DOCENT	6
4	SIMIĆ SIMA	DOCENT	6
5	ILIĆ JOVAN	DOCENT	6
7	TOT ANA	V PROF	1
8	SAVIC MILAN	R PROF	1

Prethodni rezultat se formira na sledeći način: Prvo se selektira n-torka iz tabele NASTAVNIK po uslovu u START WITH klauzuli. To je n-torka o nastavniku sa prezimenom imenom RADOVIĆ NIKOLA i šifrom 1. Zatim se izvršava uslov u CONNECT BY klauzuli koji kaže da prethodna šifra nastavnika (1) postaje šifra prepostavljenog nastavnika za selekciju sledeće n-torce. Pretraživanjem tabele NASTAVNIK za uslov S_DIR = 1 dobija se n-torka o nastavniku SAVIĆ ILIJI sa šifrom 6. Zatim ta šifra postaje kriterij selekcije po koloni S_DIR.

Čvorovi u stablu mogu biti označeni brojevima nivoa u zavisnosti od udaljenosti od korena, korišćenjem pseudokolone LEVEL.

```
SELECT LEVEL, S_NAS, PREZIME_IME, ZVANJE, S_DIR
  FROM NASTAVNIK
 CONNECT BY PRIOR S_NAS = S_DIR
 START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

LEVEL	S_NAS	PREZIME_IME	ZVANJE	S_DIR
1	1	RADOVIĆ NIKOLA	R PROF	
2	6	SAVIC ILIJA	V PROF	1
3	2	PETROVIĆ PETAR	DOCENT	6
3	3	PEIĆ PETAR	DOCENT	6
3	4	SIMIĆ SIMA	DOCENT	6
3	5	ILIĆ JOVAN	DOCENT	6
2	7	TOT ANA	V PROF	1
2	8	SAVIC MILAN	R PROF	1

Da bi se dobila organizaciona šema kao uredjena lista, potrebno je upotrebiti funkciju LPAD.

```
SELECT LPAD(' ',2 * LEVEL) || PREZIME_IME "PREZIME_IME",
        LEVEL, S_NAS, ZVANJE, S_DIR
  FROM NASTAVNIK
 CONNECT BY PRIOR S_NAS = S_DIR
 START WITH PREZIME_IME = 'RADOVIĆ NIKOLA';
```

PREZIME_IME	LEVEL	S_NAS	ZVANJE	S_DIR
RADOVIĆ NIKOLA	1	1	R PROF	
SAVIC ILIJA	2	6	V PROF	1
PETROVIĆ PETAR	3	2	DOCENT	6
PEIĆ PETAR	3	3	DOCENT	6
SIMIĆ SIMA	3	4	DOCENT	6
ILIĆ JOVAN	3	5	DOCENT	6
TOT ANA	2	7	V PROF	1
SAVIC MILAN	2	8	R PROF	1

9.2.2.3. Izmena sadržaja tabele - UPDATE

Opšti oblik naredbe je:

a) UPDATE naziv_tabele [alias]
 SET naziv_obi₁ = izraz₁, [naziv_obi₂ = izraz₂]...
 [WHERE kriterijum za izmenu n-torki];

b) UPDATE naziv_tabele [alias]
 SET (naziv_obi₁ [, naziv_obi₂]...) = (podupit)
 [WHERE kriterijum za izmenu n-torki];

gde je: izraz_i - izraz koji određuje vrednost koja se pridružuje obi_i.
 podupit - selekcija vrednosti (SELECT naredba)

Ovom naredbom menja se sadržaj navedenih obeležja onih n-torki koje zadovoljavaju kriterijum za izmenu n-torki u WHERE klauzuli. Ako nije navedena klauzula WHERE izmena sadržaja vrši se u svim n-torkama tabele.

Klauzula SET određuje obeležja koja podležu promeni i vrednosti koje će im biti dodeljene.

U svakoj n-torki koja zadovoljava WHERE klauzulu, obeležja levo od znaka jednakosti dobijaju vrednosti odredjene izrazom sa desne strane znaka jednakosti.

Ako je u SET klauzuli naveden podupit on mora vratiti tačno jednu n-torku tako da se vrednosti odredjene u SELECT klauzuli podupita pridružuju obeležjima navedenim u listi koja je u zagradama. Uočavamo da broj vrednosti podupita mora odgovarati broju obeležja te da je pridruživanje vrednosti po redosledu navodjenja.

Primer: Za nastavnike sa zvanjem VANREDNI PROFESOR upisati dodatak od 1200 dinara:

Rešenje:

```
UPDATE NASTAVNIK
SET DODATAK = 1200
WHERE ZVANJE = 'V PROF';
```

Primer: Za nastavnike sa zvanjem DOCENT povećati platu za 5 % i ukinuti dodatak:

Rešenje:

```
UPDATE NASTAVNIK
SET PLATA = 1.05 * PLATA, DODATAK = 0
WHERE ZVANJE = 'DOCENT';
```

Provera rezultata prethodne naredbe može se jednostavno izvršiti na sledeći način:

```
SELECT *
FROM NASTAVNIK
WHERE ZVANJE = 'DOCENT';
```

S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
2	PETROVIĆ PETAR	DOCENT	6	01-FEB-82	12075	0
3	PEIĆ PETAR	DOCENT	6	01-MAR-83	12075	0
4	SIMIĆ SIMA	DOCENT	6	01-APR-84	12075	0
5	LJUČ JOVAN	DOCENT	6	01-MAY-85	12075	0

Primer: Svim nastavnicima sa zvanjem DOCENTA upisati dodatak u iznosu od 10 % dodatka nastavnika RADOVIĆ NIKOLE:

Rešenje:

```
UPDATE NASTAVNIK
SET DODATAK = (SELECT 0.1 * DODATAK
                 FROM NASTAVNIK
                WHERE PREZIME_IME = 'RADOVIĆ NIKOLA')
WHERE ZVANJE = 'DOCENT';
```

Primer: Promeniti šifru predmeta "informacioni sistemi" na 6:

Rešenje:

```
UPDATE PREDMET
SET S_PRED = 006
WHERE S_PRED = 001;
```

Moramo izvršiti i odgovarajuću promenu u tabeli PREDAJE:

```
UPDATE PREDAJE
SET S_PRED = 006
WHERE S_PRED = 001;
```

Jednom naredbom UPDATE nije moguće izvršiti promenu u više od jednoj tabeli. Zbog toga se u prethodnom primeru susrećemo sa problemom integriteta podataka. Posle izvršavanja prve naredbe UPDATE iz prethodnog primera baza podataka u tabeli PREDAJE sadrži n-torce u kojima se navodi nepostojeća šifra predmeta. Promena redosleda UPDATE naredbi ne rešava problem. Zbog toga je važno obezbediti izvršavanje obe naredbe, a ne samo jedne. Problemom očuvanja integriteta baze podataka kada se zahteva više promena nećemo se ovde baviti.

9.2.2.4. Brisanje n-torki table - DELETE

Opšti oblik naredbe je:

```
DELETE FROM naziv_tabele
[WHERE kriterijum brisanja n-torki];
```

Naredbom DELETE brišu se sve n-torce tabele koje ispunjavaju kriterijum za brisanje n-torki. Ako nije zadata klauzula WHERE brišu se sve n-torce tabele.

Primer: Brisati podatke o nastavniku sa šifrom 008 (brisanje jedne n-torce):

Rešenje:

```
DELETE FROM NASTAVNIK
WHERE S_NAS = 008;
```

Primer: Brisati podatke o angažovanju svih nastavnika na predmetu sa šifrom 005 (brisanje više n-torki):

Rešenje:

```
DELETE FROM PREDAJE
WHERE S_PRED = 005;
```

Primer: Brisati sve n-torce tabele PREDMET:

Rešenje:

```
DELETE FROM PREDMET;
```

Ova operacija ne ukida definiciju tabele već samo n-torce pa po svom dejstvu nije ekvivalentna naredbi DROP.

Primer: Brisati sve n-torce u tabeli PREDAJE za koje je semestar u kojem se predmet predaje osmi:

Rešenje:

```
DELETE FROM PREDAJE
WHERE S_PRED IN (SELECT DISTINCT PREDMET.S_PRED
                  FROM PREDMET, PREDAJE
                 WHERE PREDMET.S_PRED = PREDAJE.S_PRED
                   AND PREDMET.SEMESTAR = 8);
```

9.2.3. Rečnik podataka

Rečnik podataka sistema za upravljanje relacionom bazom podataka je skup tabela i pogleda koji sadrže informacije o bazi podataka. Na primer u DB2 rečnik podataka ima 20 - 25 tabela.

Rečnik podataka opisuje tabele, poglede, indekse, grupe, korisnike, privilegije i druge informacije o trenutnom stanju baze podataka. SUBP automatski kad god se neki objekat kreira ili menja, ažurira rečnik podataka.

Tabela DTAB opisuje tabele koje čine rečnik podataka sa sledeća dva obeljžja: TNAME - naziv_tabele, REMARKS - opis tabele. Korišćenjem SQL naredbe SELECT može se prikazati sadržaj tabele DTAB.

```
SELECT *
  FROM DTAB;
```

Deo rezultata dobijen prethodnim upitom glasi:

IME TABELE	OPIS TABELE
CATALOG	Tabele i pogledi kojima korisnik ima pravo pristupa
COLUMNS	Obeležja tabele sa pravom dostupa
DTAB	Opis tabele i pogleda u rečniku podataka SUBP
INDEXES	Indeksi koje je kreirao korisnik
SPACES	Definicija prostora za kreiranje tabela i pogleda
SYSCOLUMNS	Opis obeležja u tabelama
SYSUSERLIST	Lista korisnika SUBP
SYSVIEWS	Lista pogleda kojima korisnik ima pravo pristupa

Primer: Prikazati opis korisničke tabele NASTAVNIK.

Opis obeležja u tabelama nalazi se u tabeli SYSCOLUMNS koja ima više obeležja i koje treba poznavati da bi realizovali prethodni zahtev.

```
SELECT CNAME "OBL", COLTYPE TIP, WIDTH "DUŽINA",NULLS
      FROM SYSCOLUMNS
     WHERE TNAME = 'NASTAVNIK';
```

OBL	TIP	DUŽINA	NULLS
S_MAS	NUMBER	3	NOT NULL
PREZIME_IME	CHAR	15	NOT NULL
ZVANJE	CHAR	6	NULL
S_DIR	NUMBER	3	NULL
DATZAP	DATE	7	NULL
PLATA	NUMBER	8	NOT NULL
DODATAK	NUMBER	8	NULL

Korišćenjem naredbi SQL za pretraživanje rečnika podataka, korisnik može doći do informacija o bazi podataka, a zatim preći na njeno pretraživanje. U tradicionalnim sistemima rečnik podataka i baza podataka su bili različito organizovani te je i pristup bio preko različitih interfejsa.

I dok se operacija SELECT iz SQL može obavljati nad tabelama i pogledima rečnika podataka, operacije UPDATE, DELETE i INSERT nisu dozvoljene. Ovakvo ograničenje je sasvim razumljivo s obzirom da SUBP koristi rečnik podataka za održavanje baze podataka. Promene samo u rečniku podataka mogле bi da izazovu nemogućnost korišćenja baze podataka. Naredbe SQL automatski vrše održavanje rečnika podataka. Na primer, posle naredba CREATE TABLE ... u:

- a) SYSTABLES biće upisan nova n-torka,
- b) SYSCOLUMNS za svako obeležje tabele biće upisana jedna n-torka,
- c) Neka druga dejstva koja nisu predmet razmatranja.

Rečnik podataka sadrži informacije i o samim tabelama rečnika podataka. Takve podatke generiše SUBP prilikom konfigurisanja sistema.

9.3. Jezik upita na osnovu primera - QBE

Jezik postavljanja upita na osnovu primera (Query-By-Example-QBE) je jezik u kome korisnik postavlja upit na relationalnu bazu podataka odgovarajućim popunjavanjem definisane tabele. SQL je svakako korisnicima prihvatljiviji od SQL i to pre svega korisnicima koji nemaju iskustva u programiranju. U SQL se prepostavlja odredjeno iskustvo u programiranju i on predstavlja kako jezik u klasičnom smislu tako i jezik vrlo visokog nivoa. Nasuprot SQL-u QBE se upiti definišu jednostavnom popunom tabele. Takav pristup je veoma lak za učenje i shvatanje i to pre svega za manje iskusne korisnike. Na primer, u sistemu za upravljanje relacionom bazom podataka DB2 preko dela za upravljanje upitima korisnik može izabratи da li će upit definisati u SQL ili QBE.

U QBE postoje sledeće operacije:

QBE	SQL	objašnjenje
P.	SELECT	štampa
U.	UPDATE	izmena
D.	DELETE	brisanje
I.	INSERT	dodavanje

Druge operacije nisu moguće u QBE i moraju se izvršavati korišćenjem SQL.

Neke osnovne načine korišćenja QBE ilustrovati ćemo na nekoliko primera. Pretpostavljamo da su tabele definisane prema istoj relacionoj šemi koju smo koristili za ilustraciju SQL naredbi.

Primer: Prikazati šifru, prezime ime, i platu svih nastavnika sa zvanjem docenta.

NASTAVNIK	S_NAS	PREZIME_IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
	P.	P.	DOCENT			P.	

Korisniku se nudi prazna tabela koju on popunjava saglasno potrebama upita. Korisnik može izvršiti redakciju tabele za postavljanje upita saglasno potrebama. U prethodnom primeru se redakcijom mogu isključiti kolone S_DIR, DATZAP i DODATAK.

Ako želimo prikaz svih kolona iz tabele NASTAVNIK specifikacija P. može se zadati i na sledeći način:

NASTAVNIK	S_NAS	PREZIME	IME	ZVANJE	S_DIR	DATZAP	PLATA	DODATAK
P.								

Podsećanja radi prethodni upit ekvivalentan je u SQL sledećem upitu:

```
SELECT *
  FROM NASTAVNIK;
```

Uslovi koji su povezani operatorom AND zadaju se u jednom redu tabele dok se uslovi povezani operatorom OR zadaju u više redova ili primenom *lemenata veze i bloka uslova*.

Primer: Prikazati prezime ime, zvanje i platu nastavnika koji imaju zvanje vanrednog ili redovnog profesora i platu veću od 12500. Rezultujuće n-torce prikazati po abecedi zvanja i prezimena imena nastavnika.

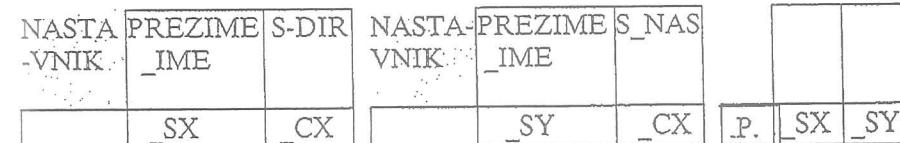
Rešenje:

NASTAVNIK	PREZIME	IME	ZVANJE	PLATA
	P.AO		P. "V PROF"	P. > 12500
	P.AO		P. "R PROF"	P. > 12500

U prethodnom primeru AO označava rastući redosled (DO - označava opadajući redosled). Uslovi navedeni u istom redu povezani su AND operatorom, dok uslovi navedeni u različitim redovima povezani su OR operatorom.

Primer: Prikazati prezime ime nastavnika i prezime ime njemu prepostavljenog nastavnika.

Rešenje:



Oznake _SX, _CX i _SY predstavljaju elemente veze. Treća tabela može biti bez naziva i bez naziva kolona, a definije oblik rezultata. To su parovi prezime ime nastavnika (SX)/prezime ime njemu prepostavljenog nastavnika (SY) nastali putem elementa veze _CX n-torki tabele NASTAVNIK.

Primer: Nastavniku sa šifrom 2 upisati zvanje vanrednog profesora i platu povećati 10%.

Rešenje:

NASTAVNIK	S_NAS	ZVANJE	PLATA	PLATA
P.	002	U. "V PROF"	WT	U. WT + 0,1 * WT

Može se uočiti da se oznaka U. (izmena sadržaja) odnosi na zvanje i platu.

Primer: U tabeli nastavnik brisati podatke o PETROVIĆ PETRU.

Rešenje:

NASTAVNIK	PREZIME	IME
D.	"PETROVIĆ PETAR"	

Oznaka D. odnosi se na kompletну n-torku i zbog toga je upisana u koloni imena tabele.

Na kraju napomenimo da postoje upiti koji se mogu realizovati u SQL ali ne mogu u QBE. Na primer QBE u DB2 nema odgovarajućih naredbi za sledeće SQL mogućnosti:

- NOT EXISTS,
- standardne funkcije (SUM, AVG itd.),
- GROUP BY i HAVING.

Korišćenjem nekih drugih rutina DB2 može ipak realizovati neke od prethodnih mogućnosti SQL, mada je i SQL deo DB2.

9.4. Transformacija upita

Transformacija upita iz jednog oblika u drugi posebno je potrebna iz razloga optimizacije upita radi efikasnog rada i korišćenja računarskog sistema. Optimizacija zahteva za obradom podataka susreće se kod *prevodioca sa optimizacijom* koji menjaju redosled operacija i otklanjaju suvišna izračunavanja. I dok se pri takvoj optimizaciji dobitak iskazuje u mikrosekundama, u slučaju složenih informacionih zahteva na bazi podataka optimizacija može smanjiti broj pristupa spoljnoj memoriji za red veličine, što predstavlja značajnu razliku. Na primer, sasvim je moguće slučaj da je za izvršavanje jednog upita na jedan način potrebno nekoliko časova, a na drugi način nekoliko minuta. Obično prethodna situacija nastaje u slučaju kada se po prvom načunu čitaju skoro svi zapisi baze podataka i za svaki od njih pokušavaju pronaći drugi odgovarajući zapisi da bi se na kraju izabralo samo nekoliko zapisa. Sasvim sigurno brži način sastoji se u pokušaju da se na samom početku izabere manji skup relevantnih zapisa koji se dalje obraduju. Prvi od načina zasniva se na izvršavanju upita na način kako je zapisan, pri čemu po pravilu korisnik koji takav zahtev definiše ne zna da je takav zahtev neefikasan. Dobar sistem treba dozvoliti korisniku definisanje upita kako je to njemu pogodno bez obaveza razmišljanja o efikasnosti pri čemu će sam sistem preuzeti ulogu optimizacije.

Drugi važan razlog transformacije upita su sve više i sve češće korišćene distribuirane baze podataka. Korisnik koji poznaje jedan upitni jezik želi pristupiti udaljenoj bazi podataka u kojoj se koristi drugi upitni jezik. Umesto da takvog korisnika upućujemo na izučavanje tog drugog upitnog jezika, može se realizovati automatsko prevodenje korisnikovog informacionog zahteva na njemu poznatom upitnom jeziku na informacioni zahtev definisan prema upitnom jeziku koji dozvoljava data baza podataka.

Postojanje nekoliko upitnih jezika predstavlja prednost i u slučaju kada korisnik radi sa jednom bazom podataka. U tom slučaju korisnik može odabrati upitni jezik koji njemu odgovara bez obaveze izučavanja novog upitnog jezika.

Najbolje rešenje je realizacija jednog upitnog jezika na najefikasniji način pri čemu bi se informacioni zahtevi na drugim upitnim jezicima prevodili na taj jezik, umesto nezavisne realizacije svakog posebnog upitnog jezika. U većini

danasa korišćenih sistema za upravljanje relacionim bazama podataka taj opšti upitni jezik zasniva se na relationalnoj algebri.

Načini formulisanja upita na relationalnoj algebri i relacionom računu su sasvim različiti. U praksi se potreba prevodenja iz relacione algebre na relacioni račun javlja vrlo retko. Većina relationalnih SUBP ima mogućnost izvršavanja naredbi kao što je spajanje pa je problem pre u prevodenju izraza na relacionom računu u izraze relacione algebre koji su pogodni za realizaciju. Detaljna razmatranja prevodenja izraza relacione algebre u izraze relationalnog računa i obrnuto nećemo razmatrati.

Značajni efekti u optimizaciji upita na relationalnoj algebri često se mogu postići jednostavnom променом načina sračunavanja izraza (promena redosleda izvodjenja operacija). Na primer, izvodjenjem operacije selekcije pre operacije spajanja često možemo doći brže do rezultata. To proizilazi iz toga što operacije tipa spajanja i Dekartovog proizvoda generišu veliki broj torki. Ako pre izvodjenja takvih operacija možemo izvršiti selekciju tada možemo značajno smanjiti broj generisanih torki. Ovo je jedan od primera primene opštег principa: uvek treba koristiti postojeće informacije datog zadatka da bi se smanjila količina informacija na izlazu generatora, umesto da se prvo formira izlaz svih informacija iz kojeg će zatim biti odstranjene nepotrebne informacije.

Prethodni princip zasniva se na sledećem pravilu:

Neka su $r(R)$ i $p(P)$ dve tabele i neka uslov selekcije F ne sadrži obeležja tabele p . Tada vredi pravilo o ekvivalentnosti izraza:

$$\sigma_F(r \Delta p) = \sigma_F(r) \Delta p.$$

Prethodni izraz možemo iskazati rečima na sledeći način: selekcija prirodnog spoja tabela r i p prema uslovu F jednaka je prirodnom spoju selekcije nad tabelom r prema uslovu F i tabele p .

Dokaz prethodnog pravila zasniva se na komutativnosti logičke i operacije.

Razmotrimo sada jedan upit koji treba izraziti sa operatorima relacione algebre nad ranije definisanim relacionim šemama NASTAVNIK, PREDMET i PREDAJE.

Primer: Korišćenjem operatora relacione algebre prikazati šifru predmeta, šifru nastavnika i prezime ime nastavnika koji predaje predmet sa šifrom 1.

Rešenje:

a) Upit izražen operacijama u relacionoj algebri glasi:

$$\pi_{S_PRED \ S_NAS \ PREZIME_IME} (\sigma_{S_PRED=1} (NASTAVNIK \ \nabla \ PREDAJE)).$$

Rezultat:

(S_PRED S_NAS PREZIME_IME)		
1	2	PETROVIC PETAR

Do željenog rezultata dolazimo izvodjenjem sledećih operacija:

(1) Prirodni spoj tabela NASTAVNIK i PREDAJE.

(1.1) Dekartov proizvod tabela NASTAVNIK i PREDAJE
(rezultat 45 torki)

(1.2) Selekcija nad tabelom dobijenom u (1.1) prema uslovu
NASTAVNIK.S_NAS = REDAJE.S_NAS (rezultat 5 torki)

(1.3) Projekcija po obeležjima koja se nalaze u tabeli NASTAVNIK i
obeležjima koja se nalaze u tabeli PREDAJE a ne nalaze se u
tabeli NASTAVNIK (eliminisanje istih kolona) (rezultat 5 torki).

(2) Selekcija torki iz rezultata dobijenog u (1) prema uslovu S_PRED = 1
(rezultat 1 torka).

(3) Projekcija po obeležjima S_PRED, S_NAS i PREZIME_IME nad
tabelom dobijenom u (2) (rezultat 1 torka).

b) Korišćenjem prethodno razmatranog pravila isti rezultat možemo dobiti
ako zahtev iskažemo sledećim izrazom u relacionoj algebri:

$$\pi_{S_PRED \ S_NAS \ PREZIME_IME} ((\sigma_{S_PRED=1} (PREDAJE)) \ \nabla \ NASTAVNIK).$$

Rezultat:

$$(S_PRED \ S_NAS \ PREZIME_IME)$$

1	2	PETROVIC PETAR
---	---	----------------

Do prethodnog rezultata možemo doći izvodjenjem sledećih operacija:

(1) Selekcija nad tabelom PREDAJE prema uslovu S_PRED = 1
(rezultat 1 torka odnosno tabela sa oznakom TEMP).

(2) Prirodni spoj tabele NASTAVNIK i tabele dobijene kao rezultat u
koraku (1).

(2.1) Dekartov proizvod tabele NASTAVNIK i rezultata iz koraka (1)
(rezultat 9 torki)

(2.2) Selekcija nad tabelom dobijenom u (2.1) prema uslovu
NASTAVNIK.S_NAS = TEMP.S_NAS (rezultat 1 torka)

(2.3) Projekcija po obeležjima koja se nalaze u tabeli NASTAVNIK
i obeležjima koja se nalaze u tabeli TEMP a ne nalaze se u
tabeli NASTAVNIK (eliminisanje istih kolona) (rezultat 1 torka).

(3) Projekcija po obeležjima S_PRED, S_NAS i PREZIME_IME nad
tabelom dobijenom u (2) (rezultat 1 torka).

Iz prethodnog primera možemo uočiti da se prema rešenju a) (prvo
izvršavanje spoja) kao prvi medjurezultat dobija tabela od 45 torki, dok se
prema rešenju b) (prvo izvršavanje selekcije) kao medjurezultat dobija tabela
sa 9 torki. Sasvim je očigledno da se do rezultata brže dolazi sračunavanjem
izraza pod b).

Prethodno pravilo na kojem se zasniva transformacija jednog izraza relacione
algebre u drugi možemo uopštiti ako uslov selekcije F predstavimo kao
konjukciju sledeća četiri uslova:

- F_r koji se odnosi samo na tabelu r ,
- F_p koji se odnosi samo na tabelu p ,
- F_q koji se odnosi na obeležja (kolone) zajednička za tabele r i p ,
- F_s koji sadrži sve što nije moguće uključiti u F_r , F_p ili F_q .

Svaki od uslova F_r , F_p , F_q ili F_s može biti izostavljen što znači da se selekcija ne
izvršava.

Sada imamo:

$$\sigma_{F_r \wedge F_p \wedge F_q \wedge F_s} (r \nabla p) = \sigma_{F_s} (\sigma_{F_r \wedge F_q} (r) \nabla \sigma_{F_p \wedge F_q} (p)).$$

Prethodna razmatranja optimizacije izraza relacione algebre imala su za cilj da ilustruju moguće posledice različito iskazanog istog upita. Kao što je bilo napomenuto od korisnika se ne može očekivati potpuno poznavanje svih mogućih principa optimizacije već se isti mogu ugraditi u SUBP i automatski izvoditi.

10. PREVODJENJE MODELA OBJEKTI-VEZE OBELEŽJA U RELACIONI MODEL

Pri prevodenju jednog modela podataka u drugi ne znači da svakom konceptu strukture jednog modela mora odgovarati koncept strukture drugog modela, svakoj operaciji jednog modela da mora odgovarati operacija drugog modela i svakom eksplicitnom ograničenju jednog modela da mora odgovarati eksplicitno ograničenje drugog modela.

Kod prevodenja jednog semantički bogatog modela (modela III generacije) u semantički manje bogat model (model II generacije) dolaze do izražaja prethodno navedena neslaganja. Po pravilu se deo semantike iskazan strukturon semantički bogatog modela mora predstaviti eksplicitnim ograničenjima semantički manje bogatog modela. Iz toga neposredno sledi da apstraktnoj operaciji semantički bogatog modela odgovara procedura semantički manje bogatog modela.

S obzirom da za sada ne postoji komercijalno raspoloživ SUBP zasnovan na MOV to je u praktičnoj realizaciji potrebno model podataka izradjen prema MOV prevesti u neki od modela za koji postoji SUBP.

Kako za relacioni model podataka postoji više komercijalnih softvera prevodenje iz MOV najčešće se vrši u relacioni model podataka.

Model objekti - veze - obeležja je semantički bogat model (model III generacije), dok je relacioni model semantički manje bogat (model II generacije), pa sve do sada rečeno za prevodenje semantički bogatog u semantički manje bogat model u potpunosti važi i za njih.

Prevodenje MOV u relacioni model izvodi se na sledeći način:

1. Deo strukture MOV, odnosno DOV se predstavlja relacionom šemom.
2. Ograničenja, operacije i deo strukture MOV se predstavljaju operacijama za očuvanje integriteta definisanim nad relacionim modelom, a implementiraju korišćenjem sredstava odabranog relacionog SUBP.

Ovde se razmatraju odnose samo na prevodjenje DOV u relacijski model. Kao rezultat prevodjenja DOV dobija se relaciona šema baze podataka sa odgovarajućim pravilima integriteta. Na osnovu skupa pravila koja se u prevodjenju primenjuju, postupak prevodjenja DOV u relacijski model može biti formalizovan i automatizovan. Postoje više postupaka za prevodjenje DOV u relacijski model podataka, a ovde se izlaže jedan praktičan postupak koji se može i automatizovati.

10.1 Postupak prevodjenja

1. Pravila za objekte:

1.1. Svaki objekat iz DOV postaje šema relacije. Ime tipa objekta postaje ime šeme relacije. Obeležja objekta su obeležja šeme relacije. Za osnovne objekte identifikator objekta postaje primarni ključ šeme relacije.

1.2. Svaki slab objekat takođe postaje šema relacije. Ime slabog objekta postaje ime šeme relacije. Obeležja objekta su obeležja šeme relacije. Identifikator nadredjenog objekta postaje obeležje šeme relacije koja odgovara slabom objektu. Identifikator slabog objekta čini identifikator nadredjenog objekta i obeležja slabog objekta koja jedinstveno identificuju pojavljivanje slabog objekta u okviru pojavljivanja njemu nadredjenog objekta.

Na primer, za DOV dat na slici 7.7. prevodjenjem u relacioni model primenom do sad definisanih pravila dobijamo sledeće šeme relacija:

DRŽAVA(S_DRŽ, NAZIV_DRŽ) (prema pravilu 1.1)
PROIZVODJAČ(S_DRŽ, S_PRO, NAZIV_PRO) (prema pravilu 1.2)

1.3. Objekat nadtip (generalizovani tip objekta) postaje šema relacije. Ime nadtipa postaje ime šeme relacije. Obeležja nadtipa su obeležja šeme relacije. Identifikator nadtipa postaje ključ šeme relacije.

1.4. Objekat podtip takođe postaje šema relacije. Ime podtipa postaje ime šeme relacije. Obeležja podtipa su obeležja šeme relacije. Identifikator nadtipa postaje obeležje šeme relacije podtipa i predstavlja ključ šeme relacije.

DOV dat na slici 7.9. prevodi se u relacioni model sa sledećim šemama relacija:

RADNIK(JMBG, IME,DATUM_RODJ,
VRSTA_POSLA), (prema pravilu 1.3)
NASTAVNIK(JMBG, ZVANJE, DATUM_IZB), (prema pravilu 1.4)
SLUŽBENIK(JMBG, ST_SPREMA), (prema pravilu 1.4)
ODRŽAVANJE(JMBG, ZANIMANJE), (prema pravilu 1.4)

DOV dat na slici 7.11. prevodimo u relacioni model sa sledećim šemama relacija:

2. Pravila za binarne veze.

2.1. Veza sa kardinalnošću 1:1

Veze tipa 1:1 po pravilu nemaju obeležja. Sva obeležja koja bi eventualno mogla biti pripisana samoj vezi, zapravo su obeležja jednog od objekata koji čine tu vezu. Dakle mogu biti pripisana tom objektu, a time postati obeležja šeme relacije kojom se taj tip objekta predstavlja.

2.1.1. Vezu sa kardinalnošću (1,1) : (1,1), i oba objekta koji u njoj učestvuju prevodimo u jednu šemu relacije čija su obeležja sva obeležja jednog i drugog objekta. Kandidati za ključ u ovoj šemi relacije su identifikatori jednog i drugog objekta koji su u vezi.

2.1.2. Vezu sa kardinalnošću $(0,1)$: $(1,1)$, i objekte u vezi prevodimo u dve šeme relacije. Za svaki objekat u vezi po jedna šema relacije (prema već definisanom pravilu 1.1), s tim što se identifikator jednog od objekta koji su u vezi ubaci za obeležje druge šeme relacije. Dakle veza se predstavlja *spojnim ključem*.

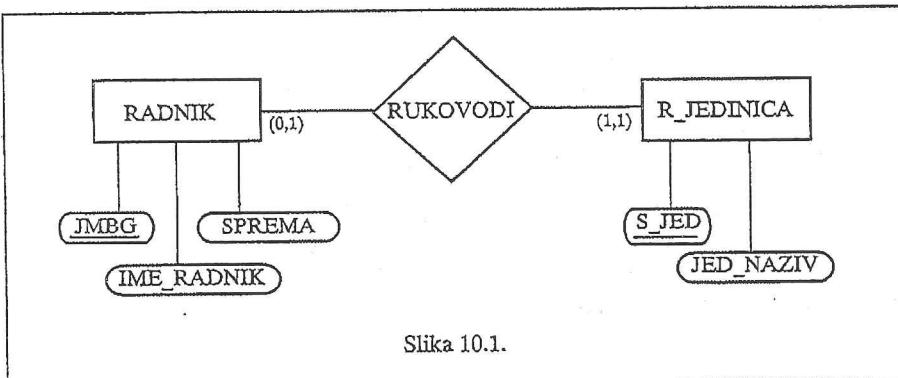
Šemu relacije u koju ćemo uvrstiti spoljni ključ biramo tako da relacija sadrži što manje nul - vrednosti i da njeno korišćenje bude što efikasnije.

DOV dat na slici 10.1. prevodimo u relacioni model sa sledećim šemama relacija:

R_JEDINICA(S_IED, JED_NAZIV) (prema pravilu 1.1)
RADNIK(JMBG,IME_RADNIK,SPREMA,S_JED) (prema pravilu 1.1 i 2.1.2)

ili

R_JEDINICA(S_JED, JED_NAZIV, JMBG) (prema pravilu 1.1 i 2.1.2)
RADNIK(JMBG, IME_RADNIK, SPREMA) (prema pravilu 1.1)



Slika 10.1.

Šeme relacija pod a) uzrokovale bi da svaka n - torka za svakog radnika koji nije rukovodilac ima nul - vrednost obeležja S_JED (tj. spoljni ključ).

Prema šemama relacija pod b), s obzirom da radna jedinica obavezno ima rukovodioca, predstavljanje veze RUKOVODI spoljnjim ključem u šemi relacije R_JEDINICA ne dovodi do pojave nul - vrednosti.

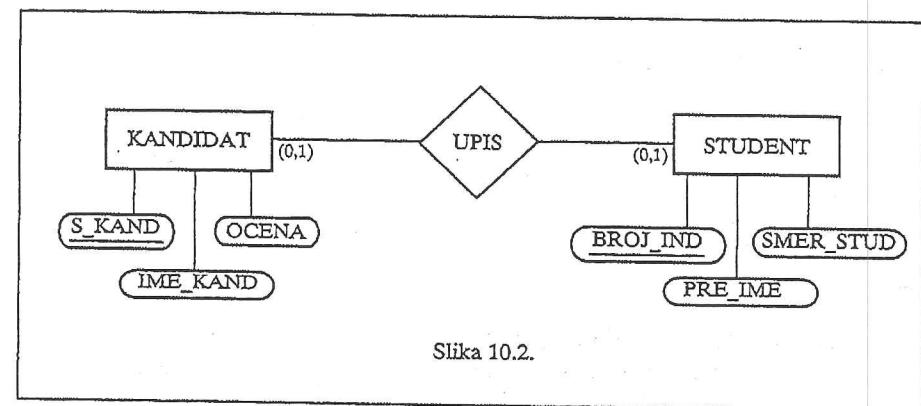
Prema tome, pravilo za prevodjenje veze sa kardinalnošću $(0,1) : (1,1)$ je njeno predstavljanje spoljnjim ključem u šemi relacije objekta sa strane $(1,1)$.

2.1.3. Za vezu sa kardinalnošću $(0,1) : (0,1)$, kreiraju se tri šeme relacija. Po jedna za svaki objekat (prema već definisanom pravilu 1.1) i jedna za vezu. Obeležja u šemi relacije koja odgovaraju vezi su i identifikatori objekata koji su u vezi i oba su kandidati za ključ.

DOV dat na slici 10.2. prevodimo u relacioni model sa sledećim šemama relacija:

KANDIDAT(S_KAND, IME_KAND, OCENA), (prema pravilu 1.1)
STUDENT(BROJ_IND, PRE_IME, SMER_STUD) (prema pravilu 1.1)
UPIS(BBROJ_IND, S_KAND) (prema pravilu 2.1.3)

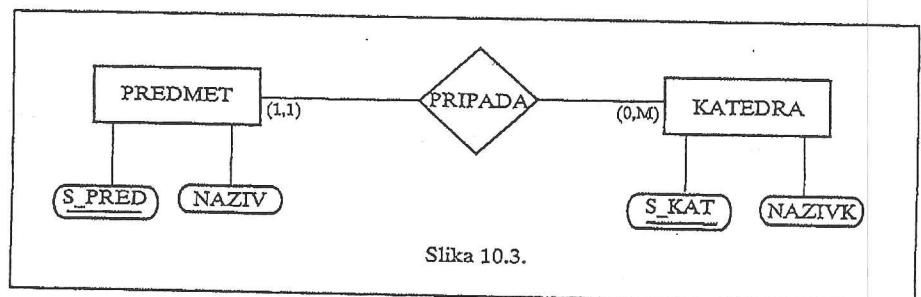
Alternativno šema relacije UPIS može za primarni ključ imati S_KAND umesto BROJ_IND.



Slika 10.2.

2.2. Veze sa kardinalnošću $(1,1) : (0,M)$ i $(1,1) : (1,M)$, ne postaju posebne šeme relacija. Identifikator objekta sa strane za koju je GG = M postaje obeležje šeme relacije koja odgovara objektu sa strane za koju je GG = 1. DOV dat na slici 10.3. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S_PRED, NAZIV, S_KAT) (prema pravilu 1.1 i 2.2)
KATEDRA(S_KAT, NAZIVK) (prema pravilu 1.1).

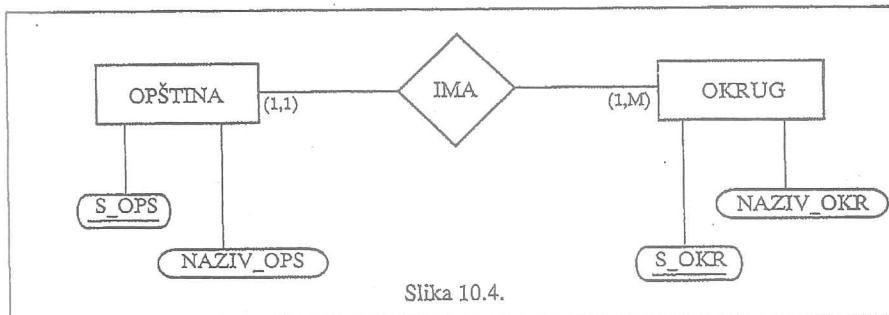


Slika 10.3.

DOV dat na slici 10.4. prevodimo u relacioni model sa sledećim šemama relacija:

OPSTINA(S_OPS, NAZIV_OPS, S_OKR) (prema pravilu 1.1 i 2.2)
OKRUG(S_OKR, NAZIV_OKR) (prema pravilu 1.1).

2.3. Veza izmedju nadredjenog i slabog objekta kao i veza izmedju nadtipa i podtipa ne postaju posebne šeme relacija. One su već ostvarene pravilima 1.2, 1.3 i 1.4.



Slika 10.4.

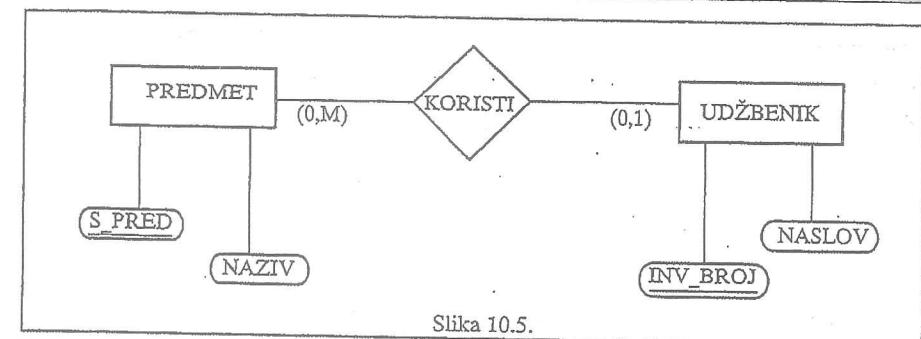
2.4. Veze sa kardinalnošću (0,1) : (0,M) i (0,1) : (1,M), postaju posebna šema relacija. Obeležja ove šeme relacije su identifikatori objekata koji su u vezi, a ključ šeme relacije je identifikator objekta za koji je GG = 1.

DOV dat na slici 10.5. prevodimo u relacioni model sa sledećim šemama relacija:

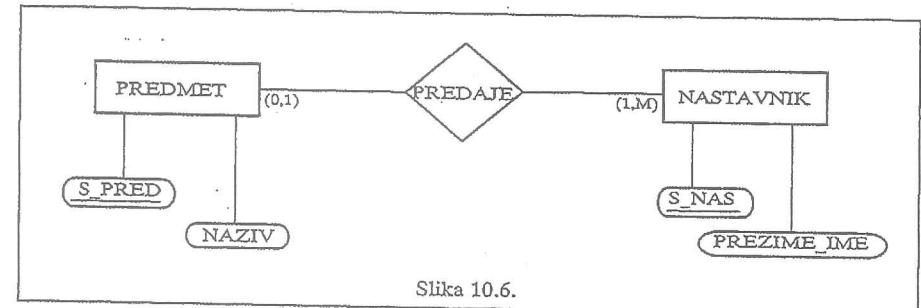
PREDMET(S_PRED, NAZIV) (prema pravilu 1.1)
UDŽBENIK(INV_BROJ, NASLOV)
KORISTI(INV_BROJ, S_PRED) (prema pravilu 2.4)

DOV dat na slici 10.6. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S_PRED, NAZIV) (prema pravilu 1.1)
NASTAVNIK(S_NAS, PREZIME_IME)
PREDAJE(S_PRED, S_NAS) (prema pravilu 2.4)

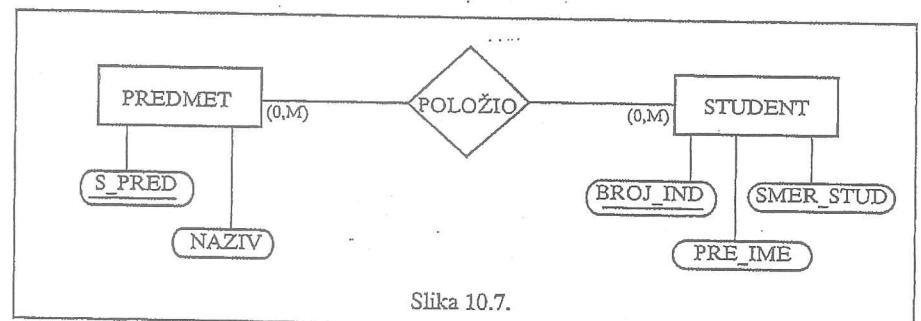


Slika 10.5.



Slika 10.6.

2.5. Veze sa kardinalnošću preslikavanja (0,M) : (0,M), (1,M) : (0,M), i (1,M):(1,M) postaju posebne šeme relacija. Obeležja ove šeme relacije su identifikatori objekata koji su u vezi, a ključ šeme relacije je složeni ključ koji se sastoji od identifikatora objekata koji su u vezi.



Slika 10.7.

DOV dat na slici 10.7. prevodimo u relacioni model sa sledećim šemama relacija:

PREDMET(S_PRED, NAZIV) (prema pravilu 1.1)
 STUDENT(BROJ_IND, PRE_IME, SMER_STUD) (prema pravilu 1.1)
 POLOŽIO(BROJ_IND, S_PRED) (prema pravilu 2.5).

2.6. Agregirani objekat (mešovit tip objekat-veza) se posmatra na isti način kao i odgovarajuća veza. Ukoliko veza poseduje obeležja ista postaju obeležja šeme relacije veze kada se veza prevodi u posebnu šemu relacije ili se uključuju u onu šemu relacije u koju se upisuje spoljni ključ.

DOV dat na slici 7.13. prevodimo u relacioni model sa sledećim šemama relacija:

STUDENT(BROJ_IND, PRE_IME) (prema pravilu 1.1)
 PREDMET(S_PRED, NAZIV) (prema pravilu 1.1)
 NASTAVNIK(S_NAS, PREZIME_IME) (prema pravilu 1.1)
 OCENJUJE(BROJ_IND, S_PRED, S_NAS) (prema pravilu 2.5)
 DIPLOMSKI(BROJ_IND, S_PRED, DATUM_DIPL) (prema pravilu 2.6).

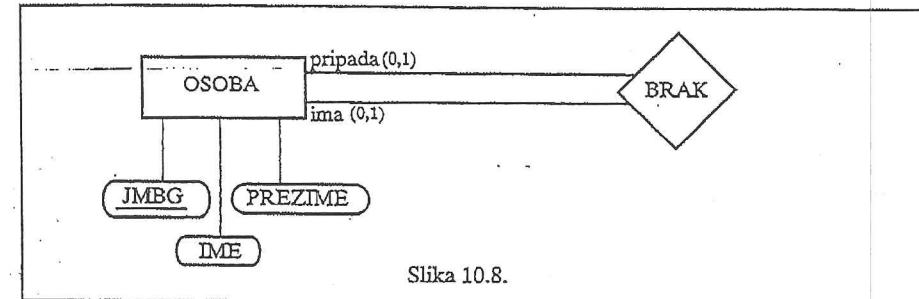
3. Pravila za unarne veze

Prevodjenje unarnih veza (unarnom nazivamo vezu izmedju dva objekata istog tipa) u relacioni model podataka zavisi od kardinalnosti tipa veze i izvodi se kao i za druge tipove ranije opisanih binarnih veza. Napomenimo da kod unarne veze tipa (1:1) parcijalnost samo na jednoj strani veze, odnosno totalnost samo na jednoj strani veze, ne bi imala smisla. Naime, time bi se istom tipu objekta istovremeno dopušтало и poricalo opciono učestvovanje u vezi.

Pri prevodjenju unarnih veza s obzirom da bi spoljni ključ u šemi relacije imao isto ime kao i primarni ključ, vršimo njegovo preimenovanje.

Model podataka dat DOV na slici 10.8. predstavlja situaciju shvatanja braka u našim uslovima. Naime, jedna osoba nije u vezi ni sa jednom drugom osobom (nije u braku), ili je u braku sa samo jednom osobom. Prevodjenjem u relacioni model dobijamo sledeće šeme relacija:

OSOBA(JMBG, IME, PREZIME) (prema pravilu 1.1)
 BRAK(JMBG, JMBG_BRAČNI_DRUG) (prema pravilu 2.1.3)



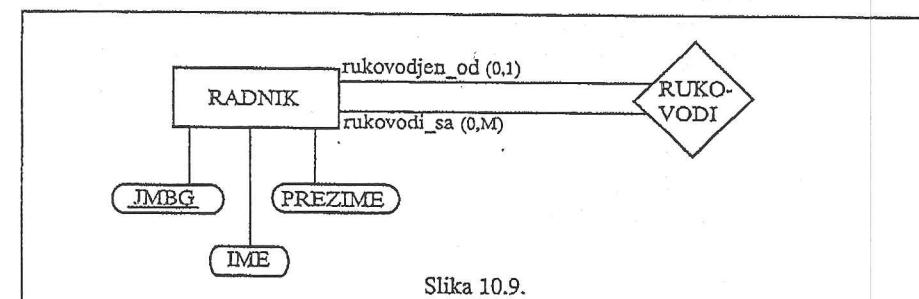
Slika 10.8.

U šemi relacije BRAK za ključ relacije mogli smo odabrat JMBG_BRAČNI_DRUG umesto JMBG.

Model podataka dat DOV na slici 10.9. predstavlja situaciju u kojoj jedan radnik može da rukovodi sa više radnika i može imati jednog nadredjenog rukovodioca. Svaki radnik ne mora imati nadredjenog rukovodioca i svaki radnik ne mora biti rukovodioc.

Prevodjenjem u relationalni model dobijamo sledeće šeme relacija:

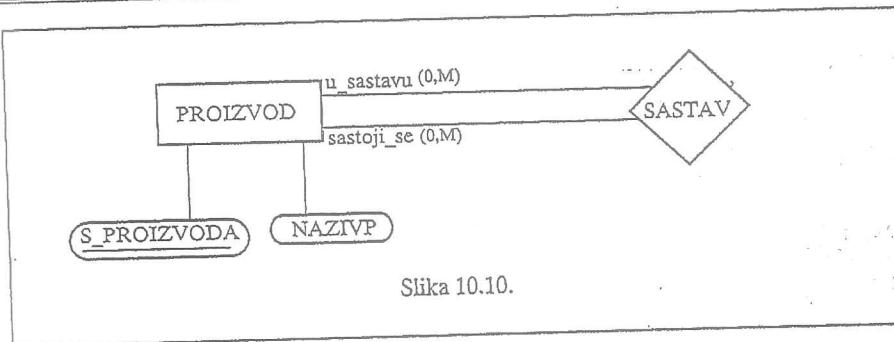
RADNIK(JMBG, IME, PREZIME) (prema pravilu 1.1)
 RUKOVODI(JMBG, JMBG_RUKOVODIOC) (prema pravilu 2.4).



Slika 10.9.

Model podataka dat DOV na slici 10.10. predstavlja model sastavnice u kojem jedan proizvod može sam da predstavlja *sastavljeni objekat* ili se sastoji od jednog ili više objekata koji *ulaze u sastav*.

Prevodjenjem u relationalni model dobijamo sledeće šeme relacija:
 PROIZVOD(S_PROIZVODA, NAZIVP) (prema pravilu 1.1)
 SASTAV(S_PROIZVODA, S_PROIZVODA_S) (prema pravilu 2.5).



Slika 10.10.

Rezime postupka prevodjenja:

Pravila se ne moraju primenjivati redom kako su zadata, već se prevodjenje vrši primenjujući sva odgovarajuća pravila odjednom na pojedine objekte i veze u sistemu, po redosledu koji je obično očigledan iz samog modela. Za svaki objekat se, pored vrste kojoj pripada (pravila 1.1, 1.2, 1.3 i 1.4), utvrđuje i da li će prilikom prevodjenja veze doći do prostiranja ključa ili će se formirati posebna šema relacije.

Kada se na ovaj način postupi sa svim objektima, prevode se preostale veze po pravilima 2.4, 2.5 i 2.6.

Pod pretpostavkom da je DOV korektno urađen njegovim prevodjenjem u relacioni model uvek se dobijaju šeme relacija koje su u trećoj normalnoj formi u kojima ne postoji nula - vrednosti kao rezultat "neprimenjivog svojstva".

Kao što je napred bilo rečeno MOV je semantički bogatiji od relacionog modela podataka. Na primer, sa DOV slike 10.6. se vidi da svaki NASTAVNIK predaje bar jedan PREDMET. Iz odgovarajućih šema relacija u relacionom modelu to se ne može zaključiti. Zbog toga se pri prevodjenju MOV u relacioni model, pored šema relacija koje se dobijaju primenom MOV u relationalni model, pored šema relacija koje se dobijaju primenom navedenih pravila, mora definisati i skup posebnih pravila integriteta, kako bi se zadržala semantika MOV.

Posebna pravila integriteta mogu biti statička i dinamička. Statičkim pravilima integriteta određeni su uslovi koji moraju važiti pre i posle izvršenja bilo koje operacije nad bazom podataka. Dinamičkim pravilima integriteta definisane

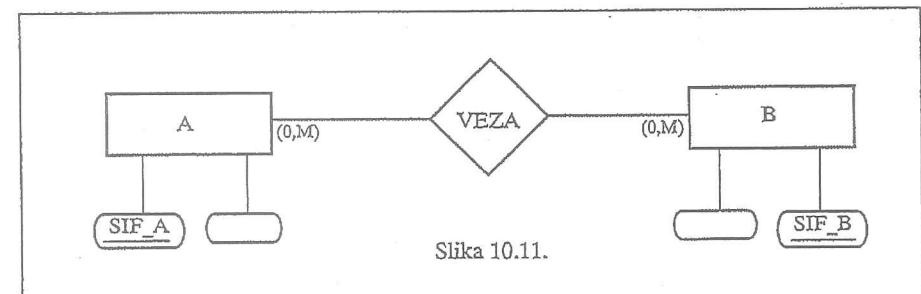
su procedure u relacionom modelu, koje odgovaraju apstraktnim operacijama MOV-a i koje garantuju ostvarenje uslova integriteta.

Do statičkih i dinamičkih pravila integriteta dolazimo posmatranjem jedne veze i objekata koji u njoj učestvuju. Na primer, na slici 10.11. dat je DOV u kome su objekti A i B povezani vezom (0,M):(0,M) parcijalnom sa obe strane. Šeme relacija koje prema pravilima prevodjenja odgovaraju prikazanom DOV su:

A (SIE_A, ...)
B (SIE_B, ...)
VEZA (SIE_A, SIE_B).

Neka je a vrednost obeležja SIF_A, a b obeležja SIF_B u šemi relacije VEZA. Pravilo integriteta za dati tip veze dva objekta može se iskazati zahtevom da vrednost a mora postojati kao vrednost ključa u relaciji A i vrednost b mora postojati kao vrednost ključa u relaciji B. Ovaj integritet odgovara referencijalnom integritetu relacionog modela i može se zapisati na sledeći način:

1. VEZA [SIF_A] \subseteq A [SIF_A],
2. VEZA [SIF_B] \subseteq B [SIF_B].



Slika 10.11.

Kako oba iskaza 1. i 2. moraju biti ispunjena, podrazumevamo da je između njih logički AND operator.

Pored navedenog načina specifikacije statičkog integriteta isti se može iskazati i pomoću naredbi SQL.

SQL naredbe za iskaze 1. i 2. glase:

```
SELECT DISTINCT SIF_A
  FROM VEZA
 IN
SELECT DISTINCT SIF_A
  FROM A;
```

2. SELECT DISTINCT SIF_B
FROM VEZA
IN
SELECT DISTINCT SIF_B
FROM B;

U opštem slučaju, operaciji u MOV ne odgovara operacija u relacionom modelu već procedura. Procedura se sastoji od niza akcija koje su ili izvršne ili proveravaju odredjene uslove. Za operaciju u MOV u proceduri se pojavljuju kao akcije odgovarajuće operacije relacionog modela. Na primer, operaciji MOV UPIŠI objekat... odgovara akcija iskazana operacijom INSERT INTO relacija....

Za predstavljanje procedure tj. specifikaciju dinamičkih uslova integriteta koristićemo SQL.

Za DOV na slici 10.11. dinamički integritet može se specificirati na sledeći način:

UPIŠLA(SIF_A:a)
INSERT INTO A(SIF_A,...)
VALUES (a,...);

BRIŠLA(SIF_A:a)
1. DELETE A
WHERE SIF_A = a;
2. DELETE VEZA
WHERE SIF_A = a;

UPIŠLB(SIF_B:b)
INSERT INTO B(SIF_B,...)
VALUES (b,...);

BRIŠLB(SIF_B:b)
1. DELETE B
WHERE SIF_B = b;
2. DELETE VEZA
WHERE SIF_B = b;

UPIŠI VEZA(SIF_A:a,SIF_B:b)

1. INSERT INTO A(SIF_A)
SELECT a
FROM TEST
WHERE a NOT IN (SELECT DISTINCT SIF_A FROM A);
2. INSERT INTO B(SIF_B)
SELECT b
FROM TEST
WHERE b NOT IN (SELECT DISTINCT SIF_B FROM B);
3. INSERT INTO VEZA(SIF_A, SIF_B,)
VALUES (a, b,);

U operacijama 1. i 2. TEST je pomoćna relacija čija je namena da zadovolji formalnu strukturu SELECT naredbe. Pomoćna tabela TEST ima samo jedno obeležje dužine jednog karaktera i sa samo jednom unetom n-torkom.

Operacijom 1. proverava se da li u relaciji A postoji n-torka sa SIF_A = a, i ukoliko ne postoji, u relaciju A upisuje se takva n-torka. Operacija 2. radi slično sa relacijom B.

Operacija 3. u relaciju VEZA upisuje zahtevanu n-torku.

BRIŠI VEZA(SIF_A:a,SIF_B:b)

DELETE VEZA
WHERE SIF_A = a AND SIF_B = b;

Sledeća statička pravila integriteta koja moraju važiti u relacionom modelu, a koja proizilaze iz DOV su:

Pravilo I1 - integritet objekta. Vrednost primarnog ključa kao celine, i niti jedne njegove komponente ne sme biti jednaka nul - vrednosti.

Pravilo I2 - integritet slabog objekta. Neka je SIF_A obeležje u šemci relacije slabog objekta koja je dobijena primenom pravila prevodjenja 1.2 (prevodjenje identifikaciono zavisnog objekta) i koje odgovara ključu nadredjenog objekta. Tada obeležje SIF_A ne može dobiti neku vrednost (npr. a), ako ta vrednost a ne postoji kao vrednost ključa u relaciji nadredjenog objekta.

za S_DRZ ako se ta ista vrednost u relaciji podtipa ne može da postoji neka ista vrednost ne javlja kao vrednost ključa u relaciji podtipa.

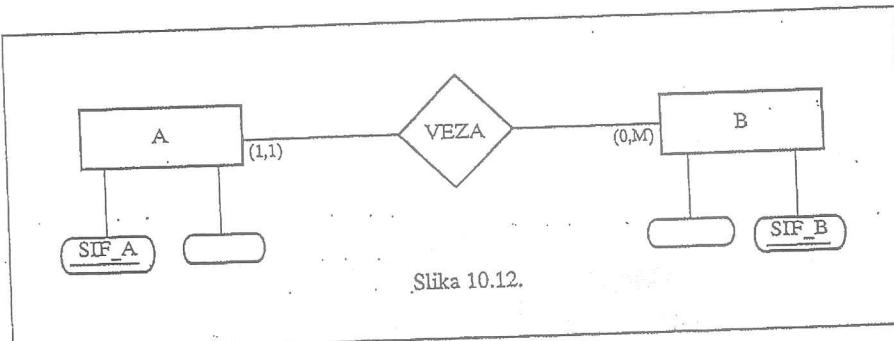
Na primer, za DOV dat na slici 7.9. statički uslov integriteta glasi:

NESTAVNIK [JMBG] \subseteq RADNIK [JMBG],
SLUŽBENIK [JMBG] \subseteq RADNIK [JMBG],
ODRŽAVANJE [JMBG] \subseteq RADNIK [JMBG].

Pravilo I4 - integritet veze 1:M.

I4.1 - veze sa kardinalnošću (1,1) : (0,M) slika 10.12.

Ne može postojati neka vrednost za SIF_B (npr. b) u relaciji A, ako se ta ista vrednost b ne javlja kao vrednost ključa u relaciji objekta B.



Na primer, za DOV dat na slici 10.12. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

PREDMET [S_KAT] \subseteq KATEDRA [S_KAT].

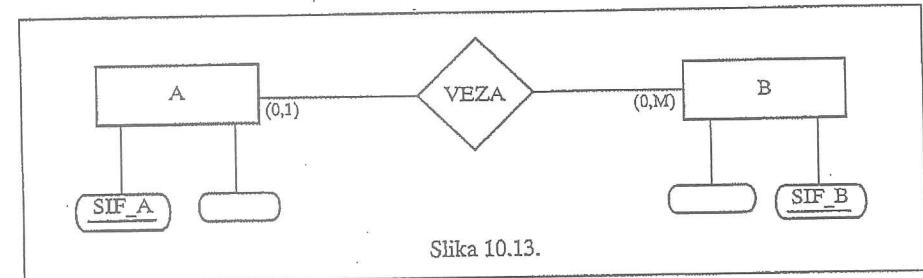
I4.2 - veza sa kardinalnošću (0,1) : (0,M) slika 10.13.

Neka vrednost obeležja SIF_A (npr. a) i neka vrednost obeležja SIF_B (npr. b) u relaciji koja odgovara vezi tih objekata ne mogu da postoje ako ne postoji ista vrednost a obeležja SIF_A u relaciji objekta A, odnosno ista vrednost b obeležja SIF_B u relaciji objekta B.

Na primer, za DOV dat na slici 10.13. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

A (SIF_A, ...)	(prema pravilu 1.1)
B (SIF_B, ...)	(prema pravilu 1.1)
VEZA (SIF_A, SIF_B)	(prema pravilu 2.4)

VEZA [SIF_A] \subseteq A [SIF_A],
VEZA [SIF_B] \subseteq B [SIF_B].



Na primer, za DOV dat na slici 10.5. statički uslovi integriteta su:

KORISTI [INV_BROJ] \subseteq UDŽBENIK [INV_BROJ],
KORISTI [S_PRED] \subseteq PREDMET [S_PRED].

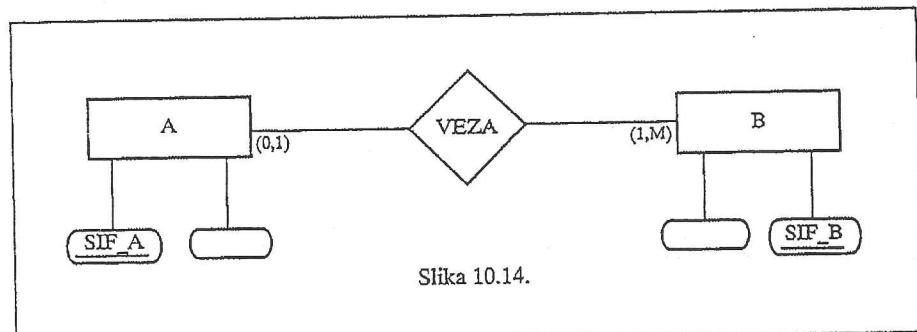
neka vrednost obeležja SIF_B (npr. b), ako se ta ista vrednost b obeležja SIF_B ne javlja i u veznoj relaciji. Radi se o slučaju *uzajamnih integriteta* koji zahteva istovremeno ažuriranje relacije objekta B i vezne relacije.

Pored ovoga, neka vrednost obeležja SIF_A (npr. a) u relaciji koja odgovara vezi ne može da postoji, ako u relaciji A ne postoji ista vrednost a obeležja SIF_A.

Na primer, za DOV dat na slici 10.14. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

A (SIF_A, ...)	(prema pravilu 1.1)
B (SIF_B, ...)	(prema pravilu 1.1)
VEZA (SIF_A, SIF_B)	(prema pravilu 2.4)

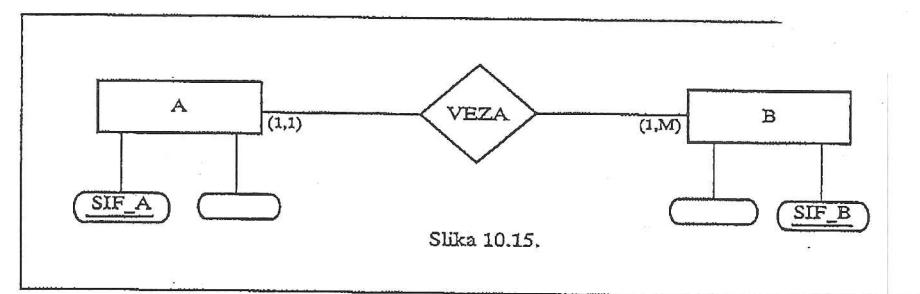
VEZA [SIF_B] = B [SIF_B],
VEZA [SIF_A] ⊆ A [SIF_A].



Na primer, za DOV dat na slici 10.6. statički uslovi integriteta su:

PREDAJE [S_NAS] = NASTAVNIK [S_NAS],
PREDAJE [S_PRED] ⊆ PREDMET [S_PRED].

Na primer, za DOV dat na slici 10.15. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:



Slika 10.15.

Na primer, za DOV dat na slici 10.15. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:

A (SIF_A, SIF_B ...)	(prema pravilu 1.1 i 2.2)
B (SIF_B, ...)	(prema pravilu 1.1)

A [SIF_B] = B [SIF_B].

Na primer, za DOV dat na slici 10.4. statički uslov integriteta glasi:

OKRUG [S_OKR] = OPSTINA [S_OKR].

Pravilo 15 - integritet veze za koju oba preslikavanja za gornju granicu imaju vrednost GG = M .

Neka je SIF_A ključ jedne, a SIF_B ključ druge šeme relacije objekata koji su u vezi. Ključ odgovarajuće vezne šeme relacije je složen ključ SIF_A,SIF_B.

I5.1 - veza sa kardinalnošću (0,M) : (0,M) slika 10.11.

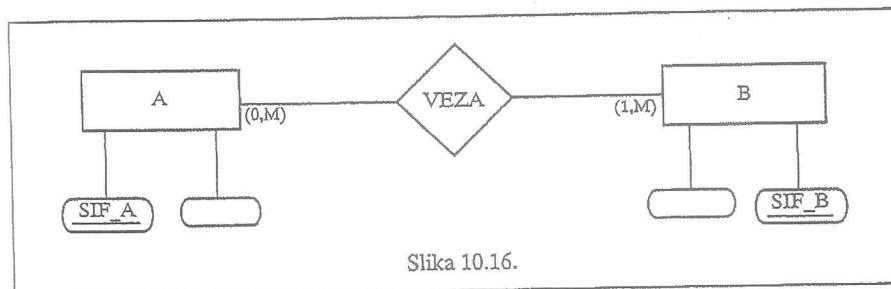
Neka je a vrednost obeležja SIF_A a, b obeležja SIF_B u veznoj relaciji. Tada ista ta vrednost a mora postojati kao vrednost ključa u relaciji A i ista ta vrednost b mora postojati kao vrednost ključa u relaciji B.

Na primer, za DOV dat na slici 10.7. statički uslovi integriteta su:

$$\begin{aligned} \text{POLOŽIO [S_PRED]} &\subseteq \text{PREDMET [S_PRED]}, \\ \text{POLOŽIO [BROJ_IND]} &\subseteq \text{STUDENT [BROJ_IND]}. \end{aligned}$$

I5.2 - veza sa kardinalnošću (0,M) : (1,M) slika 10.16.

U veznoj relaciji, ne može postojati vrednost obeležja SIF_A (npr. a) ako ta ista vrednost a ne postoji kao vrednost ključa SIF_A u relaciji objekta A. Isto tako, ne može, u veznoj relaciji, postojati vrednost obeležja SIF_B (npr. b) ako ta ista vrednost ne postoji u relaciji objekta B, kao ključ, i istovremeno, ne može u relaciji objekta B postojati neka vrednost obeležja SIF_B (npr. b) ako ta ista vrednost b ne postoji kao deo ključa u veznoj relaciji. Očigledno radi se o uzajamnom integritetu, odnosno biće potrebno istovremeno ažuriranje vezne relacije i relacije objekta B.



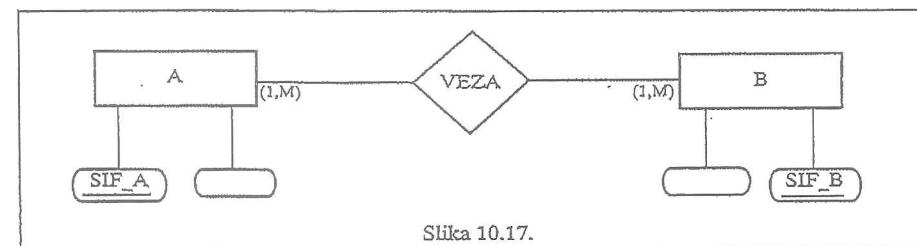
Na primer, za DOV dat na slici 10.16. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{aligned} A (\text{SIF_A}, \dots) && (\text{prema pravilu 1.1}) \\ B (\text{SIF_B}, \dots) && (\text{prema pravilu 1.1}) \\ \text{VEZA} (\text{SIF_A}, \text{SIF_B}) && (\text{prema pravilu 2.5}) \end{aligned}$$

$$\begin{aligned} \text{VEZA} [\text{SIF_A}] &\subseteq A [\text{SIF_A}], \\ \text{VEZA} [\text{SIF_B}] &\subseteq B [\text{SIF_B}]. \end{aligned}$$

I5.3 - veza sa kardinalnošću (1,M) : (1,M) slika 10.17.

Neka vrednost a obeležja SIF_A, u veznoj relaciji, ne može da postoji ako ta ista vrednost ne postoji kao vrednost ključa u relaciji odgovarajućeg objekta, i istovremeno, ne može postojati vrednost a ključa SIF_A u relaciji objekta A, ako se ta ista vrednost ne pojavljuje kao vrednost dela ključa SIF_A u veznoj relaciji. Isto tako, ne može da postoji, u veznoj relaciji neka vrednost b obeležja SIF_B ako ta ista vrednost ne postoji kao ključ u relaciji objekta B, i istovremeno, ne može postojati vrednost b ključa SIF_B u relaciji objekta B, ako ta ista vrednost b nije vrednost dela ključa SIF_B u veznoj relaciji. Ovde se radi o *dvostrukom uzajamnom integritetu*, odnosno potrebno je istovremeno ažuriranje sve tri relacije.



Slika 10.17.

Na primer, za DOV dat na slici 10.17. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{aligned} A (\text{SIF_A}, \dots) && (\text{prema pravilu 1.1}) \\ B (\text{SIF_B}, \dots) && (\text{prema pravilu 1.1}) \\ \text{VEZA} (\text{SIF_A}, \text{SIF_B}) && (\text{prema pravilu 2.5}) \end{aligned}$$

$$\begin{aligned} \text{VEZA} [\text{SIF_A}] &= A [\text{SIF_A}], \\ \text{VEZA} [\text{SIF_B}] &= B [\text{SIF_B}]. \end{aligned}$$

Pravilo 16 - integritet 1:1 veze.

I6.1 - veza sa kardinalnošću (1,1) : (1,1).

U jedinstvenoj šemi relacije koja se dobija prevodenjem i koja predstavlja objekte u vezi i samu vezu ni SIF_A ni SIF_B, kao kandidati za ključ, ne smeju

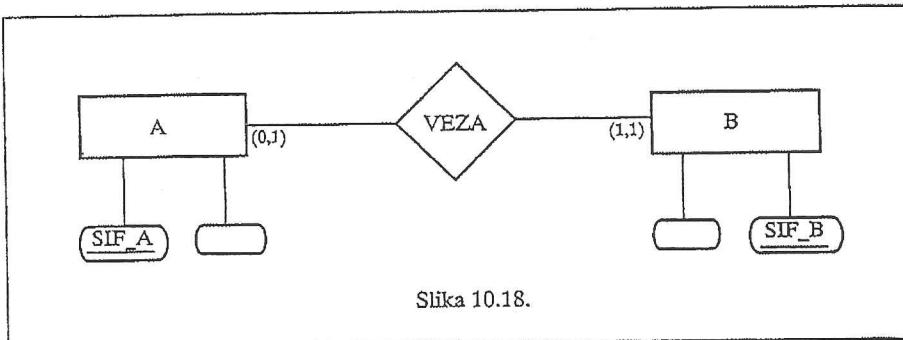
da imaju nul-vrednost. Radi se o proširenju pravila integriteta I1 i na kandidate za ključ.

I6.2 - veza sa kardinalnošću (0,1) : (1,1) slika 10.18.

U relaciji objekta B, ne može postojati vrednost obeležja SIF_A (npr. a) ako ta ista vrednost a ne postoji kao vrednost ključa u relaciji objekta A.

Na primer, za DOV dat na slici 10.18. prevodjenje u relacioni model sa statičkim uslovom integriteta glasi:

$$\begin{array}{ll} A (\text{SIF_A}, \dots) & \text{(prema pravilu 1.1)} \\ B (\text{SIF_B}, \text{SIF_A} \dots) & \text{(prema pravilu 1.1 i 2.1.2)} \\ \\ B [\text{SIF_A}] \subseteq A [\text{SIF_A}]. & \end{array}$$

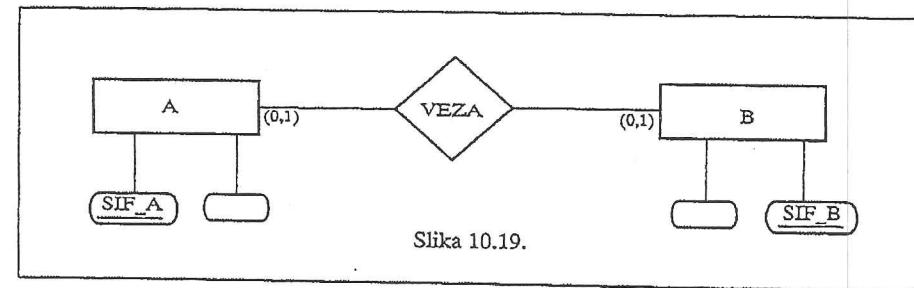


Na primer, za DOV dat na slici 10.1. statički uslov integriteta glasi:

$$\text{R_JEDINICA} [\text{JMBG}] \subseteq \text{RADNIK} [\text{JMBG}].$$

I6.3 - veza sa kardinalnošću (0,1) : (0,1) slika 10.19.

Neka vrednost obeležja SIF_A (npr. a) ne može postojati u veznoj relaciji ako ta ista vrednost a ne postoji kao vrednost ključa u relaciji objekta A. Isto tako vrednost obeležja SIF_B (npr. b), ne može da postoji u veznoj relaciji ako ta ista vrednost b ne postoji kao vrednost ključa u relaciji objekta B.



Na primer, za DOV dat na slici 10.19. prevodjenje u relacioni model sa statičkim uslovima integriteta glasi:

$$\begin{array}{ll} A (\text{SIF_A}, \dots) & \text{(prema pravilu 1.1)} \\ B (\text{SIF_B}, \dots) & \text{(prema pravilu 1.1)} \\ \text{VEZA} (\text{SIF_A}, \text{SIF_B}) & \text{(prema pravilu 2.1.3)} \end{array}$$

$$\begin{array}{l} \text{VEZA} [\text{SIF_A}] \subseteq A [\text{SIF_A}], \\ \text{VEZA} [\text{SIF_B}] \subseteq A [\text{SIF_B}]. \end{array}$$

Na primer, za DOV dat na slici 10.2. statički uslovi integriteta su:

$$\begin{array}{l} \text{UPIS} [\text{S_KAND}] \subseteq \text{KANDIDAT} [\text{S_KAND}], \\ \text{UPIS} [\text{BROJ_IND}] \subseteq \text{STUDENT} [\text{BROJ_IND}]. \end{array}$$

Kada su odredjeni uslovi integriteta za svaku vezu i objekte koji je čine vrši se objedinjavanje uslova integriteta za one objekte koji učestvuju u više veza.

II. OSNOVE OBRADE TRANSAKCIJA

Transakcija je vremenski uredjeni niz nedeljivih radnji nad bazom podataka koje u celini ne remete uslove integriteta. Transakcija predstavlja *logičku jedinicu* rada nad bazom podataka. Sa aspekta definisanih uslova integriteta transakcija transformiše jedno konzistentno stanje baze podataka u drugo takodje konzistentno stanje baze podataka. Kao što ćemo videti iz primera koji će uslediti između ova dva konzistentna stanja dozvoljeno je da se baza podataka nadje i u nekonzistentnom stanju. Bitno je da transakcija bude tako označena da iz bilo kojih razloga (npr. otkaz hardvera ili softvera, višekorisnički rad) ne ostavi trajno bazu podataka u nekonzistentnom stanju.

Razmotrimo primer transakcije na sledećem pojednostavljenom modelu podataka:

Šeme relacija:

PREDMET (S_PRED, NAZIV),
PREDAJE (S_NAS, S_PRED, BR_GR),

Uslov integriteta:

PREDAJE [S_PRED] \subseteq PREDMET [S_PRED].

Zahtev: Promeniti šifru predmeta "INFORMACIONI SISTEMI" koja glasi 001 na šifru 006.

Upotreboom notacije jezika SQL možemo definisati sledeći niz naredbi koje treba da realizuju prethodni zahtev:

```
UPDATE PREDMET  
    SET S_PRED = 006  
    WHERE S_PRED = 001;  
UPDATE PREDAJE  
    SET S_PRED = 006  
    WHERE S_PRED = 001;
```

Smisao prethodnog primera je u tome što zahtev promene šifre predmeta "INFORMACIONI SISTEMI" sa tekuće vrednosti 001 na novu 006 od strane korisnika se posmatra kao jedinstvena operacija koja zahteva dve operacije UPDATE nad bazom podataka. Izmedju ove dve operacije UPDATE baza podataka može se naći i u nekonzistentnom stanju. Posle izvršavanja prve naredbe UPDATE uslovi integriteta ne važe. Posle izvršavanja druge naredbe UPDATE uslovi integriteta ponovo važe. Sa tačke gledišta uslova integriteta transakcija se ponaša kao jedinična radnja što ne važi za pojedinačne radnje od kojih se transakcija sastoji. Primetimo da ako se u prethodnoj transakciji redosled UPDATE operacija promeni baza podataka se takođe privremeno dovodi u nekonzistentno stanje.

Jasno je iz prethodnog primera da ne smemo dozvoliti da se od dve operacije UPDATE transakcije izvrši samo jedna, a druga ne. U tom slučaju baza podataka ostala bi trajno u nekonzistentnom stanju. Mi bi smo želeli biti sigurni da će obe operacije UPDATE biti izvršene bez obzira na mogućnosti pojave greške i to u najnepovoljnijem trenutku za transakciju. Na primer, greška u hardveru ili softveru računarskog sistema može se dogoditi izmedju dve operacije UPDATE. Računarski sistemi koji podržavaju *transakcionu obradu* moraju da garantuju da, ukoliko se transakcijom vršilo ažuriranje baze podataka i iz bilo kojih razloga transakcija se nije normalno završila, poništite sva ažuriranja delimično izvršenih transakcija. Na taj način transakcija se ili u potpunosti izvršava ili se u potpunosti poništavaju njena dejstva. Transakciona obrada odvija se pod kontrolom *administratora transakcija* kao dela programske podrške čije naredbe COMMIT i ROLLBACK u potpunosti određuju način njegovog funkcionisanja.

Naredba COMMIT signalizira administratoru transakcija o uspešnom završetku transakcije. Baza podataka ponovo se nalazi u konzistentnom stanju i sva dejstva transakcije (privremena ažuriranja) mogu biti trajno preneta na bazu podataka. Nasuprot, naredba ROLLBACK signalizira administratoru transakcija o neuspešnom završetku transakcije. U tom slučaju potrebno je sva dejstva nad bazom podataka takve transakcije poništiti. Naredbe COMMIT i ROLLBACK se zadaju direktno ili automatski od strane SUBP.

Da bi se mehanizam trajnog prenošenja ažuriranja na bazu podataka ili poništavanja ažuriranja mogao uspešno i korektno realizovati potrebno je čuvati informacije o bazi podataka pre i posle ažuriranja. Da bi se znao vremenski trenutak od kojeg počinje čuvanje informacija i kada prestaje, nad bazom podataka postavljaju se *tačke sinhronizacije*.

Tačka sinhronizacije predstavlja graničnu tačku izmedju dve serijske transakcije. To je tačka u kojoj se baza podataka nalazi u konzistentnom stanju.

Naredbe COMMIT i ROLLBACK kao što je bilo napomenuto ne moraju biti direktno zadate.

Naredba COMMIT jezika SQL ima sledeći opšti oblik:

COMMIT [WORK];

Naredba COMMIT signalizira uspešno izvršavanje transakcije i postavlja tačku sinhronizacije. Sva ažuriranja baze podataka izvršena datim programom od prethodne tačke sinhronizacije postaju trajna.

Neobavezni element WORK nema posebnog značaja u naredbi.

Naredba ROLLBACK jezika SQL ima sledeći opšti oblik:

ROLLBACK [WORK];

Naredba ROLLBACK signalizira neuspešno izvršavanje transakcije i postavlja tačku sinhronizacije. Sva ažuriranja baze podataka od prethodne tačke sinhronizacije, od strane datog programa se poništavaju.

Neobavezni element WORK nema posebnog značaja u naredbi.

Iz navedenih razmatranja sledi da transakcije ne mogu biti uložene jedna u drugu obzirom da naredbe COMMIT ili ROLLBACK završavaju jednu i iniciraju drugu transakciju. Kao posledica sledi da se jedan program sastoji od jedne ili više serijskih transakcija.

Administrator transakcija mora garantovati da će posle uspešnog izvršavanja naredbe COMMIT od strane transakcije sva ažuriranja nad bazom podataka postati trajna, mada je moguće da se u sledećem trenutku dogodi otkaz računarskog sistema. Na primer, otkaz sistema nastaje u trenutku posle izdavanja naredbe COMMIT ali pre nego što ažuriranja budu fizički zapisana u bazu podataka.

Ažuriranja koja se u trenutku otkaza nalaze u baferima operativne memorije na taj način mogla bi biti izgubljena. Ako se to dogodi procedura restarta računarskog sistema treba na osnovu žurnala (podataka o izvršenim promenama na bazi podataka) ipak da izvrši trajno prenošenje ažuriranja na bazu podataka. Sledi da podaci o žurnalnu trebaju biti trajno memorisani pre izdavanja naredbe COMMIT.

II.1. Paralelno izvršavanje transakcija

Sistem za upravljanje bazom podataka treba da omogući paralelni rad većem broju korisnika nad istom bazom podataka što znači paralelno izvršavanje više transakcija. Da bi to bilo moguće potrebno je obezbediti adekvatno upravljanje paralelnim izvršavanjem transakcija kako transakcije ne bi nepoželjno mešale svoja dejstva jedna na drugu.

U osnovi postoje tri slučaja kada je transakcija samostalno korektna, ali zbog mešanja transakcija može nastati pogrešan rezultat ukoliko ne postoji odgovarajući mehanizam upravljanja. Radi se o situacijama u kojima dolazi do mešanja operacija iz dve korektne transakcije pri čemu je rezultat nekorekstan.

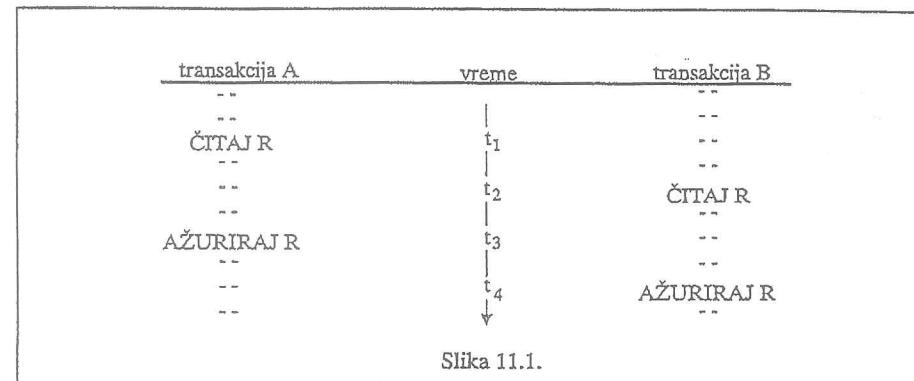
Na prethodno ukazane probleme odnose se:

1. Izgubljena ažuriranja.
2. Zavisnosti od privremenih ažuriranja.
3. Narušavanje serijabilnosti.

Problemi izgubljenog ažuriranja

Razmotrimo situaciju priказанu na slici 11.1. koju možemo interpretirati na sledeći način:

Transakcija A čita zapis R u trenutku t_1 . Transakcija B čita isti zapis R u trenutku t_2 . Transakcija A ažurira zapis R u trenutku t_3 na osnovu vrednosti pročitanih u trenutku t_1 . Transakcija B ažurira zapis R u trenutku t_4 na osnovu vrednosti pročitanih u trenutku t_2 koje su iste sa vrednostima pročitanim u trenutku t_1 od strane transakcije A. Ažuriranje od strane transakcije A biva poništeno u trenutku t_4 jer transakcija B ne uzima u obzir ažuriranja od strane transakcije A.

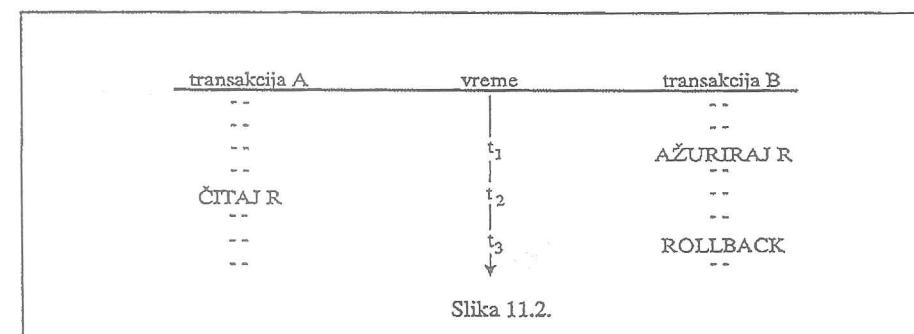


Slika 11.1.

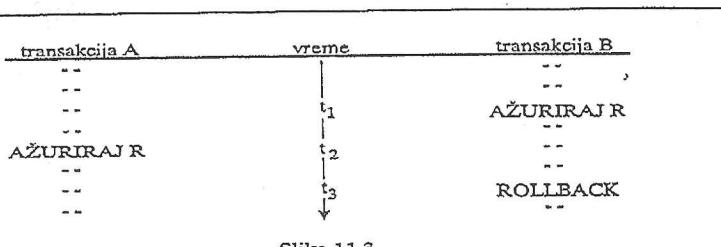
Problem zavisnosti od privremenih ažuriranja

Problem povezan sa zavisnošću od privremenih ažuriranja nastaje u slučaju kada se transakciji A dozvoljava čitanje ili što je još lošije ažuriranje zapisa koji je privremeno ažuriran od transakcije B pri čemu taj zapis još nije trajno ažuriran od transakcije B. S obzirom na činjenicu da zapis nije trajno ažuriran od transakcije B, postoji mogućnost da transakcija B poništa svoja privremena ažuriranja. Nastaje situacija u kojoj je transakcija A izvršila neke operacije na bazi podataka koji više ne postoje odnosno kao da nikada nisu ni postojali u bazi podataka.

Opisanu situaciju ilustruju slike 11.2 i 11.3. Transakcija A sa slike 11.2 "vidi" u trenutku t_2 privremeno ažurirane podatke zapisa R. Transakcija B u trenutku t_3 poništava svoja ažuriranja zapisa R. Očigledno je da je transakcija A izvršena na osnovu podataka u trenutku t_2 koji mogu biti različiti od onih u trenutku t_1 , što znači da i rezultat transakcije A može biti pogrešan.



Slika 11.2.



Slika 11.3.

Napomenimo da poništavanje transakcije B ne mora biti uzrokovano nekom greškom u B. Uzrok poništavanja transakcije B može biti na primer, otkaz sistema u trenutku kada se transakcija A u potpunosti završila, a transakcija B nije. Kao posledica otkaza sistema izdaće se naredba za poništavanje samo onih transakcija koje su bile u toku u trenutku otkaza odnosno poništavanje samo transakcije B.

Situacija sa slike 11.3. je još lošija. Transakcija A ne samo da je zavisna od privremeno ažiriranih podataka transakcije B već se njena ažuriranja zapisa R u trenutku t_2 poništavaju od strane transakcije B u trenutku t_3 koja zapisu R vraća stanje pre trenutka t_1 .

Problem narušavanja serijabilnosti

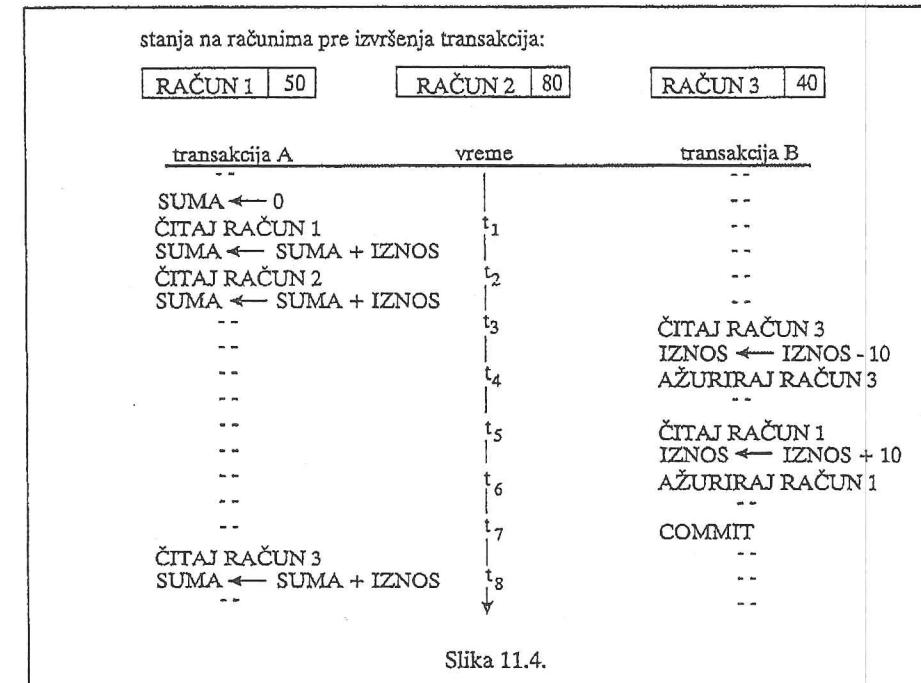
Razmotrimo situaciju na slici 11.4. na kojoj su prikazane dve transakcije A i B koje se izvršavaju nad zapisima tekućih računa gradjana. Transakcija A sumira stanja na računima. Transakcija B prenosi iznos 10 sa računa 3 na račun 1. Očigledno da je dobijeni rezultat 160 od transakcije A nekorektan. Ako bi transakcija A takav rezultat upisala u bazu podataka ista bi trajno ostala u protivrečnom stanju.

Uočimo razliku datog primera od problema zavisnosti transakcije A od privremenih ažuriranja iz ranijeg primera. Ovde transakcija B svoja ažuriranja trajno prenosi na bazu podataka pre nego što transakcija A čita zapis računa 3.

11.2. Rešavanje problema paralelnog izvršavanja transakcija

Uobičajeni način rešavanja problema paralelnog izvršavanja transakcija su *protokoli zaključavanja*. Osnovna ideja zaključavanja sastoji se u tome da

transakcija koja zahteva garancije da neki objekat (obično zapis baze podataka) za koji je ona zainteresovana ne može biti ažuriran u nekom vremenu. Takav objekat transakcija zaključava i sprečava njegovo ažuriranje od strane drugih transakcija. Na taj način transakcija raspolaže *stabilnim* podacima o objektu koliko ona to zahteva.



Slika 11.4.

Pre detaljnijih razmatranja principa delovanja zaključavanja radi pojednostavljenja polazimo od sledećih prepostavki:

1. Kao objekt podvrgnut protokolu zaključavanja može se pojaviti zapis baze podataka odnosno n-torka bazne tabele.
2. Biće razmatrana dva oblika protokola zaključavanja:
 - ekskluzivna zaključavanja (E) - omogućava transakciji koja je n-torku zaključala, čitanje i promenu vrednosti iste,
 - deljiva zaključavanja (D) - omogućava transakciji koja je n-torku zaključala, samo njeno čitanje.
3. Posmatraju se samo operacije na nivou zapisa (ČITAJ, AŽURIRAJ itd.). Operacije zaključavanja na nivou skupova mogu se posmatrati kao niz operacija na nivou zapisa.

Razmotrimo sada detaljnije protokol zaključavanja.

1. Ako transakcija A izvrši ekskluzivno zaključavanje (tip E) zapisa R, tada će zahtev za zaključavanje, bilo koje vrste, iz transakcije B preći u stanje čekanja. Transakcija B biće u stanju čekanja sve dok zapis ne bude otključan (deblokiran) od strane transakcije A.
2. Ako transakcija A izvrši deljivo zaključavanja (tip D) zapisa R tada:
 - a) Zahtev od transakcije B na zaključavanje tipa E zapisa R preći će u stanje čekanja i ostati u tom stanju sve dok transakcija A ne izvrši otključavanje zapisa R;
 - b) Zahtev od transakcije B na zaključavanje tipa D zapisa R biće izvršeno tj. i transakcija A i B mogu istovremeno imati zaključan zapis R.

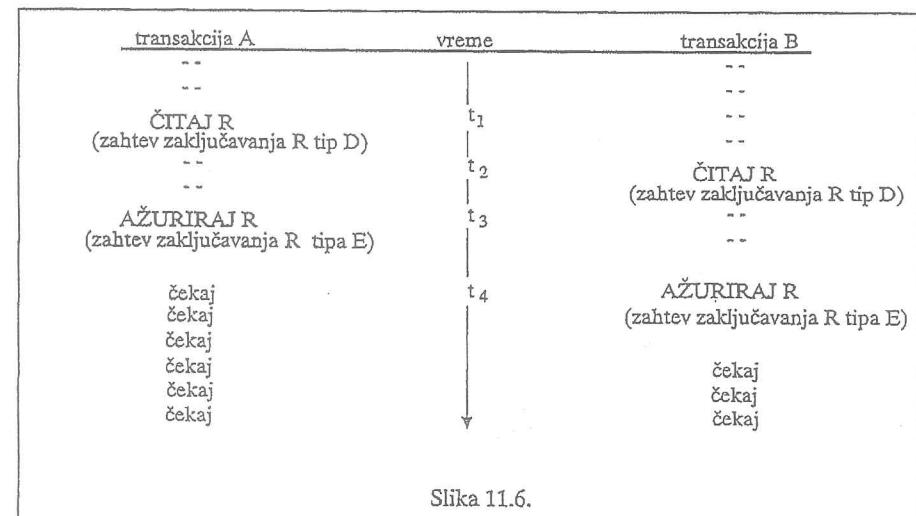
Prethodni zahtevi zaključavanja zapisa R od strane transakcija A i B mogu se rezimirati matricom uskladjenosti tipova zaključavanja koja je data na slici 11.5. i koju možemo interpretirati na sledeći način. Posmatrajmo neki zapis R. Pretpostavimo da u sadašnjem trenutku transakcija A ima zaključavanje zapisa R označeno u zagлавju tabele (crtica znači nezaključan zapis). Neka transakcija B zahteva zaključavanje zapisa R prema tipu označenom u prvom stupcu tabele. Oznaka N na preseku vrste i kolone matrice ukazuje na nesaglasnost zahteva transakcije A i B odnosno zahtev transakcije B neće biti izvršen i B prelazi u stanje čekanja. Oznaka Y ukazuje na saglasnost zahteva transakcije A i B što znači da će zahtev transakcije B biti izvršen.

transakcija A			
transakcija B	E	D	-
E	N	N	Y
D	N	Y	Y
-	Y	Y	Y

Slika 11.5.

3. Zahtevi transakcije za zaključavanje zapisa obično nisu direktni već se iniciraju automatski sa zahtevanjem izvršavanja određenih operacija: Da bi transakcija uspešno izvršila operaciju čitanja zapisa (ČITAJ R) ona prethodno zahteva zaključavanje tipa D na zapisu R. Ako je zaključavanje uspešno tada je i čitanje uspešno. Ako se zaključavanje ne može izvršiti tada ni čitanje nije moguće, a transakcija prelazi u stanje čekanja, odnosno u stanje pokušavanja da se zaključavanje-čitanje uspešno izvrši. Kada transakcija zahteva izvršavanje operacije ažuriranja (AŽURIRAJ R) ona prvo zahteva zaključavanje zapisa tipa E. Slično kao i kod operacije čitanja ako je zaključavanje uspešno tada je i ažuriranje uspešno. Ako se zaključavanje ne može izvršiti tada ni ažuriranje nije moguće, a transakcija prelazi u stanje čekanja, odnosno u stanje pokušavanja da se zaključavanje-ažuriranje uspešno izvrši. Ako je transakcija već izvršila zaključavanje zapisa (kao što to i proizilazi u slučaju sekvence operacija ČITAJ ... AŽURIRAJ) tada će operacija AŽURIRAJ pojačati zaključavanje sa tipa D na tip E.
4. Zaključavanja tipa E zadržavaju se do sledeće tačke sinhronizacije dok se zaključavanja tipa D takođe obično zadržavaju do sledeće tačke sinhronizacije, ali mogu biti ukinuta i ranije.

Uzimajući u obzir zaključavanje zapisa sada možemo ponovo razmotriti načine rešavanja problema paralelnog izvršavanja transakcija.



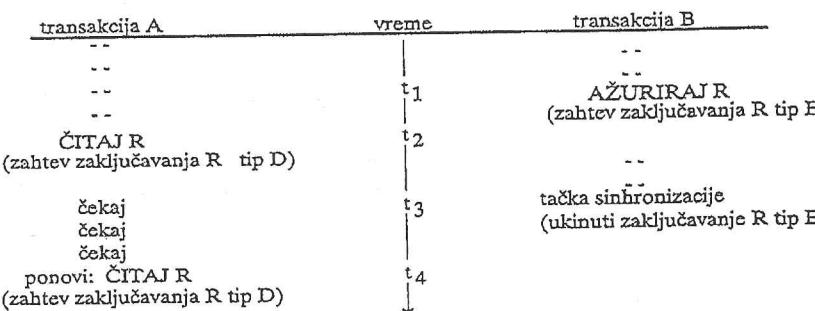
Problemi izgubljenog ažuriranja

Slika 11.6. predstavlja modifikovanu varijantu sa slike 11.1 uzimajući u obzir paralelno izvršavanje transakcija A i B sa mehanizmom zaključavanja zapisa.

Uočavamo da zahtev transakcije A u trenutku t_3 za ažuriranje zapisa R prelazi u stanje čekanja zbog toga što se ne može izvršiti zaključavanje tipa E na zapisu R koji je već zaključan od transakcije B prema tipu D. Zbog toga transakcija A prelazi u stanje čekanja. Analogno u trenutku t_4 transakcija B prelazi u stanje čekanja. Nijedna od transakcija ne može biti nastavljena i ne može doći do gubljenja ažuriranja koja smo ranije imali. Očigledno da smo rešavanjem jednog problema stvorili drugi problem koji nazivamo **medjusobno blokiranje transakcija**. Rešavanje problema medjusobnog blokiranja transakcija biće razmatrano kasnije.

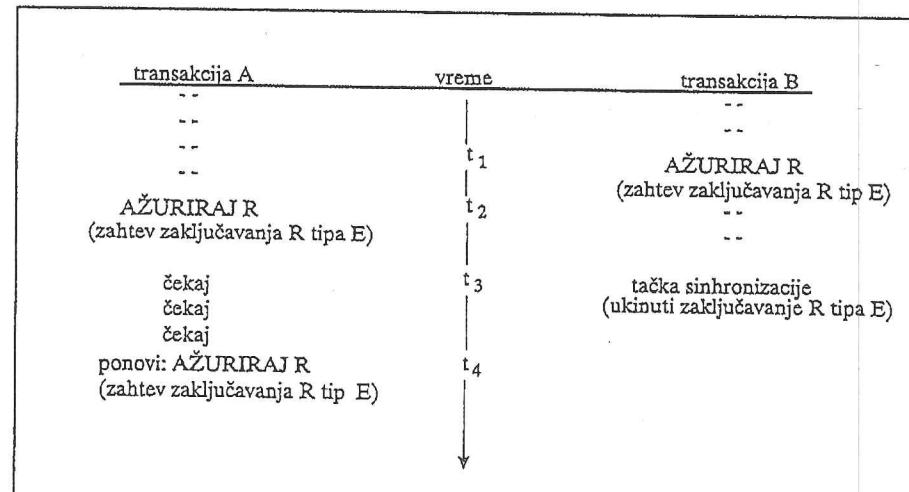
Problem zavisnosti od privremenih ažuriranja

Uzimajući u obzir mehanizam zaključavanja pri paralelnom izvršavanju transakcija situacija sa slike 11.2. i 11.3. menja se u situaciju ilustrovanu na slikama 11.7. i 11.8. Operacije ČITAJ R odnosno AŽURIRAJ R od strane transakcije A u trenutku t_2 neće biti izvršene zato što su njihovi zahtevi na zaključavanje u konfliktu sa zaključavanjima koja su već postavljena od strane transakcije B.



Slika 11.7.

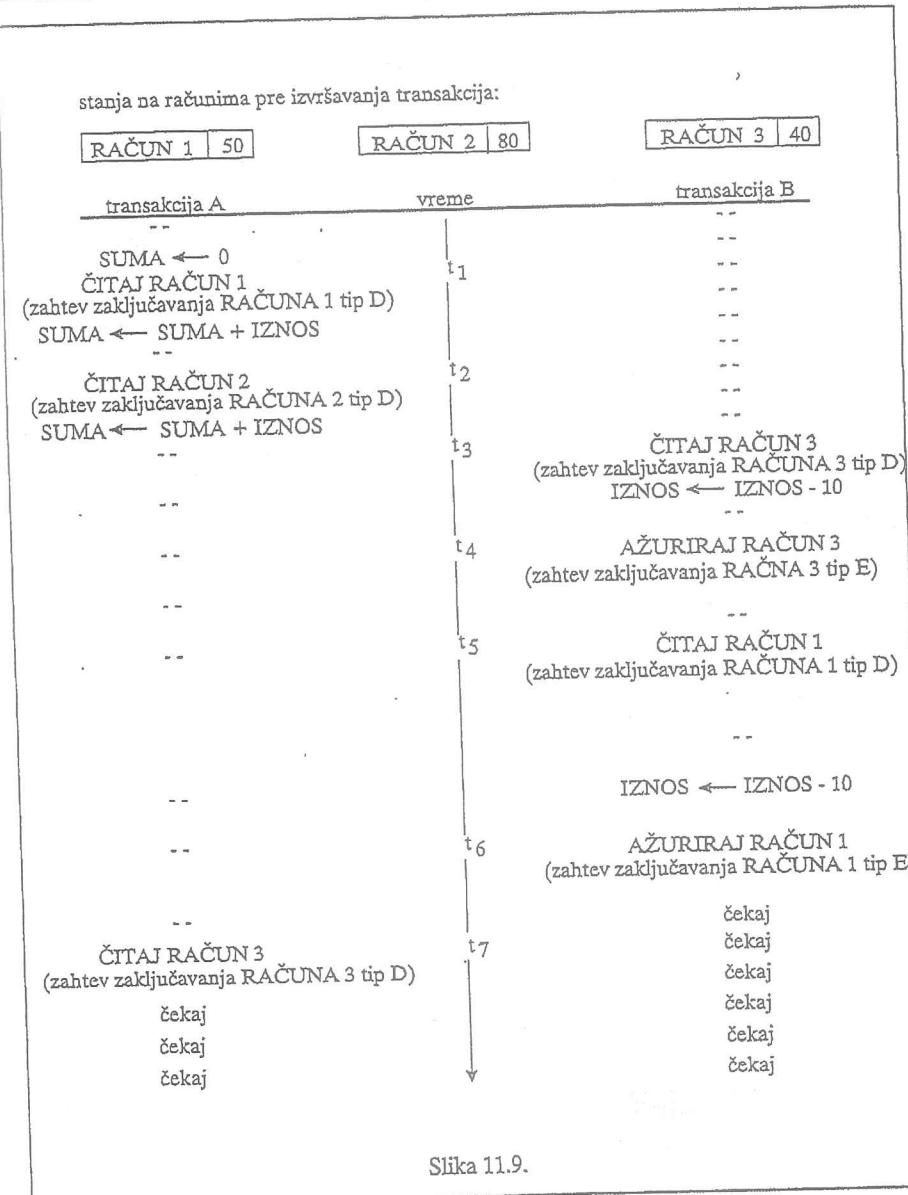
Zbog toga transakcija A prelazi u stanje čekanja sve dok transakcija B ne dostigne tačku sinhronizacije izvršavajući operaciju COMMIT ili ROLLBACK (trenutak t_3). U trenutku t_3 transakcija B deblokira zapis R i transakcija A može nastaviti izvršavanje. Transakcija A u trenutku t_3 raspolaže zapisom R pre izvršavanja transakcije B ako je B završila sa ROLLBACK ili zapisom R koji je rezultat uspešnog završavanja transakcije B. U oba slučaja transakcija A više ne zavisi od privremenih ažuriranja druge transakcije.



Slika 11.8.

Problem narušavanja serijabilnosti

Na slici 11.9. data je modifikovana varijanta slike 11.4. uzimanjem u razmatranje mehanizma zaključavanja pri paralelnom izvršavanju transakcija. Možemo uočiti da zahtev transakcije B u trenutku t_6 za operaciju AŽURIRAJ neće biti izvršen, jer se ne može izvršiti zaključavanje tipa E zapisa RAČUN 1 jer je taj zapis zaključan po tipu D od strane transakcije A. Od trenutka t_6 transakcija B prelazi u stanje čekanja. U trenutku t_7 ne izvršava se zahtev ČITAJ RAČUN 3 transakcije A jer se ne može izvršiti zaključavanje tog zapisa po tipu D zbog toga što je RAČUN 3 zaključan od transakcije B po tipu E. Transakcija A u trenutku t_7 prelazi u stanje čekanja. Rešavajući problem narušavanja serijabilnosti došli smo do medjusobnog blokiranja transakcija.



Slika 11.9

11.3. Direktno zaključavanje

Prethodno su razmatrane mogućnosti načina automatskog zaključavanja zapisa odnosno n-torki prilikom izvršavanja određenih operacija. SUBP poseduju i načine direktnog zaključavanja podataka mada ih u većini situacija nije neophodno koristiti. S obzirom da načini direktnog zaključavanja zavise od konkretnog SUBP ovde ćemo dalje razmotriti samo direktno zaključavanje korišćenjem naredbe **LOCK TABLE** jezika SQL.

Opšti oblik naredbe SQL za zaključavanje svih n-torki tabele glasi:

**LOCK TABLE naziv_tabele [, naziv_tabele]...
IN {SHARE | SHARE UPDATE | EXCLUSIVE } MODE [NOWAIT].**

gde je naziv tabele ime bazne tabele i ne može biti ime pogleda.

Tabelu zaključavamo na jedan od sledećih načina:

SHARE - kada tabelu koristimo za pretraživanje i kada drugim korisnicima (transakcijama) dozvoljavamo korišćenje iste tabele za pretraživanje, ali ne i za ažuriranje.

SHARE UPDATE - kada možemo ažurirati tabelu i kada drugi korisnik može zaključati n-torku korišćenjem naredbe `SELECT ... FOR UPDATE`. Dozvoljeno je ažuriranje n-torke koja je zaključana od drugog korisnika pod uslovom da još nije ažurirana od drugog korisnika.

EXCLUSIVE - kada ažuriramo više n-torki tabele i sprečavamo druge korisnike da zaključaju ili ažuriraju bilo koju n-torku tabelu:

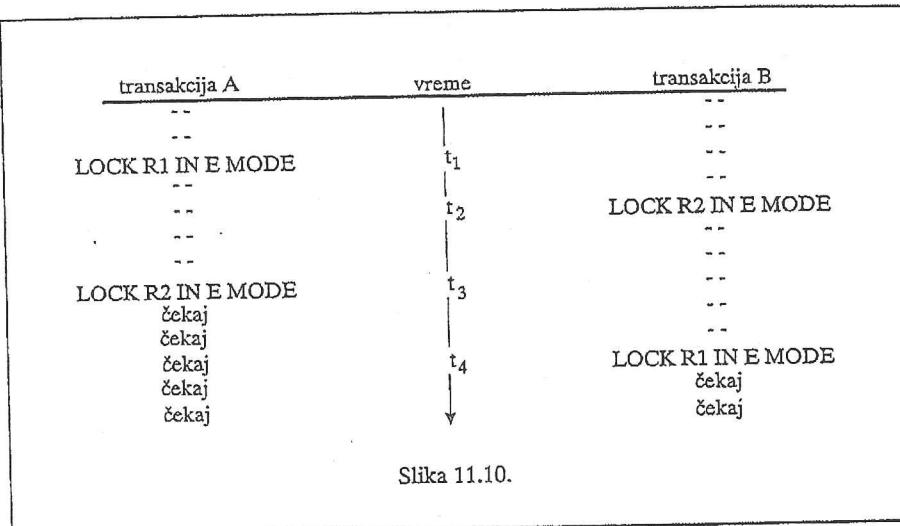
Ukoliko zahtev LOCK TABLE ne može biti izvršen zato što je na primer, drugi korisnik zaključao tabelu, LOCK TABLE prelazi u stanje čekanja do trenutka kada će moći biti izvršen. Ukoliko se navede opcija NOWAIT i zahtev LOCK TABLE ne može biti izvršen kontrola se vraća programu tako da dok čekate možete zahtevati izvršavanje nekih drugih naredbi.

Kada je jednom zaključana tabela ostaje zaključana sve do tačke sinhronizacije (COMMIT ili ROLLBACK).

11.4. Medjusobno blokiranje transakcija

Prethodno smo razmatrali kako se medjusobno blokiranje transakcija može koristiti za rešavanje nekih problema paralelnog izvršavanja transakcija i izbegavanja nepoželjnog mešanja podataka. Međutim videli smo na žalost da medjusobno blokiranje transakcija dovodi do blokiranog stanja.

I pored toga što smo ranije ilustrovali medjusobno blokiranje transakcija, razmotrimo stanje na slici 11.10. koje ilustruje opštiji slučaj razmatranog problema. Primetimo da operacije zaključavanja mogu biti bilo koje operacije koje iniciraju zaključavanja.



Slika 11.10.

Blokirano stanje je takva situacija u kojoj se dve ili više transakcija istovremeno nalaze u stanju čekanja i svaka od njih čeka da ona duga transakcija skine blokiranje kako bi se omogućio njen dalji nastavak. Na slici 11.10. ilustrovano je medjusobno blokiranje dve transakcije A i B. Moguće je takođe medjusobno blokiranje tri, četiri i više transakcija. U praksi blokirana stanja skoro nikada nisu uzrokovana od strane više od dve transakcije.

Kada nastane blokirano stanje potrebno je isto prepoznati i ukloniti. Da bi se rešilo blokirano stanje jedna od transakcija koja učestvuje u blokiranju odabira se kao *žrtva* koja će biti našilno prekinuta uz oslobođanje svih

zaključavanja koje je ona postavila i poništavanjem svih efekata na bazi podataka koje je ona izvršila. Na taj način dozvoljava se drugoj blokiranoj transakciji nastavak izvršavanja dok se poništена transakcija aktivira od početka.

Ukažimo bez detaljnih razmatranja da je jedan od načina identifikovanja blokiranih stanja analiza koda koji pokazuje ishod izvršavanja date SQL naredbe, a koji se dobija posle izvršavanja svake naredbe.

Za smanjenje verovatnoće pojave blokiranih stanja kao preporuka može se dati da sve transakcije zaključavaju po mogućnosti n-torce u istom redosledu. Primljeno na situaciju ilustrovano na slici 11.10. to bi zahtevalo na primer, izmenu redosleda zaključavanja u transakciji B tako da se prvo zahteva zaključavanje R1, a posle toga zaključavanje R2.

11.5. Restauracija konzistentnog stanja baze podataka

Prethodno su razmatrani pojednostavljeni modeli transakcija bez nekih praktičnih aspekata koje ćemo sada nešto detaljnije analizirati. Realne transakcije se mogu opisati na sledeći način:

Transakcija je niz pojedinačnih radnji sa osobinama:

a) Pojedinačne radnje

- (i) Radnja se ili uspešno obavi i tada ostavlja odgovarajući efekat na bazu podataka ili nema nikakvog efekta na bazu podataka.
- (ii) Radnje su nedeljive, što znači ako se dve radnje obavljaju nad istim objektom, ukupan efekat odgovara situaciji u kojoj se prvo jedna radnja izvrši u celosti, pa zatim druga.

b) Kompozicija radnji

- (i) Radnje jedne transakcije se izvršavaju u redosledu koji je određen navodnjem tih radnji u transakciji.
- (ii) Transakcija se ili u celini uspešno izvrši i tada ostavlja na bazu podataka efekat sekvenčnog izvršenja svojih radnji ili ne ostavlja nikakav efekat na bazu podataka.

- (iii) Ako se transakcija uspešno završi, njeni efekti na bazu podataka su trajni. Jedini način da se ponište efekti uspešno završene transakcije je nova uspešna transakcija čiji efekti poništavaju efekte prve transakcije.
- (iv) Pre završetka transakcija može samu sebe prekinuti ili može biti prekinuta od SUBP. U oba slučaja transakcija ne ostavlja nikakvog efekta na bazu podataka.
- (v) U okviru transakcije je moguće definisati tačku sinhronizacije tj. pamćenja stanja izvršenja transakcije. Pre uspešnog završetka transakcije moguće je zahtevati poništavanje svih njenih efekata na bazu podataka do neke prethodno zapamćene tačke sinhronizacije.

Sledi nekoliko modela obrade transakcija koji imaju cilj da zadovolje prethodno navedene uslove:

Privatni memorijski prostor

Efekat pojedinih akcija od kojih je transakcija T sastavljena može se opisati na sledeći način:

1. Počni transakciju T.

Rezerviše se privatni memorijski prostor transakcije T čija je funkcija da prihvata vrednosti koje T čita iz baze podataka ili ih upisuje u bazu podataka.

2. Pročitaj vrednost objekta X.

Ispituje se da li privatni memorijski prostor transakcije T sadrži objekat X. Ako sadrži, tada se vrednost tog objekta X stavlja na raspolaganje transakciji T. U protivnom se iz baze podataka pročita vrednost objekta X. Ta se vrednost stavlja na raspolaganje transakciji T i istovremeno upisuje u privatni memorijski prostor transakcije T.

3. Upiši novu vrednost objekta X.

Pretražuje se privatni memorijski prostor transakcije T kao i kod čitanja. Ako se u njemu nalazi objekat X, ažurira se vrednost objekta X. U suprotnom se u privatnom memorijskom prostoru transakcije T kreira primerak objekta X sa datom vrednošću. Možemo uočiti da efekat ove radnje nije menjanje vrednosti objekta X u bazi podataka.

4. Kraj transakcije T.

Vrednosti svih objekata iz privatnog memorijskog prostora transakcije T se upisuju u bazu podataka. Posebnim *protokolom nedeljivog kompletiranja transakcije* se garantuje da se ili sva ažuriranja iz privatnog memorijskog prostora prenesu u bazu podataka ili nijedno. Oslobodi se privatni memorijski prostor transakcije T.

Protokol nedeljivog kompletiranja transakcije obezbeđuje da se obrade sva ažuriranja jedne transakcije ili nijedno. Sastoјi se iz sledeće dve faze:

Prva faza - za svaku radnju ažuriranja objekta X obavlja se radnja prethodnog upisivanja nove vrednosti objekta X u memoriju za koju se pretpostavlja da je pouzdana.

Druga faza - za svaku radnju ažuriranja objekta X obavlja se upisivanje nove vrednosti objekta X u bazu podataka.

Definisani protokol ima sledeće osobine:

- Ako dodje do pada sistema u toku prve faze, to neće izazvati nikakve probleme, jer se u toku te faze nikakve promene ne vrše na bazu podataka.
- Ako dodje do pada sistema u toku druge faze, baza podataka može biti u nekonistentnom stanju, jer su u njoj parcijalno zapisana ažuriranja jedne transakcije. U postupku uspostavljanja konzistentnog stanja baze podataka ovaj se problem rešava na osnovu podataka o izvršenim ažuriranjima koji su upisani u pouzdanu memoriju.

Modelom privatnog memorijskog prostora realizuju se sve osobine transakcije osim osobine (v). Poništavanje svih efekata transakcije (iv) postiže se oslobadjanjem privatnog memorijskog prostora.

Dnevnik ažuriranja

Po modelu privatnog memorijskog prostora zahteva se upisivanje nove vrednosti objekta u sigurnu memoriju koju možemo nazvati i *log transakcije*. Ako se u log transakcije, umesto nove vrednosti, upisuje stara vrednost objekta koji se ažurira, tada privatni memorijski prostor transakcije nije

neophodan. Svaka radnja ažuriranja vrednosti objekta prvo upisuje prethodnu vrednost objekta (stara vrednost) u log transakcije, a zatim direktno u bazu podataka upisuje novu vrednost. Suprotno modelu privatnog memorijskog prostora, ne odlaze se ažuriranje baze podataka do naredbe za kompletiranje transakcije. Poništavanje efekata neke transakcije pre njenog uspešnog kompletiranja postiže se čitanjem unatrag loga transakcije i restauriranjem prethodnih vrednosti objekta u bazi podataka. Za poništavanje efekata transakcije pored prethodnih vrednosti objekta za svaku radnju iz skupa radnji pomoću kojih se transakcija realizuje treba definisati i njoj odgovarajuću radnju poništavanja. Ovom metodom može se realizovati i osobina b.(v) tako što se svaka tačka sinhronizacije unosi u log transakcije čime se obezbeđuje mogućnost poništavanja ažuriranja transakcije do nekog prethodno zapamćenog stanja izvršenja.

Uspešan završetak transakcije se registruje upisivanjem tačke sinhronizacije u log transakcije. Ako dodje do pada sistema pre tog trenutka, svi efekti transakcije će biti automatski poništeni na osnovu prethodnih vrednosti i drugih relevantnih podataka iz loga transakcije prilikom ponovnog starta rada sa bazom podataka. Logovi pojedinačnih transakcija se kombinuju u jedinstven sistemski log koji nazivamo *dnevnik ažuriranja*, u kojem su pokazivačima povezani zapisi koji predstavljaju log jedne transakcije.

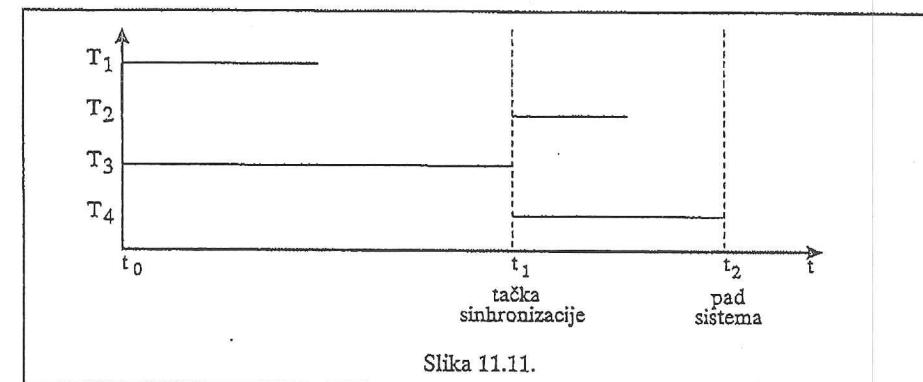
Protokol restauracije konzistentnog stanja baze podataka na osnovu dnevnika ažuriranja zavisi od načina postavljanja tačke sinhronizacije.

Najjednostavniji, ali istovremeno i najmanje prihvatljiv za korisnike, način postavljanja tačke sinhronizacije je odlaganje svih novih transakcija dok se transakcije koje su u toku ne završe. U trenutku kada više ni jedna transakcija nije u toku, u dnevnik ažuriranja se upisuje zapis tačke sinhronizacije i mogu da započnu nove transakcije. Restauriranje konzistentnog stanja baze podataka obavlja se prema sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku t_1 i dnevnik svih promena do trenutka t_2 . Ako dodje do pada sistema u trenutku t_2 , poništavaju se sve promene transakcija koje su počele posle poslednje tačke sinhronizacije tj. t_1 . Zatim se ponavljaju radnje svih transakcija koje su počele posle poslednje tačke sinhronizacije, a uspešno su završene pre pada sistema.

Na slici 11.11. ilustrovano je jedno stanje odvijanja transakcija. Trenutak t_0 predstavlja tačku sinhronizacije i tada su započele transakcije T_1 i T_3 . Dok se

ove dve transakcije ne završe ni jedna druga transakcija ne može započeti. U trenutku t_1 transakcija T_1 i T_3 su završene, u dnevnik ažuriranja se upisuje tačka sinhronizacije i aktiviraju se transakcije T_2 i T_4 . U trenutku t_2 nastaje pad sistema. Konzistentno stanje uspostavlja se poništavanjem svih efekata transakcija T_2 i T_4 i ponavljanjem svih radnji transakcije T_2 .



Slika 11.11.

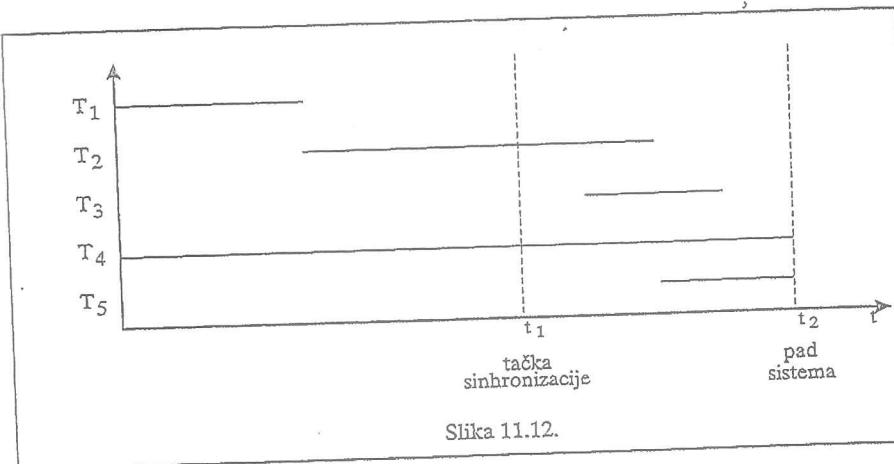
Nedostatak opisanog načina postavljanja tačaka sinhronizacije i uspostavljanja konzistentnog stanja su dugi periodi čekanja na završetak transakcija koje su u toku da bi startovali nove transakcije.

Raspoloživost sistema se može znatno poboljšati ako se tačka sinhronizacije može upisati u dnevnik ažuriranja u trenutku kada se ne obavlja nijedna radnja transakcija koje su u toku. U tački sinhronizacije uspostavlja se korespondencija između stanja baze podataka i dnevnika ažuriranja, tj. sve promene upisane u dnevnik ažuriranja od prethodne tačke sinhronizacije unose se u bazu podataka. Zatim se u dnevnik ažuriranja upisuje zapis tačke sinhronizacije koji sadrži spisak svih transakcija koje su u toku i za svaku od njih pokazivač na njen poslednji zapis u dnevniku ažuriranja.

Uspostavljanje konzistentnog stanja baze podataka nakon pada sistema se obavlja po sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku t_1 i dnevnik promena do trenutka t_2 . Ako dodje do pada sistema u trenutku t_2 , poništavaju se sve promene transakcija koje su imale neko dejstvo na bazu podataka od trenutka t_1 do trenutka t_2 . Zatim se ponavljaju sve radnje od trenutka t_1 , onih transakcija koje su se uspešno završile u periodu od trenutka t_1 do trenutka t_2 .

Na slici 11.12. ilustrovano je jedno stanje odvijanja transakcija.



Slika 11.12.

Efekti transakcija T_2 , T_3 , T_4 i T_5 koje su bile u toku u trenutku pada sistema u trenutku t_2 , pri restauraciji konzistentnog stanja će biti poništeni. Sve radnje transakcije T_2 će biti ponovljene od tačke sinhronizacije t_1 . Transakcija T_3 će transakcije T_2 biti ponovljene od tekuće verzije baze podataka. Nekoliko radnjama transakcije T_4 uvek utiču samo na tekuću verziju i ne menjaju radnjama transakcije T_2 . Poništavaju se sva ažuriranja koja je obavila transakcija T_4 .

Dalje poboljšanje poslednjeg modela uspostavljanja konzistentnog stanja može se ostvariti uvodjenjem dve verzije baze podataka koje se mogu koristiti istovremeno. Jedna verzija se zove *tekuća*, a druga *sigurnosna*. Svojim radnjama transakcije uvek utiču samo na tekuću verziju i ne menjaju radnjama transakcije T_2 . Postoje sledeće dve karakteristične operacije sa tekućom i sigurnosnom verzijom baze podataka.

(1) Tekuća verzija postaje sigurnosna. Time se ažuriranja baze podataka koja su obavljena od poslednjeg izvršenja ove operacije i faktički upisana u bazu podataka.

(2) Sigurnosna verzija postaje tekuća. Time se poništavaju sva ažuriranja baze podataka koja su se odigrala od trenutka poslednjeg izvršenja operacije (1).

Operacija (1) se obavlja u vreme zapisivanja tačke sinhronizacije, a operacija (2) u postupku restauracije konzistentnog stanja baze podataka. Nakon operacije (2) se koristeći dnevnik ažuriranja, iz baze uklone efekti transakcija

koje se nisu uspešno završile, a restauriraju efekti uspešno završenih transakcija.

Uspostavljanje konzistentnog stanja baze podataka nakon pada sistema se obavlja po sledećem protokolu:

Neka je data tačka sinhronizacije koja odgovara vremenskom trenutku t_1 . Pretpostavljamo da je pre njenog registrovanja izvršena operacija zamene sigurnosne verzije baze podataka njenom trenutnom verzijom. Neka je dat dnevnik svih ažuriranja do trenutka t_2 kada nastaje pad sistema.

Obavlja se operacija zamene tekuće verzije baze podataka sigurnosnom verzijom tj. tekućom verzijom baze podataka postaje sigurnosna verzija.

Poništavaju se sve promene nastale do trenutka t_1 , onih transakcija koje nisu uspešno završene do trenutka t_2 .

Ponavljaju se sve radnje izvršene od trenutka t_1 do trenutka t_2 transakcija koje su se uspešno završile do trenutka t_2 .

Na slici 11.12. ilustrovano je jedno stanje odvijanja transakcija koje smo već razmatrali. Prema modelu tekuće i sigurnosne verzije baze podataka, konzistentno stanje na datom primeru uspostavlja se na sledeći način:

Operacijom zamene tekuće verzije sigurnosnom, sva ažuriranja transakcija T_2 , T_3 , T_4 i T_5 posle tačke sinhronizacije t_1 poništena su. Zatim se poništavaju sva ažuriranja koja je obavila transakcija T_4 . Ponavljaju se sve radnje transakcija T_2 i T_3 koje su uspešno završene pre pada sistema.