Relacione Baze Podataka

Mina Milošević

2019/2020

Sadržaj

1 Arhitektura

1.1 ANSI/SPARC arhitektura

Arhitektura sistema baze podataka je apstraktni opis njegovih komponenti i njihovih interakcija.
ANSI - American National Standards Institute
SPARC - System Planning and Requirements Committee
Prema ANSI/SPARC arhitekturi, baze podataka sadrze tri nivoa:

1. Spoljašnji nivo - definiše način na koji individualni korisnik vidi podatke iz baze. Ovaj nivo je najbliži korisniku, jer korisnika zanima samo jedan deo cele baze gde on vidi samo spoljašnje slogove, ali ne i njihovu fizičku reprezentaciju u bazi. Svaki spoljašnji izgled je opisan spoljašnjom shemom koja se sastoji iz definicija svakog od različitih tipova slogova. Svaki korisnik ima na raspolaganju matični jezik (Java, C, PL1, Cobol...) u koji se ugradjuje jezik podataka (SQL, DB2, QUEL,...) pomoću koga može da vrši operacije nad svojim delom podataka iz baze. Ovi jezici mogu biti čvrsto vezani, kada matični jezik ne može da se odvoji od jezika podataka, ili labavo vezani kada oni mogu lako i jasno da se razdvoje.
Jezik podataka je kombinacija jezika za definiciju podataka (DDL) koji se koristi za deklarisanje ili definisanje objekata u bazi, i jezika za rad sa podacima (DML) koji se koristi pri radu i obradi objekata iz baze.

2. Konceptualni nivo - predstavlja ukupni informacioni kontekst baze podataka u obliku koji je na nešto višem nivou u poredjenju sa načinom kako su podaci fizički smešteni. Podaci se predstavljaju nezavisno i od upitnog jezika i od hardvera na kome se nalaze. Konceptualni izgled je definisan konceptualnom shemom koja sadrži definicije svakog od tipova konceptualnih slogova i zapisuje se pomoću konceptualnog DDL-a, bez ikakve veze sa fizičkom reprezentacijom tih slogova ili pristupa njima. Definicije u konceptualnoj shemi mogu sadržati i dodatne funkcionalnosti vezane za bezbednost i integritet podataka.

3. Unutrašnji nivo - predstavlja celokupnu bazu podataka na niskom nivou. Sastoji se od velikog broja različitih unutrašnjih slogova. Unutrašnji izgled je definisan preko unutrašnje sheme napisane na unutrašnjem DDL-u koja sadrži ne samo definicije različitih slogova vec sadrži i informacije o postojanju indeksa, reprezentaciji sačuvanih polja, kako su fizički smešteni sačuvani slogovi, itd. Neki programi mogu da rade nad unutrašnjim izgledom baze što donosi bezbednosni rizik.
Osim ova 3 nivoa, arhitektura uključuje odredjene vrste preslikavanja:

- *konceptualno/unutrašnje* - preslikavanje izmedju konceptualnog nivoa i baze tj. kako su konceptualna polja i slogovi predstavljeni na unutrašnjem nivou. Ako se promeni definicija sačuvane baze mora se promeniti i konceptualno/unutrašnje preslikavanje. Ključno je za nezavisnost podataka od promene fizičke strukture.
- *spoljašnje/konceptualno* - definiše vezu izmedju spoljašnjeg i konceptualnog nivoa. Ključno je za nezavisnost podataka od promene logičke strukture.
- *spoljašnje/spoljašnje* - definiše jedan spoljašnji pogled preko ostalih. Često je u relacionim sistemima.

1.2 Glavne komponente SUBP-a

1. ***Podaci*** - mogu biti *integrisani* i *deljivi*. Kod deljivih podataka različiti korisnici pristupaju istim podacima često i u isto vreme. Kod integrisanih podataka bazu čini skup inače nepovazanih fajlova koji medjusobno gotovo da nemaju viškova (suvišnih podataka ili onih koji se ponavljaju).
2. ***Hardver*** - spoljašnji memorijski uredjaji i procesori i glavna memorija.

3. ***Softver*** - *SUBP* (Sistem za upravljanje bazom podataka) i on predstavlja nivo softvera koji se nalazi izmedju korisnika i fizičkih podataka u bazi, štiti korisnike od detalja na hardverskom nivou i upravlja svim zahtevima za direktan pristup bazi. Alati za razvoj aplikacija, pisanje izvestaja, pomoćni (utility) programi, program za upravljanje transakcijama (TP monitor).

4. ***Korisnici*** - *aplikativni programeri* pišu programe na višim programskim jezicima (Cobol, PL1, C++, Java,...) koji služe za pristup bazi. *Krajnji korinsnici* interaktivno prisupaju bazi pomoću programa koji pišu aplikativni progameri. *Administrator baze podataka* (DBA) - profesionalac u IT, formira i implementira kontrolne strukture, odgovoran je za implementaciju odluke administratora podataka kao i za rad sistema, performanse, itd. Njegovi poslovi su definisanje konceptualne sheme (logičko projektovanje baze), definisanje unutrašnje sheme (fizičko projektovanje baze) i komunikacija sa korisnicima (da li su im obezbedjeni svi željeni podaci, konsultacija pri projektovanju aplikacija, pomoć pri rešavanju problema, itd.). *Administrator podataka* (DA) - on razume postojeće podatke i odlučuje koji podaci će biti čuvani u bazi. On takodje ustanovljava pravila za održavanje i rad sa podacima po njihovom čuvanju u bazi; nije tehničko lice već pripada upravljačkim strukturama.

1.3 Najvažnije funkcije SUBP-a

• definisanje podataka - SUBP treba da primi podatke u izvornom formatu i pretvori ih u objekte tako da on zapravo mora da ima DDL procesor ili kompajler da razume DDL definicije.

• obrada podataka - SUBP treba da rešava zahteve dohvatanja, menjanja, dodavanja ili brisanja podataka, zapravo mora da ima DML kompajler ili procesor. DML zahtevi mogu biti planirani (zahtev je poznat unapred) i neplanirani (zahtev nije poznat unapred).

• optimizacija izvršavanja upita - DML zahtevi bilo da su planirani ili ne, moraju proći kroz optimizaciju i potom tako optimizovani se izvršavaju pod kontrolom runtime menadžera.

• obezbedjivanje zaštite i integriteta podataka - SUBP mora da kontroliše zahteve korisnika i odbije svaki zahtev koji bi narušio integritet i bezbednost baze.

• obezbedjivanje konkuretnog pristupa podacima i oporavka podataka - pomoću TP monitora.

• formiranje kataloga podataka - informacije o definiciji svih objekata

• obezbedjivanje korinsničkog interfejsa

• izvodjenje drugih akcija u svrhu obezbedjivanja što efikasnijeg rada

1.4 Nezavisnost podataka u bazi podataka

U bazama podataka, ***nezavisnost podataka*** predstavlja otpornost aplikacije na promene fizičke reprezentacije podataka i pristupnih tehnika. Glavne karakteristike su da baza podataka treba da bude sposobna da se širi bez promene postojećih aplikacija kao i da u slučaju širenja baze ne sme da bude negativnih uticaja na postojeće aplikacije. Pojmovi vezani za nezavisnost podataka:

- sačuvano polje - najmanja jedinica podataka koja moze da se čuva
- sačuvani slog - skup sačuvanih polja
- sačuvana datoteka - skup svih trenutno postojećih pojava sačuvanih slogova istog tipa

Aspekti sačuvanih reprezentacija koji mogu da budu predmet promena od strane DBA:

- reprezentacija brojčanih podataka
- reprezentacija znakovnih podataka
- jednice za brojčane podatke
- kodiranje podataka

1.5 Prednosti rada sa bazom podataka u odnosu na rad sa podacima koji se nalaze u datotekama

• ***Podaci mogu biti deljeni*** - deljeni znači da ne samo da aplikacije mogu da dele podatke u okviru baze, već i da mogu biti pisane nove aplikacije koje će raditi sa istim podacima.

• ***Smanjenje redundantnosti podataka*** - u sistemima koji nisu baze podataka svaka aplikacija ima svoje privatne fajlove što može dovesti do mnogo ponavljanja medju podacima i bespotrebnog trošenja memorije.

• ***Izbegavanje nekonzistentnosti*** - povezano sa prethodnim

• ***Podrška za transakcioni rad*** - ako korisnik traži da se izvrše dve transakcije odjednom, sistem može da garantuje da su se ili obe izvršile ili nijedna.

• ***Održavanje integriteta*** - ako imamo redundantnost podataka, može da se desi da prilikom reažuiranja ažuriramo jedan podatak, a ne drugi, čime baza neće vraćati validne podatke kada joj se pristupi. Rešenje je da kada reažuriramo jedan, automatski će biti reažuirana i sva ostala ponavljanja i SUBP garantuje da korisnik neće videti redundantnost podataka.

• ***Bezbednost*** - konfliktnim zahtevima i standardima se bavi DBA koji bira na koji način će rešiti ove slučajeve u cilju što optimalnijeg sistema.

1.6 Prednosti relacionog modela u odnosu na hijerarhijski i mrežni

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

1.7 Utility programi. Programi koji rade sa unutrašnjim izgledom baze podataka.

Utility programi - programi koji služe da pomognu DBA sa različitim administrativnim poslovima. Oni mogu da rade na spoljašnjem nivou i to su aplikacije specijalne namene, i unutrašnjem nivou i deo su servera. U praksi su nam potrebni ovakvi programi najčešće za kreiranje inicijalne verzije baze od regularnih podataka, za reorganizovanje podataka u bazi, za statističke rutine (računaju statistiku performansi baze: velicina fajlova, I/O brojači,...) i analizu statističkih podataka.
Utility programi: load, copy, import, reorg, recover, runstats, check,...

2 Uvod u relacione baze podataka

2.1 Aspekti relacionog modela podataka

Intuitivno, ***relacioni model*** predstavlja jedan način gledanja na podatke (preko tabela) i sadrži pravila za rad sa tim podacima (izdvajanje, spajanje,...).

1) *Aspekt strukture* - svi podaci u bazi se korisniku prikazuju isključivo u obliku tabela

2) *Aspekt integriteta* - tabele zadovoljavaju izvesna ograničenja (primarni i spoljasnji ključevi,...)

3) *Aspekt obrade* - operatori koji su na raspolaganju korisnicima za obradu tabela su takvi da izvode tabele iz tabele.

2.2 Karakteristike relacione baze podataka

Relacioni sistemi zahtevaju samo da se baza prikaže korisniku u obliku tabele. Način smeštanja i čuvanja na medijumu nije specifičan. Informacioni pristup: celokupan informacioni kontekst baze se prikazuje na tačno jedan način kao eksplicitne vrednosti u pozicijama vrsta i kolona tabele. Posledica: nema pokazivača koji medjusobno povezuju tabele (mogu da postoje na fizičkom nivou). Rezultat primene svakog operatora je tabela. Rezultat primene operatora je istog tipa kao i njegov argument. Sve operacije se primenjuju na ceo skup istovremeno.

Relacioni model se sastoji od:

1. otvorenog skupa skalarnih tipova
2. generatora relacionih tipova i njihove odgovarajuće interpretacije
3. mogućnost definisanja relacionih promenljivih za generisane relacione tipove
4. operacije relacione dodele kojom se dodeljuju relacione vrednosti definisanim relacionim promenljivim
5. otvorenog skupa opštih relacionih operatora za izvodjenje relacionih vrednosti iz drugih relacionih vrednosti (relaciona algebra i račun)

2.3 Terminologija

Codd je pri formulisanju principa relacionih baza uveo novu terminologiju:

- relacija - matematički termin za tabelu
- torka - red u tabeli
- atribut - kolona u tabeli
- kardinalnost- broj torki
- stepen - broj atributa
- domen - skup važećih vrednosti/tipova
- primarni ključ - atribut ili kombinacija atributa koja jedinstveno identifikuje tabelu

2.4 Osobine transakcije

Transakcija je logička jedinica posla koja obično uključuje više operacija nad bazom.

1. *Atomičnost* - garantuje se da će se izvršiti ili sve što se nalazi u transakciji ili ništa od toga.

2. *Trajnost* - garantuje se da će po uspešnom izvršavanju (COMMIT-a) sve promene ostati trajno zapamćene u bazi, bez obzira na kasnije eventualne padove sistema.

3. *Izolovanost* - transakcije su medjusobno izolovane u smislu da su efekti izvršavanja jedne transakcije nevidljivi za drugu transakciju sve do izvršavanja COMMIT naredbe.

4. Izvršavanje isprepletanog (u smislu početka i kraja) skupa transakcija će obično biti *serijalizovano* u smislu da se dobija isti rezultat kao da se te iste transakcije izvršavaju jedna po jedna u unapred neodredjenom redosledu izvršavanja.

2.5 Relacija i njene osobine

Neka je dat skup od n tipova ili domena, $T_i, i = 1, n$, pri čemu ne moraju svi tipovi da budu medjusobno različiti. R je ***relacija*** nad tim tipovima ako se sastoji od dva dela, zaglavlja i tela. *Zaglavlje* je skup od n atributa oblika $A_i = T_i$, gde su A_i imena atributa relacije R. *Telo* je skup od m torki t, gde je t skup komponenti oblika A_i:V_i, u kojima je V_i vrednost tipa T_i.
Osobine:
- nema ponovljenih torki
- torke su neuredjene
- atributi su neuredjeni
- svaka torka sadrži tačno jednu vrednot za svaki atribut

Relacija i tabela nisu jednake jer tabela može da sadrži duplikate, redovi u tabeli su uredjeni od vrha ka dnu i kolone u tabeli su uredjene s leva u desno.
Relacija koja ima prazan skup torki - neprazno zaglavlje i prazno telo. Relacija koja ima praznu torku - prazno zaglavlje i telo sa jednom torkom bez komponenti

3 Relaciona algebra i relacioni račun

3.1 Relacioni operatori

Codd je originalno predložio 8 operatora: restrikcija (selekcija), projekcija, proizvod, unija, presek, razlika, (prirodno) spajanje, deljenje.
Kasnije su dodati operatori: promena imena, poluspajanje, polurazlika, ekskluzivna unija, proširenje, slika relacije, operatori agregata, sumarizacija...
Minimalni skup operatora cine: restrikcija (selekcija), projekcija, prozivod, unija, razlika.

3.2 Relaciono zatvorenje. Relaciona kompletnost

Osobina da su argumenti i rezultati primene bilo kog relacionog operatora takodje relacije se naziva ***relaciono zatvorenje***. Posledica relacionog zatvorenja je mogućnost pisanja ugnježdenih relacionih izraza tj. relacionih izraza čiji su operandi takodje relacioni izrazi. Treba obezbediti da i novodobijene relacije imaju odgovarajuće zaglavlje i telo bez obzira da li su u pitanju osnovne ili izvedene relacije.
Jezik je ***relaciono kompletan*** ako je moćan isto kao i algebra, tj. ako bilo koja relacija predstavljiva u algebri može da se predstavi i u tom jeziku.
SQL je relaciono kompletan, jer postoje SQL izrazi za svaki od 5 primitivnih operatora relacione algebre.

3.3 SQL ekvivalenti osnovnih operatore relacione algebre

Algebra	*SQL*
A WHERE P	SELECT * FROM A WHERE P
A $\{x, y, ..., z\}$	SELECT DISTINCT x, y, ..., z FROM A
A TIMES B	A CROSS JOIN B
A UNION B	SELECT * FROM A UNION SELECT * FROM B
A MINUS B	SELECT * FROM A EXCEPT SELECT * FROM B
A RENAME x AS y	SELECT x AS y FROM A

3.4 Kodov algoritam redukcije

Kodov algoritam redukcije je prikaz da je relaciona algebra moćna koliko i relacioni račun i obratno.

3.5 Relacioni operatori poluspajanja i polurazlike pomoću osnovnih Kodovih operatora

Operator poluspajanja možemo izraziti preko projekcije i prirodnog spajanja:

$$r1\ MATCHING\ r2\ =\ (r1\ JOIN\ r2)\{H1\}$$

gde je $\{H1\}$ zaglavlje relacije r1.
Iz poluspajanja i razlike mozemo izvesti polurazliku kao:

$$r1\ NOT\ MATCHING\ r2\ =\ r1\ MINUS\ (r1\ MATCHING\ r2)$$

4 SQL

4.1 Domen u SQL-u

Domen je u SQL-u prost, korisnički definisan imenovan objekat koji se može koristiti kao alternativa za predefinisan tip podatka nad kojim se definiše. Može imati default vrednost i jedno ili više ograničenja. Može biti ugradjen (Integer, Char...) ili korisnički definisan (Ime, Indeks...). SQL podržava 8 relacionih domena: brojevi, niske karaktera, niske bitova, datumi, vremena, kombinacija datuma i vremena, intervali godina/mesec, intervali dan/vreme.

4.2 Primarni i strani ključevi

Kandidat za ključ relacije predstavlja podskup atributa X te relacije, ako vazi:
- *pravilo jedinstvenosti*: ne postoje dve torke u relaciji R koje imaju iste vrednosti za X
- *pravilo minimalnosti*: ne postoje pravi podksup skupa X koji zadovoljava pravilo jedinstvenosti.

Vrste kljuceva su:
- *primarni ključ*: jedan od kandidata za ključ
- *alternativni ključevi*: ostali kandidati
- *spoljašnji (strani) ključ*: skup atributa jednog relvar-a R2 čije vrednosti treba da odgovaraju vrednostima nekog kandidata za ključ nekog relvar-a R1
- *superključ*: nadskup kandidata za ključ; poseduje jedinstvenost, ali ne i minimalnost.

4.3 Ograničenja integriteta

Postoje ograničenja stanja i ograničenja prelaza.
Ograničenja stanja definišu prihvatljiva stanja u bazi i ona se dele na *ograničenja baze* koja se odnose na vrednosti koje je dozvoljeno čuvati u bazi (tj. koje se odnose na dve ili više različitih relacija), *ograničenja relacija* (relvar-a) kojima se zadaje ograničenje na vrednost pojedinačne relacije (relvar-a) koje se proverava pri ažuriranju te relacije, *ograničenja atributa* koja predstavljaju ograničenja na skup dozvoljenih vrednosti datog atributa i *ograničenja tipa* koja predstavljaju definiciju skupova vrednosti koji čine dati tip.

Primeri: a) Ograničenje baze

```
CONSTRAINT BAZA1
FORALL DOSIJE D FORALL ISPIT I
IS_EMPTY (( D JOIN I )
WHERE I.INDEKS > 20150000
AND I.INDEKS = D.INDEKS
AND GODINA_ROKA=GODINA_ROKA(2015);
```

b) Ograničenje relacije

```
CONSTRAINT REL1
IF NOT ( IS_EMPTY ( PREDMET ) ) THEN
COUNT ( PREDMET
WHERE SIFRA= SIFRA ('R270')) > 0
END IF;
```

c) Ograničenje atributa

```
VAR PREDMET BASE RELATION {
    ID_PREDMETA INTEGER,
    SIFRA
    SIFRA ,
    NAZIV
    NAZIV ,
    BODOVI
    SMALLINT
}
```

d) Ograničenje tipa

```
TYPE POINT POSSREP
CARTESIAN (X RATIONAL, Y RATIONAL)
CONSTRAINT ABS
(THE_X (POINT)) <= 100.0 AND
ABS(THE_Y (POINT)) <= 100.0 ;
```

d) Ograničenje prelaza na primeru studentske baze: student ne može da sluša neki predmet koji nije na tom smeru

4.4 Referencijalni integritet

Relacija koja sadrži primarne ključeve se naziva roditelj relacija, a relacija koja sadrži spoljašnje ključeve koji se referišu na roditelj relaciju se naziva dete relacija. ***Referencijalni integritet*** - baza ne sme da sadrži neuparene vrednosti spoljašnjih ključeva.

• Relvar-i koji nemaju kandidate za ključ (tj. sadrže duple slogove) se ponašaju nepredvidivo u pojedinim situacijama

• Sistem koji ne poseduje znanje o kandidatima za ključ ponekad pokazuje karakteristike koje nisu "čisto relacione".

Definicija stranog ključa:

```
FOREIGN KEY lista atributa
REFERENCES ime relvar-a
```

Referencijalni ciklus je specijalni slučaj referencijalnog integriteta kod koga roditelj tabela i dete tabela predstavljaju istu tabelu, ili se, u slučaju da u referencijalnom nizu postoji više tabela, poslednja tabela referencijalnog niza referise na prvu tabelu. Ako u ciklusu učestvuje samo jedna tabela tada se on može formirati naredbom CREATE TABLE. U suprotnom, naredba koja je neophodna za formiranje referncijalnog ciklusa je ALTER TABLE pomoću koje se definišu spoljašnji ključevi kojima se zatvara referencijalni ciklus.

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

4.5 Pravila referencijalne akcije

Ova pravila mozemo da primenjujemo pri *create table* ili *alter table* navodeći opcije *on delete [pravilo]* ili *on update [pravilo]*. Ova pravila će biti primenjivana na redove dete tabele koja je spoljašnjim ključem povezana sa roditelj tabelom.

Pri brisanju postoje pravila:

- NO ACTION(default)
- RESTRICT- CASCADE
- SET NULL

NO ACTION i RESTRICT - ako je specificirano neko od ova dva pravila, prijavi se greška i nijedan red nije obrisan.
CASCADE - kada se obriše red u roditelj tabeli, svi redovi dete tabele povezani sa roditelj tabelom se takodje brišu.
SET NULL - kada se obriše red u roditelj tabeli, svi redovi dete tabele povezane sa roditelj tabelom su postavljeni na NULL (ako je moguće njihove vrednosti postaviti na NULL).
Pravlila za azuriranje su:
- NO ACTION(default)
- RESTRICT

NO ACTION - je default, a jedina alternativa je RESTRICT. Razlika izmedju RESTRICT i NO ACTION je u tome što se RESTRICT primenjuje pre bilo kojih drugih referencijalnih ograničenja za menjanje kao što su CASCADE i SET NULL, dok se NO ACTION primenjuje posle svih drugih referencijalnih ograničenja. Pri prijavljivanju greške za NO ACTION i RESTRICT biće drugačiji SQLSTATE.
Kod unošenja nemamo dodatne naredbe, pravilo za unošenje je takvo da ako se nešto unosi u roditelj tabelu, nikada neće biti uneseno u dete tabelu koja je povezana sa roditelj tabelom osim ako već postoji vrednost u odgovarajućim kolonama povezanim sa roditelj tabelom.

4.6 Dodatne mogućnosti

Objekti u DB2 nad kojima se primenjuje DDL: memorijske grupe, baze podataka, tabele, sheme, prostori za čuvanje tabela, funkcije, pul bafera, alias i sinonimi, okidači, katanci, pogledi, indeksi, planovi i paketi, log datoteke...

Naredbe za definisanje podataka (DDL):
CREATE - formira nove objekte. Sintaksa: CREATE <objekat> ... Objekti na koje može da se primeni: memorijske grupe, baze podataka, sheme, prostori za čuvanje tabela, tabele, pogledi, korisnički definisane funkcije, pul bafera, aliasi i sinonimi, okidači, planovi i paketi, indeksi, serveri...
ALTER - menja karakteristike postojećih objekata. Sintaksa: ALTER <objekat> ... Objekti na koje može da se primeni: memorijske grupe, particije baze podataka, tabele, pogledi, prostori za čuvanje tabela, pula bafera, nadimci, serveri, sekvence, indeksi, korisnički definisane funkcije i tipovi..
DROP - koristi se za fizičko brisanje objekata na tekućem serveru. Svi objekti koji direktno ili indirektno zavise od obrisanih objekata se takodje brišu. Sintaksa: DROP <objekat>...
DECLARE - slična CREATE naredbi sem što se koristi za formiranje privremenih tabela koje se koriste jedino za vreme tekuće sesije. Koristi se u svrhu optimizacije. Jedini objekat koji može da se deklariše je tabela koja se upisuje u privremeni prostor za čuvanje tabela korisnika. Sintaksa: DECLARE <objekat> ...

Naredbe za obradu podataka (DML):
SELECT - prikazuje rezultat SQL upita
INSERT - unosi vrednosti u tabelu (pogled). Sintaksa:
INSERT INTO <objekat> VALUES/WITH/<puna select naredba> ...
UPDATE - ažurira postojeće vrednosti. Sintaksa: UPDATE <objekat> SET...
DELETE - prazni sadržaj objekata. Sintaksa: DELETE <objekat> [WHERE uslov]
MERGE - ažurira ciljni objekat na osnovu podataka iz izvornih objekata. Sintaksa: MERGE <objekat> USING...

4.7 Pogledi

Relacioni sistemi podržavaju **poglede** - imenovane relvar-e. Vrednost pogleda je rezultat izvršavanja odredjenog relacionog izraza u tom trenutku. Relacioni izraz se navodi pri formiranju pogleda. Sistem pretvara upit naveden pri formiranju pogleda u ekvivalentan upit nad osnovnim relvar-ima. Pretvaranje se vrši supstitucijom koja je moguća na osnovu osobine relacionog zatvorenja. Pogled može da bude definisan i nad drugim pogledom, a ne samo nad osnovnim relvar-om.
Sintaksa naredbe za formiranje pogleda:

$$VAR < ime\ relvar - a > VIEW < relacioni\ izraz >< lista\ kandidata\ za\ ključeve >$$

[RESTRICT/CASCADE]

• lista kandidata za ključeve može biti i prazna ako pogled može da nasledi kandidate za ključeve.
• RESTRICT/CASCADE - može ali ne mora da postoji

Sintaksa naredbe za brisanje pogleda:

$$DROP\ VAR < ime\ relvar - a >$$

Definicija pogleda kombinuje spoljašnju šemu, preslikavanje izmedju spoljašnjeg i konceptualnog nivoa (sadrži izgled spoljašnjeg objekta i opis kako se on preslikava na konceptualni nivo), spoljašnje-spoljašnje preslikavanje.

Funkcije pogleda: obezbedjuju automatsku zaštitu za skrivene podatke; omogućuju da različiti korisnici istovremeno vide iste podatke na različite načine; uprošćavaju složene operacije; omogućuju logičku nezavisnost podataka.
Logička nezavisnost podataka: proširenje relacije baze ne sme da ima efekat na izvršavanje aplikativnih programa; restruktuiranje baze ne sme da ima efekat na postojeće aplikativne programe; nova i stara baza treba da budu informaciono ekvivalentne.

Operacija čitanja podataka preko pogleda se konvertuje u ekvivalentnu operaciju nad osnovnim relvar-ima - materijalizacija relacije koja je trenutna vrednost pogleda i sustitucija relacionog izraza u drugom relacionom izrazu. Semantika pogleda se definiše preko materijalizacije relacija. Sve operacije sa čitanjem podataka preko pogleda bi trebalo da rade korektno, ali u praksi to nije slučaj, pogotovu ako se pogledi ne materijalizuju.

Zlatno pravilo se primenjuje i na poglede tj. ažuriranje pogleda ne sme da naruši ograničenja integriteta nad pogledima. Ograničenje integriteta nad pogledima su izvedena iz ograničenja integriteta osnovnih relvar-a.
Ako je D baza, V pogled nad D i X funkcija nad D kojom se definiše pogled V, tada za dati pogled V = X(D) i operaciju ažuriranja U nad V, potrebno je odrediti operaciju ažuriranja U_1 nad D tako da važi U(X(D)) = X (U_1(D)). Moguće je da postoji više operacija U1, pitanje je koju odabrati. Codd-ov pristup: definisanje pogleda koji mogu da se ažuriraju. Date-in pristup: svi pogledi mogu da se ažuriraju. Operacije ažuriranja se izvode izbegavanjem ograničenja integriteta u medjukoracima ažuriranja - ažuriranje se izvodi kao brisanje postojećih i unošenje novih podataka.

SQL podrška:
CREATE VIEW naredba
CREATE [OR REPLACE] VIEW <ime pogleda> AS <izraz nad tabelom> [WITH [<kvalifikator>] CHECK OPTION]

DROP VIEW <ime pogleda>

Podrška za ažuriranje pogleda je vrlo ograničena. Jedini pogledi koji se smatraju mogućim za ažuriranje su pogledi koji su izvedeni iz jedne osnovne tabele preko kombinacija restrikcije i projekcije. U SQL/92 pogled može da se ažurira ako važi: izraz kojim se definiše pogled je *select* izraz koji ne sadrži JOIN, UNION, INTERSECT ili EXCEPT; *select* klauzula ne sadrži ključnu reč DISTINCT; svaka *select* stavka sadrži kvalifikovano ime koje predstavlja referencu na kolonu osnovne tabele; *from* klauzula sadrži referencu na tačno jednu tabelu koja je ili osnovna tabela ili pogled koji može da se ažurira; *where* klauzula *select* izraza ne sadrži podupit u kome se *from* klauzula referiše na istu tabelu kao i *from* klauzula u *select* izrazu na najvišem nivou; *select* izraz ne sadrži *group by* niti *having* klauzulu.

4.8 OLAP

OLAP (online analytical processing) - interaktivni proces formiranja, upravljanja, analiziranja i prikaza podataka. Obično se podaci sa kojima se radi posmatraju i sa njima se upravlja kao da se čuvaju u višedimenzionalnom nizu.
Proces analize zahteva odredjenu *agregaciju podataka*, obično na različite načine i prema različitim grupisanjima. Problemi: pravljenje više sličnih ali neznatno različitih upita je dosadno i zamorno za korisnika; izvršavanje svih upita može da bude jako skupa operacija. Sa više nivoa agregacije u jednom upitu se olakšava posao korisniku i nudi se mogućnost da se sve agregacije izračunaju mnogo efikasnije.
OLAP softverski paketi često prikazuju rezultate ne u obliku SQL tabela, već u obliku ukrštenih tabela. ***Ukrštena tabela*** je višedimenziona tabela koja sadrži vrednosti zavisnih atributa i koja je indeksirana sa ključnim atributima SQL tabela.

5 Normalizacija i funkcionalne zavisnosti

5.1 Definicija funkcionalne zavisnosti

DEF1 Neka je R relacija i neka su X i Y proizvoljni podskupovi atributa iz R. Tada *Y funkcionalno zavisi od X* (X funkcionalno odredjuje Y), u oznaci X $\rightarrow$ Y, akko je svakoj važećoj vrednosti torke X u R pridružena tačno jedna vrednost Y iz R.

$$\exists f : f(X) = Y$$

DEF2 *Funkcionalna zavisnost* u relaciji R je tvrdjenje oblika "Ako su dve torke od R identične na svim atributima A_1,..., A_n tada moraju da imaju iste vrednosti i u ostalim atributima B_1,..., B_m". Dve torke su identične ako su im identične vrednosti respektivnih komponenata.

DEF Podskup atributa X $\subseteq$ R relacije R je kandidat za ključ relacije R ako važi:

$$\forall Y : Y = R \setminus X \Longrightarrow X \rightarrow Y$$

$$\nexists Z, W : (Z \in X) \wedge (W = R \setminus Z) \wedge (Z \rightarrow W)$$

Superključ relacije R je skup atributa koji uključuje kao podskup bar jedan kandidat za ključ relacije R.
Svaka funkcionalna zavisnost predstavlja ograničenje integriteta. Posledica je da svaki atribut relacije funkcionalno zavisi od nekog kandidata za ključ. Funkcionalna zavisnost ne zavisi od trenutne vrednosti relvar-a.
Dva skupa funkcionalnih zavisnosti S i T nad relacijom R su ekvivalentni ako je skup instanci relacije R koji zadovoljava S jednak skupu instanci relacije R koji zadovoljava T. U opštem slučaju, skup S funkcionalne zavisnosti se izvodi iz skupa T funkcionalne zavisnosti ako svaka instanca relacije koja zadovoljava sve FZ iz T takodje zadovoljava sve FZ iz S. Posledica je da su dva skupa FZ S i T ekvivalentni akko se svaka FZ u S izvodi iz FZ u T i obrnuto.

5.2 Pravila i zatvorenje

DEF (*Dekompozicija*) FZ A_1,...,$A_n \rightarrow B_1$,...,B_m je ekvivalentna sa skupom FZ

$$A_1, A_2, ..., A_n \rightarrow B_1$$

$$A_1, A_2, ..., A_n \rightarrow B_2$$

$$...$$

$$A_1, A_2, ..., A_n \rightarrow B_m$$

DEF (*Kompozicija*) Skup FZ

$$A_1, A_2, ..., A_n \rightarrow B_1$$

$$A_1, A_2, ..., A_n \rightarrow B_2$$

$$...$$

$$A_1, A_2, ..., A_n \rightarrow B_m$$

je ekvivalentan sa FZ A_1,...,$A_n \rightarrow B_1$,...,B_m

DEF *Trivijalna zavisnost* je FZ koja ne može a da ne bude zadovoljena za bilo koji skup vrednosti u relaciji. $Y \subseteq X \Rightarrow FZ\ X \to Y$ je trivijalna.

DEF Neka je S skup FZ nad relacijom R. Skup svih FZ koje mogu da se izvedu iz skupa S FZ se naziva *zatvorenje* od S i označava se sa $\{S\}^+$.
Posledica je da su dva skupa funcionalnih zavisnosti S i T nad relacijom R ekvivalentni ako važi $\{S\}^+ = \{T\}^+$.
Zatvorenje $\{S\}^+$ skupa FZ S može da se odredi primenom pravila (Armstrongovim aksiomama) kojima se nove FZ izvode iz postojećih. Neka su A, B i C proizvoljni podskupovi atributa relacije R. Tada važe pravila:

$$Refleksivnost: \ B \subseteq A \Rightarrow A \to B$$

$$Prosirenje: \ A \to B \Rightarrow AC \to BC$$

$$Tranzitivnost: A \to B \wedge B \to C \Rightarrow A \to C$$

Iz prethodna tri se mogu izvesti dodatna pravila:

$$Samoodredjenje: A \to A$$

$$Dekompozicija: A \to BC \Rightarrow A \to B \wedge A \to C$$

$$Unija: A \to B \wedge A \to C \Rightarrow A \to BC$$

$$Kompozicija: A \to B \wedge C \to D \Rightarrow AC \to BD$$

$$Opsta\ teorema\ unifikacije: A \to B \wedge C \to D \Rightarrow A \cup (C - B) \to BD$$

Algoritam za računanje zatvorenja skupa FZ F:
Inicijalno $F^+ = F$;
repeat
za svaku FZ f $\in F^+$
primeniti refleksivnost i proširenje na f
dodati dobijene FZ u F^+
za svaki par FZ (f_1, f_2) $\in F^+$
ako f_1 i f_2 mogu da se kombinuju pomoću tranzitivnosti
tada dodati dobijenu FZ u F^+
until više ne bude promena u F^+

Često treba izračunati da li data FZ pripada zatvorenju skupa FZ. U praksi se odredjivanje zatvorenja FZ relativno retko radi. Da bi odredili da li je K nadključ treba odrediti skup atributa relacije R koji funkcionalno zavisi od K. Ako je skup atributa K nadključ, tada K funkcionalno odredjuje sve atribute u R. K je nadključ akko je $\{K\}^+$ od K jednako skupu svih atributa relacije R.

DEF Neka je $A = \{A_1, A_2, ..., A_n\}$ skup atributa relacije R i S skup FZ nad R. *Zatvorenje* skupa atributa A u odnosu na skup S FZ je skup atributa B takvih da svaka relacija koja zadovoljava sve FZ u skupu S zadovoljava i FZ $A_1, A_2, ..., A_n \to B$.
Zatvorenje skupa atributa A se označava sa $\{A\}^+$. Uvek važi da su svi pojedinačni atributi iz A u $\{A\}^+$.

Neka je $A = \{A_1, ..., A_n\}$ skup atributa i S skup FZ. Algoritam za odredjivanje zatvorenja $\{A\}^+$ je:
1. Izvršiti dekompoziciju svih FZ tako da imaju samo jedan atribut na desnoj strani.
2. Neka je X skup atributa koji predstavlja zatvorenje. Inicijalno $X = \{A_1, A_2, ..., A_n\}$
3. Tražiti FZ oblika $B_1, B_2, ..., B_m \to C$ takve da $\forall i : B_i \subset X \wedge C \nsubseteq X$. Dodati C u X. Ponavljati pretragu sve dok ima promena skupa X.
4. Ukoliko ne postoji atribut koji bi mogao da se doda skupu X, tada je $X = \{A_1, A_2, ..., A_n\}^+$

Ako su S_1 i S_2 dva skupa FZ i ako je svaka FZ iz S_1 uključena u S_2 tj. ako je $\{S_1\}^+$ podskup od $\{S_2\}^+$, tada je S_2 pokrivač za S_1. Ako je S_1 pokrivač od S_2 i S_2 pokrivač od S_1, tada su S_1 i S_2 ekvivalentni. Ako RSUBP obezbedi FZ u S_2, tada će automatski biti obezbedjene i FZ u S_1.

DEF FZ $X \to Y$ u skupu S funkcionalnih zavisnosti je *levo-nereducibilna* ako iz X ne može da se ukloni ni jedan atribut bez promene zatvorenja $\{S\}^+$.
Nadključ K je kandidat za ključ relacije R akko je njen nereducibilni nadključ.
DEF Skup FZ je *nereducibilan* ako:
- desna strana svake FZ u S sadrži tačno jedan atribut
- leva strana svake FZ je levo nereducibilna
- ni jedna FZ ne može da se ukloni iz S bez promene $\{S\}^+$

Skup FZ koji zadovoljava prethodna pravila se naziva i minimalan ili kanonički.

Algoritam za nalaženje nereducibilnog skupa S FZ:
- koristeći pravilo dekompozicije prepisati sve FZ u S tako da desna strana sadrži tačno jedan atribut
- eliminisati sve redundantne FZ iz skupa funkcionalnih zavisnosti dobijenih prethodnim postupkom

Neka se relacija R sa skupom S FZ projektuje na relaciju $R_1 = \pi\{R\}$. Koje FZ su važeće u R_1? Odredjuje se *projekcija* FZ S koja sadrži sve FZ takve da:
- mogu da se izvedu iz S
- uključuju jedino atribute iz R_1

Algoritam:
- neka je T skup FZ u R_1. Inicijalno $T = \emptyset$
- $\forall X \subseteq R_1$ odrediti $\{X\}^+$ u odnosu na FZ u S. Atributi koji su u R ali ne i u R_1 mogu da se koriste u izračunavanju $\{X\}^+$. Dodati u T sve netrivijalne FZ $X \to A$ takve da $A \subset \{X\}^+ \wedge A \subset R_1$
- odredjuje se minimalni skup T na sledeći način: ako postoji $F \subset T$ koja može da se izvede iz drugih FZ u T, ukloniti FZ F. Neka je $Y \to B$ FZ u T sa najmanje dva atributa u Y i neka je Z dobijeno iz Y uklanjanjem jednog od atributa. Ako $Z \to B$ može da se izvede iz FZ u T (uključujući $Y \to B$), tada se $Y \to B$ zamenjuje sa $Z \to B$. Ponoviti prethodne korake sve dok ima promena u T.

5.3 Normalizacija

U logičko projektovanje baza spada: normalizacija - korišćenje ideja o normalizaciji radi razbijanja velikih u male relacije, i semantičko modeliranje - upotreba modela entiteta i odnosi radi formiranja velikih relacija.
Normalizacija je proces zamene relacija skupom relacija koje su u pogodnijem obliku. Svrha normalizacije je izbegavanje redundantnosti i pojedinih anomalija ažuriranja. U procesu normalizacije operator projekcije se više puta primenjuje na datu relaciju na takav način da spajanjem projekcija može da se dodje do početne relacije. Na taj način, proces normalizacije je reverzibilan i čuva informacije tj. uvek je moguće da se uzme izlaz iz procesa i preslika unazad do ulaza. Postoji 6 normanih formi, van toga su relacije koje nisu realizovane.

5.4 Normalne forme

DEF Relvar je u *1NF* akko u svakoj važećoj vrednosti tog relvar-a svaka torka sadrži tačno jednu vrednost za svaki atribut (vrednosti atributa su uvek atomične).

Nereducibilna funkcionalna zavisnost: Ako $A \to B$ i uklanjanje bilo kog atributa iz A povlači da $A \to B$ postaje netačno, tada je B nereducibilno zavisno od A. Nereducibilna definicija podrazumeva postojanje samo jednog kandidata za ključ koji je istovremeno i primarni ključ.
DEF Relvar je u *2NF* akko je u 1NF i svaki neključni atribut nereducibilno zavisan od primarnog ključa.

DEF1 Relvar je u *3NF* akko je u 2NF i svaki neključni atribut je netranzitivno zavisan od primarnog ključa (ne zavisi od atributa drugih neključeva).
Prethodne definicije podrazumevaju postojanje samo jednog kandidata za ključ koji je istovremeno i primarni ključ. Posledica je da su neključni atributi uzajamno nezavisni.
DEF2 Relvar R je u *3NF* akko za svaku FZ: $A_1A_2...A_n \to B_1B_2...B_m$ koja nije trivijalna važi:

- $\{A_1A_2...A_n\}$ je superključ ili
- svaki atribut $B_1B_2...B_m \notin \{A_1A_2...A_n\}$ je element nekog (ne obavezno istog) kandidata za ključ

Kodova originalna definicija 3NF nije uzimala u obzir slučajeve kada relacija: ima više od jednog kandidata za ključ; kandidat za ključ je kompozitan; kompozitni kandidati za ključeve se preklapaju. Ovi slučajevi su obuhvaćeni Bojs-Kodovom normalnom formom.

DEF Relvar je u *BCNF* akko je u 3NF i svaka netrivijalna levo-nereducibilna FZ ima kandidat za ključ (superključ) kao svoju levu stranu (jedini kandidati za ključ su leve strane FZ).
Relvar je u BCNF akko su jedini kandidati za ključ leve strane FZ.
Poželjne osobine pri redukciji:

- eliminacija anomalija
- mogućnost rekonstrukcije informacija
- očuvanje FZ

Pravila:

- sve FZ polaznog skupa moraju da budu očuvane (direktno ili mogućim izvodjenjem iz skupa relacija dobijenih dekompozicijom)
- ako u novodobijenim projekcijama nastalim razbijanjem osnovne relacije postoji zajednički atribut, on mora da bude ključ u bar jednoj od novodobijenih relacija

Algoritam za dekompoziciju relacije R u BCNF:

1. proveriti da li je relacija R u BCNF. Ako jeste, proces je završen i $\{R\}$ je rešenje
2. ako FZ $X \to Y$ narušava BCNF, tada za nove relacije uzeti $R_1 = \{X\}^+$ i $R_2 = \{X\} \cup (R - \{X\}^+)$
3. odrediti skup FZ za relacije R_1 i R_2
4. primeniti rekurzivno algoritam na relacije R_1 i R_2, uzimajući uniju svih dekompozicija kao rešenje

DEF Neka je R relvar i neka su A, B i C podskupovi atributa od R. Kaže se da je B *višeznačno zavisno* od A, u oznaci $A \to\to B$ akko u svakoj mogućoj važećoj vrednosti od R, skup vrednosti B koji se uparuje sa parom (vrednost A, vrednost C) zavisi jedino od vrednosti A i nezavisan je od vrednosti C.

DEF Relvar R je u *4NF* akko je u BCNF i svaki put kada postoje podsupovi A i B atributa od R takvi da je zadovoljena netrivijalna višeznačna zavisnost $A \to\to B$, tada su svi atributi od R takodje funkcionalno zavisni od A.

VZ $A \rightarrow\rightarrow B$ je trivijalna ako je ili A nadskup od B ili $A \cap B$ sadrži sve atribute od R.
Neka je R relvar i neka su A, B,..., Z podskupovi atributa od R. Tada R zadovoljava *zavisnost spajanja* $*\{A, B, ..., Z\}$ akko je R u 4NF i svaka moguća važeća vrednost u R je jednaka spajanju njenih projekcija na A, B,..., Z.

DEF Relvar je u *5NF* (projekcija-spajanje-NF) akko je R u 4NF i svaka netrivijalna zavisnost spajanja koja važi u R je posledica kandidata za ključ u R, gde je: zavisnost spajanja u R trivijalna akko je najmanje jedan od A,B,...,Z skup svih atributa R, i zavisnost spajanja u R je posledica kandidata za ključ relvar-a R akko je svaki od A,B,...,Z nadključ za R.

5.5 Proces normalizacije

Normalizacija dovodi do mnogo sitnih relacija, što je nepoželjno. Nekada se zbog toga ne ide do kraja, već se duplikati ostave, zbog efikasnosti.
Relaciona promenljiva R je:

1) normalizovana akko je u 1NF

2) u 2NF akko za svaki ključ K u R i svaki neključni atribut A iz R FZ $K \rightarrow \{Y\}$ koja važi u R je nereducibilna (tj. K je minimalni ključ)

3) u 3NF akko svaka netrivijalna FZ $X \rightarrow Y$ koja važi u R ili je X superključ ili je podskup ključa

4) u BCNF akko u svakoj netrivijalnoj FZ $X \rightarrow Y$ koja važi u R, X je superkluč

Proces normalizacije:

1. uzeti projekcije originalnog relvar-a u 1NF radi eliminisanja FZ koje nisu nereducibilne. Dobijeni skup relvar-a je u 2NF.

2. uzeti projekcije u 2NF radi eliminisanja tranzitivnih zavisnosti. Dobijeni skup relvar-a je u 3NF.

3. uzeti projekcije relvar-a u 3NF radi eliminisanja preostalih FZ u kojima na levoj strani nije kandidat za ključ. DObijeni skup relvar-a je u BCNF.

4. uzeti projekcije relvar-a u BCNF radi eliminisanja VZ koje nisu u FZ. DObijeni skup relvar-a je u 4NF.

5. uzeti projekcije relvar-a koji su u 4NF i eliminisati ZS koje ne slede iz kandidata za ključ(eve). Dobijeni skup relvar-a je u 5NF.

U praksi se često ne sprovodi puna normalizacija zbog dobrih performansi. Puna normalizacija dovodi do velikog broja logički razdvojenih relvar-a. Veliki broj razdvojenih relvar-a znači veliki broj razdvojenih datoteka u kojima se čuvaju. Veliki broj datoteka znači veliki broj U/I operacija. U praksi se normalizacija najčešće sprovodi do 3NF.

6 Sigurnost podataka i autorizacija

Sigurnost - zaštita podataka od neautorizovanih korisnika (zaštita protiv neautorizovanog pristupa, promene ili uništenja).
Integritet - zaštita podataka protiv autorizovanih korisnika (obezbedjenje ispravnosti i korektnosti podataka)
Sličnosti izmedju sigurnosti i integriteta:

- sistem mora da bude svestan izvesnih ograničenja koje korisnici ne smeju da prekrše
- ograničenja moraju da budu zadata (od strane DBA) u nekom jeziku
- ograničenja moraju da budu evidentirana u sistemskom katalogu (rečniku podataka)
- SBP mora da vrši nadzor nad operacijama korisnika.

Aspekti problema sigurnosti:

- pravni, socijalni i etički
- fizička kontrola
- politička pitanja
- operativni problemi
- hardverska kontrola
- podrška operativnog sistema
- problemi vezani za same baze podataka

Jedinica podataka na koju se osigurava: baza podataka, relvar, pojedinčna torka ili vrednost atributa, aliasi, indeksi, šeme, paketi, prostori za čuvanje tabela...

6.1 Sigurnost u SQL-u

Sigurnost se obično zapisuje preko kontrolne matrice pristupa - predstavljanje po korisnicima, po objektima, po dozvoli za pristup. Postoje dva mehanizma koja su nezavisno jedan od drugog uključeni u sistem zaštite:

1. *pogledi* koji mogu da se koriste za sakrivanje osetljivih podataka od neautorizovanih korisnika

2. *podsistem za autorizaciju* koji dopušta korisniku sa odredjenim pravima pristupa da ta prava selektivno i dinamički prenosi na druge korisnike i/ili da preneta prava povuče. Da bi korisnik izvršio bilo kakvu operaciju nad nekim objektom on mora da poseduje dozvolu/autorizaciju za tu operaciju nad tim objektom. Tipovi i vrste dozvola nisu isti kod svih SUBP. Sistemski administrator je inicijalni vlasnik svih dozvola. Davanje dozvola se vrši GRANT naredbom, a povlačenje REVOKE naredbom.

Sintaksa GRANT naredbe za dozvole nad tabelama ili pogledima:

$$GRANT\ dozvola\ [ON\ [tip]\ objekat]\ TO\ korisnik,$$

dozvola je lista jedne ili više vrsta dozvola, razdvojenim zarezima ili fraza ALL PRIVILEGES ili ALL koja označava sve privilegije koje se mogu dati GRANT naredbom. Dozvole koje se odnose na osnovne tabele i poglede: CONTROL, DELETE, INSERT, SELECT, UPDATE (mogu da se navedu pojedinačne kolone). Dozvole koje se odnose samo na osnovne tabele: ALTER (dozvola za izvršavanjem ALTER TABLE nad tabelom), INDEX (dozvola za izvršavanje CREATE INDEX nad tabelom), REFERENCES (dozvola za formiranje/brisanje spoljašnjeg ključa koji referiše tu tabelu kao roditelj tabelu)
korisnik je lista korisnika razdvojenih zarezima ili PUBLIC za sve korisnike
objekat je lista imena jednog ili više objekata koji su svi istog tipa, razdvojenih zarezima
tip označava tip objekta; ako se izostavi podrazumeva se TABLE
ON se ne upotrebljava kada se daje dozvola za sistemske privilegije

Ako korisnik K1 želi da prenese dozvolu D korisniku K2, to može da uradi: naredbom GRANT dozvola...; naredbom GRANT dozvola... WITH GRANT OPTION čime omogućuje korisniku K2 da dalje distribuira dozvolu koja mu je preneta.

Sintaksa REVOKE naredbe za dozvole nad tabelama ili pogledima:

$$REVOKE\ dozvola\ [ON\ [tip]\ objekat]\ FROM\ korisnik\ [BY\ ALL]$$

Povlačenje dozvole za nekog korisnika uzrokuje da svi planovi/paketi zasnovani na toj dozvoli postanu neispravni i uzrokuju automatsko vezivanje/ponovno vezivanje prilikom pozivanja takvog plana/paketa. Nije moguće ukinuti UPDATE dozvolu samo za pojedine kolone.

Ostali aspekti sigurnosti:
- kompletan sistem treba da bude zaštićen
- ne pretpostavljati da je sistem zaštite savršen
- voditi evidenciju o prijavljivanju na bazu