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.