

Upravljanje izvršenjem transakcija
Oporavak baze podataka
Sigurnost baze podataka
Katalog baze podataka

Upravljanje izvršenjem transakcija

Upravljanje izvršenjem transakcija

Baza podataka je zajednički resurs koji istovremeno (konkurentno) koristi veći broj programa. Pri ovakvom korišćenju baze podataka može doći do mnogih neželjenih efekata kao što su, na primer:

- • otkaz sistema u toku izvršenja nekog programa koji može da ostavi bazu podataka u nekonzistentnom stanju,
- • neželjena interferencija dva ili više programa nad podacima za koje istovremeno konkurišu, može, takođe, da dovede bazu podataka u nekonzistentno stanje.

Osnovni cilj baze podataka je da omogući efikasnu obradu transakcija. **Transakcija** je jedno izvršenje neke "logičke jedinice posla", jedno izvršenje neke logičke celine jednog programa, ili jedno izvršenje celog programa.

Problemi u izvr{avanju transakcija

Otkaz sistema u toku obrade transakcije. Prenos novca (N dinara) komitenta banke sa računa X na račun Y.

Gubljenje rezultata ažuriranja. Transakcija A podiže, a transakcija B ulaže novac na isti račun.

Problem nekorektne analize podataka. Ovim nazivom se definiše skup problema do kojih dolazi kada, za vreme nekog “sračunavanja” koje se obavlja u jednoj transakciji, druga promeni vrednost argument koji je prva već obradila.

Problemi u izvr{avanju transakcija

Otkaz sistema u toku obrade transakcije. Prenos novca (N dinara) komitenta banke sa računa X na račun Y.

Gubljenje rezultata ažuriranja. Transakcija A podiže, a transakcija B ulaže novac na isti račun.

Problem nekorektne analize podataka. Ovim nazivom se definiše skup problema do kojih dolazi kada, za vreme nekog “sračunavanja” koje se obavlja u jednoj transakciji, druga promeni vrednost argument koji je prva već obradila.

Transakcije

Transakcija mora da poseduje sledeći skup osobina (ACID osobine):

1. **Atomnost** (Atomicity). Transakcija, kao "logička jedinica posla", mora predstavljati atomski skup aktivnosti.
2. **Konzistentnost** (Consistency). Izvršenje transakcije treba da prevede bazu podataka iz jednog u drugo konzistentno stanje.
3. **Izolacija** (Isolation). Transakcija ne treba svoje promene baze podataka da učini vidljivim drugim transakcijama pre nego što se ona okonča, odnosno promene potvrde u bazi podataka.
4. **Trajnost** (Durability). Kada su, u bazi podataka, potvrđene promene koje je izvršila neka transakcija, ove promene se više ne mogu izgubiti.

Transakcije

Da bi se ACID osobine transakcije obezbedile skup instrukcija koje predstavljaju transakciju počinje, po pravilu, sa specijalnom instrukcijom `BEGIN TRANSACTION`,

a završava se bilo sa instrukcijom `COMMIT` (potvrđuju se promene u bazi podataka, ako su sve instrukcije transakcije uspešno izvršene)

ili instrukcijom `ROLLBACK` (poništavaju se promene u bazi, ako sve instrukcije nisu uspešno završene).

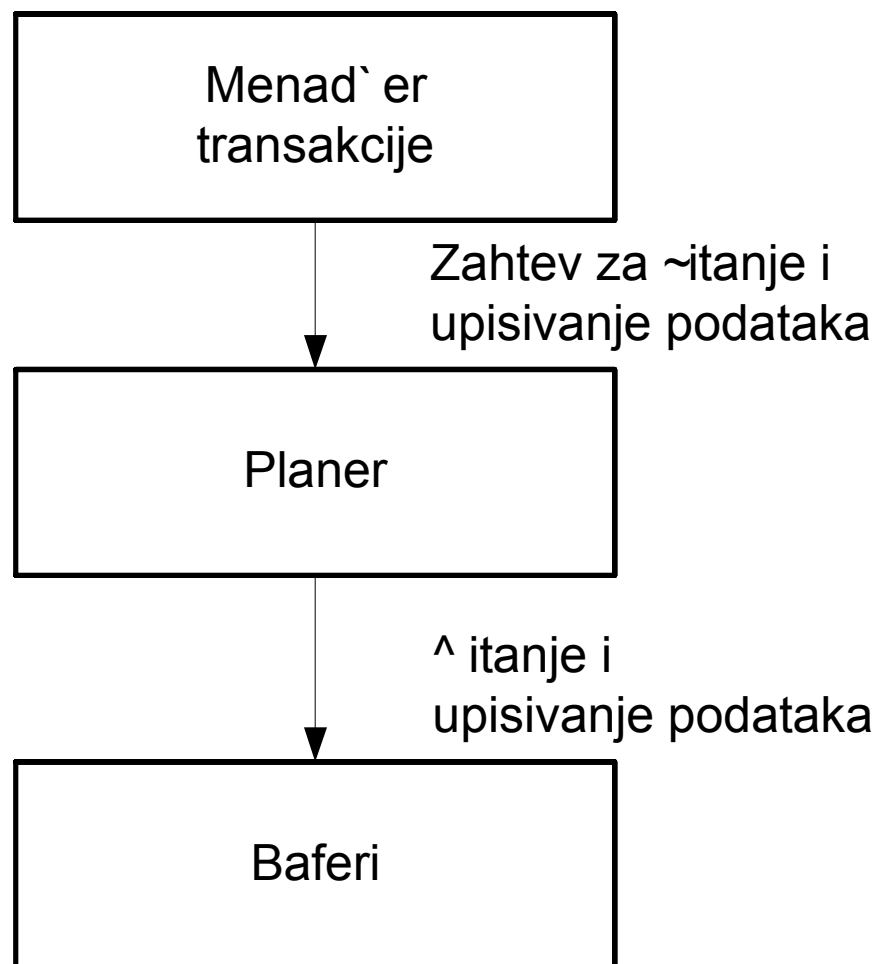
Konkurentna (uporedna) obrada transakcija

Komponenta SUBP-a koja vodi računa o redosledu izvršavanja akcija nad bazom podataka u skupu transakcija koje se konkurentno izvršavaju, naziva se **Planer** (**Scheduler**).

Komponenta koja upravlja celokupnim izvršenjem transakcija naziva se **Menadžer transakcije** (**Transaction manager**).

Na sledećoj slici prikazana je i razmena zahteva i podataka između ovih komponenti i bafera.

Konkurentna (uporedna) obrada transakcija



Planer izvršenja skupa transakcija

Konkurentna (uporedna) obrada transakcija

Rezultat serijskog izvršenja skupa transakcija u bilo kom redosledu se smatra korektnim.

Koristeći ovaj stav, možemo da uvedemo koncept serijabilnosti ili linearnosti izvršenja skupa transakcija.

Uporedno izvršavanje skupa transakcija je ***serijabilno (linearno)*** ako proizvodi isti rezultat kao i neko serijsko izvršenje istog skupa transakcija.

Usvaja se da je skup uporednih transakcija izvršen korektno, ako i samo ako je taj skup serijabilan.

Protokoli za ostvarivanje serijabilnosti izvršenja skupa transakcija

Različiti protokoli serijabilnosti izvršenja skupa transakcija zasnivaju se na različitim načinima utvrđivanja serijabilnosti i različitim postupcima ostvarivanja serijabilnosti.

View-serijabilnost

Za definisanje tzv. view-ekvivalencije dva izvršenja skupa transakcija i view-serijabilnosti datog izvršenja skupa transakcija koriste se sledeći pojmovi:

- • Ako u nekom izvršenju skupa transakcija neko upisivanje $w_j(X)$ prethodi nekom čitanju istog elementa baze $r_i(X)$ i nema nijedne druge operacije upisivanja $w_k(X)$ između njih, kaže se da “ $r_i(X)$ čita iz $w_j(X)$ ”, odnosno da između $r_i(X)$ i $w_j(X)$ postoji “čita iz” odnos.
- • Operacija $w_i(X)$ se zove “krajnje upisivanje”, ako je to poslednje upisivanje elementa X u bazu.

View-serijabilnost

“View” ekvivalencija i “view” serijabilnost se definišu na sledeći način:

- Dva izvršenja skupa transakcija su **view-ekvivalentna** ($S_i \approx_v S_j$) ako poseduju iste “čita iz” odnose i ista “krajnja upisivanja”.

Izvršenje skupa transakcija je **view-serijabilno** ako je view-ekvivalentno sa nekim serijskim izvršenjem.

Zbog kompleksnosti utvrđivanja, view-serijabilnost se praktično ne koristi.

Konflikt-serijabilnost

Jedan od praktičnijih pristupa utvrđivanju serijabilnosti izvršenja skupa transakcija se ostvaruje preko definisanja koncepta konflikt-serijabilnosti. Pod **konfliktom** se ovde podrazumeva situacija u kojoj izmena redosleda dve operacije u izvršenju dovodi do izmene efekata na bazu barem jedne od transakcija iz posmatranog izvršenja.

Ako pretpostavimo da transakcije T_i i T_j pripadaju nekom izvršenju, tada sledeće parovi operacija neće biti u konfliktu:

1. $r_i(X), r_j(Y)$ nije konflikt čak i kada je $X = Y$ jer ni jedna ni druga operacija ne menjaju stanje baze podatka.
2. $r_i(X), w_j(Y)$ očigledno nije konflikt, pod pretpostavkom da je $X \neq Y$.
3. $w_i(X), r_j(Y)$ nije konflikt, pod pretpostavkom da je $X \neq Y$.
4. $w_i(X), w_j(Y)$ nije konflikt.

Konflikt-serijabilnost

Planer ne može da izmeni redosled operacija u okviru jedne transakcije, jer se time menja semantika transakcije. Najopštije, dve susedne operacije različitih transakcija mogu zameniti mesta ako nije zadovoljen jedan od uslova:

1. Operacije se obavljaju nad istim elementom baze podataka i
2. Barem jedna od njih je upisivanje.

Dva izvršenja skupa transakcija su ***konflik-ekvivalentna*** ako se jedan u drugi mogu transformisati nekonfliktnim izmenama mesta susednih operacija.

Izvršenje skupa transakcija je ***konflikt-serijabilno*** ako je konflikt-ekvivalentno sa nekim serijskim izvršenjem.

Protokoli zaključavanja

Proveru serijabilnosti i preduzimanje odgovarajućih akcija planer izvršenja teško može da obavi u realnom vremenu. Zbog toga se serijabilnosti izvršavanja skupa transakcija najčešće ostvaruje “forsirano” primenjujući mehanizam ***zaključavanja (locking)***.

Ekskluzivno zaključavanje (exclusive (XL) lock ili write lock).

Ako neka transakcija postavi ekskluzivni lokot na objekat baze podataka, nijedna druga transakcija ne može na taj objekat da postavi bilo koji drugi lokot.

Deljivo zaključavanje (shared (SL) lock ili read lock). Ako neka transakcija postavi deljivi lokot na objekat baze podataka, neka druga transakcija takođe može da postavi deljivi lokot na isti objekat baze, ali nijedna druga ne može da postavi ekskluzivni lokot na taj objekat.

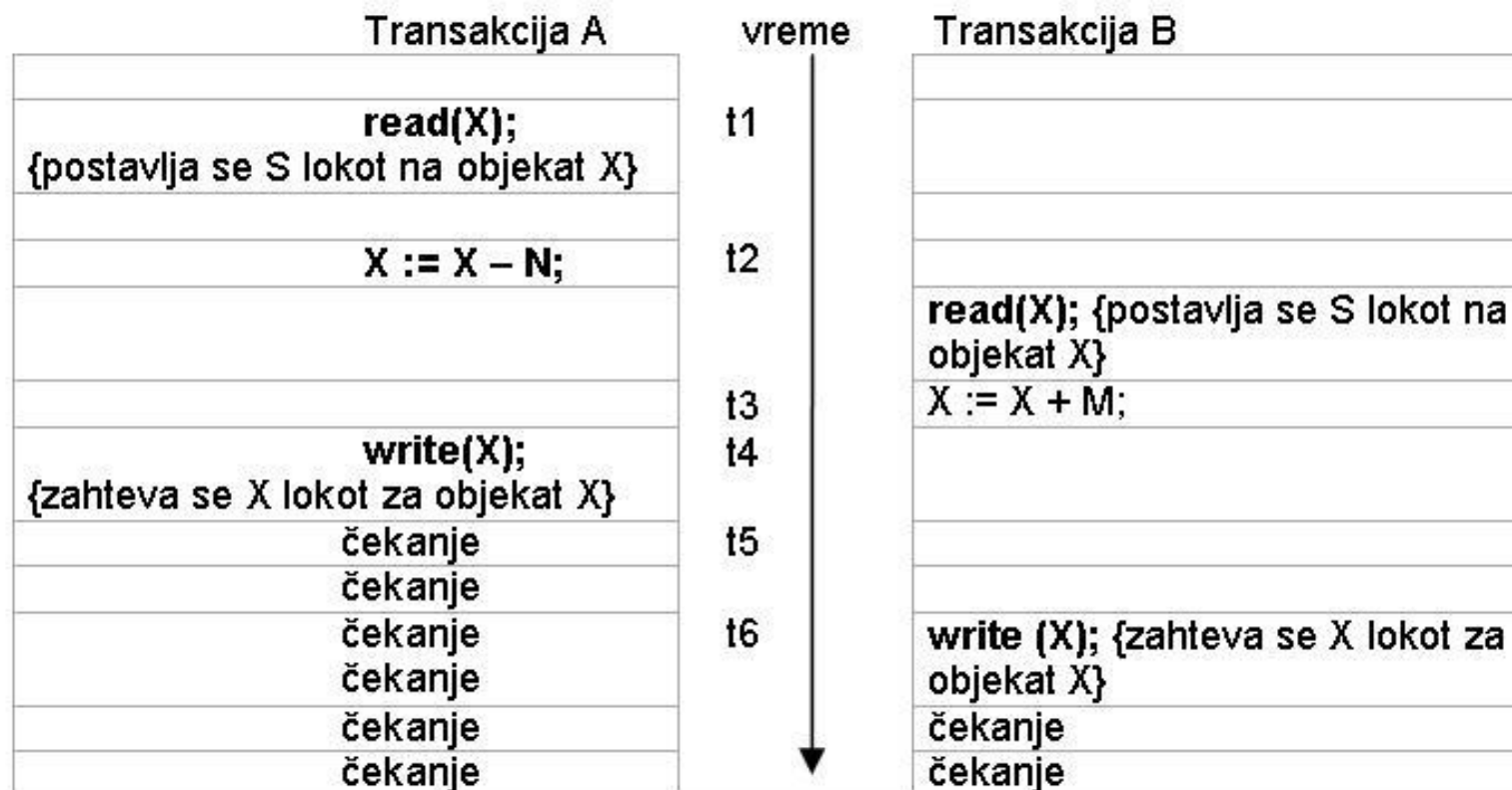
Matrica kompatibilnosti tipova zaključavanja

A \ B	XL	SL	-
XL	NE	NE	DA
SL	NE	DA	DA
-	DA	DA	DA

Imajući u vidu ekskluzivni i deljivi lokot, **protokol zaključavanja** koji bi mogao da reši prikazane probleme može se iskazati na sledeći način:

1. Transakcija koja želi da pročita neki objekat baze podataka (n-torku, na primer) mora prvo da postavi deljivi lokot na taj objekat;
 2. Transakcija koja želi da ažurira neki objekat baze podataka mora prvo da postavi ekskluzivni lokot na taj objekat. Ako je ta transakcije ranije postavila S lokot, ona treba da taj lokot transformiše u X lokot;
 3. Ako transakcija nije uspela da postavi lokot na željeni objekat baze podataka, zbog toga što neka druga transakcija već drži nekompatibilan lokot nad posmatranim objektom, tada ona prelazi u stanje čekanja;
 4. Transakcija oslobađa E lokot obavezno, a po pravilu i S lokot na kraju, sa COMMIT ili ROLLBACK naredbom.
- I X i S lokot se postavljaju implicitno, zajedno sa operacijama ažuriranja, odnosno čitanja podataka baze.

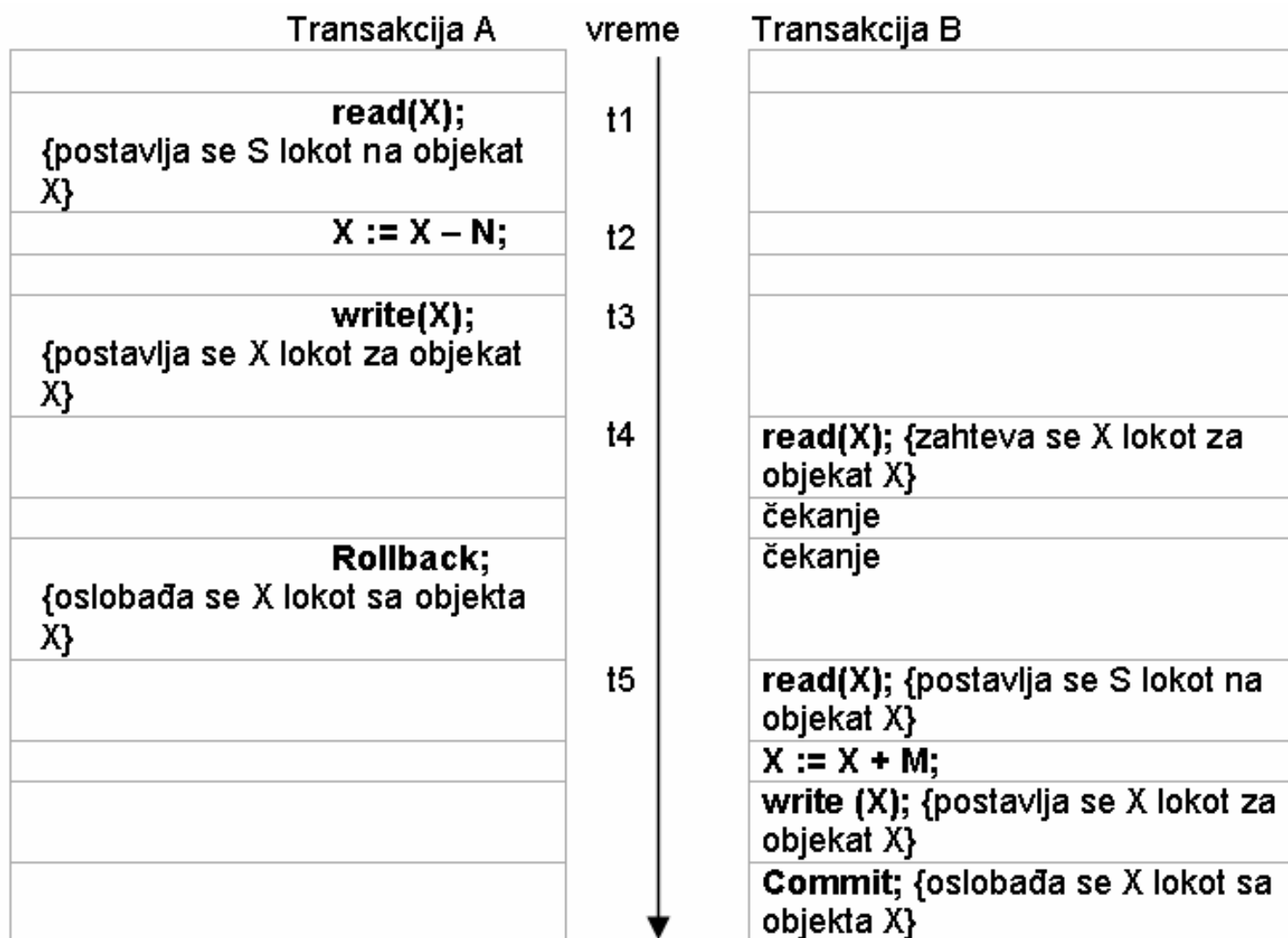
Mrtvi čvor



Slika 11.8. “Mrtvi čvor” umesto gubljenje rezultata ažuriranja

Dvofazni protokol zaključavanja

- *Pre nego što operiše sa nekim objektom baze podataka, transakcija mora da postavi lokot na njega;*
- *Posle oslobađanja nekog lokota, transakcija ne može više postaviti lokot ni na jedan objekat baze.*
- *Može se dokazati teorema da ako u nekom skupu sve transakcije poštuju dvofazni protokol zaključavanja, taj skup se uvek serijabilno izvršava.*



Slika 11.9. Rešenje problema nepotvrđenih promena

Vremensko označavanje (Timestamping)

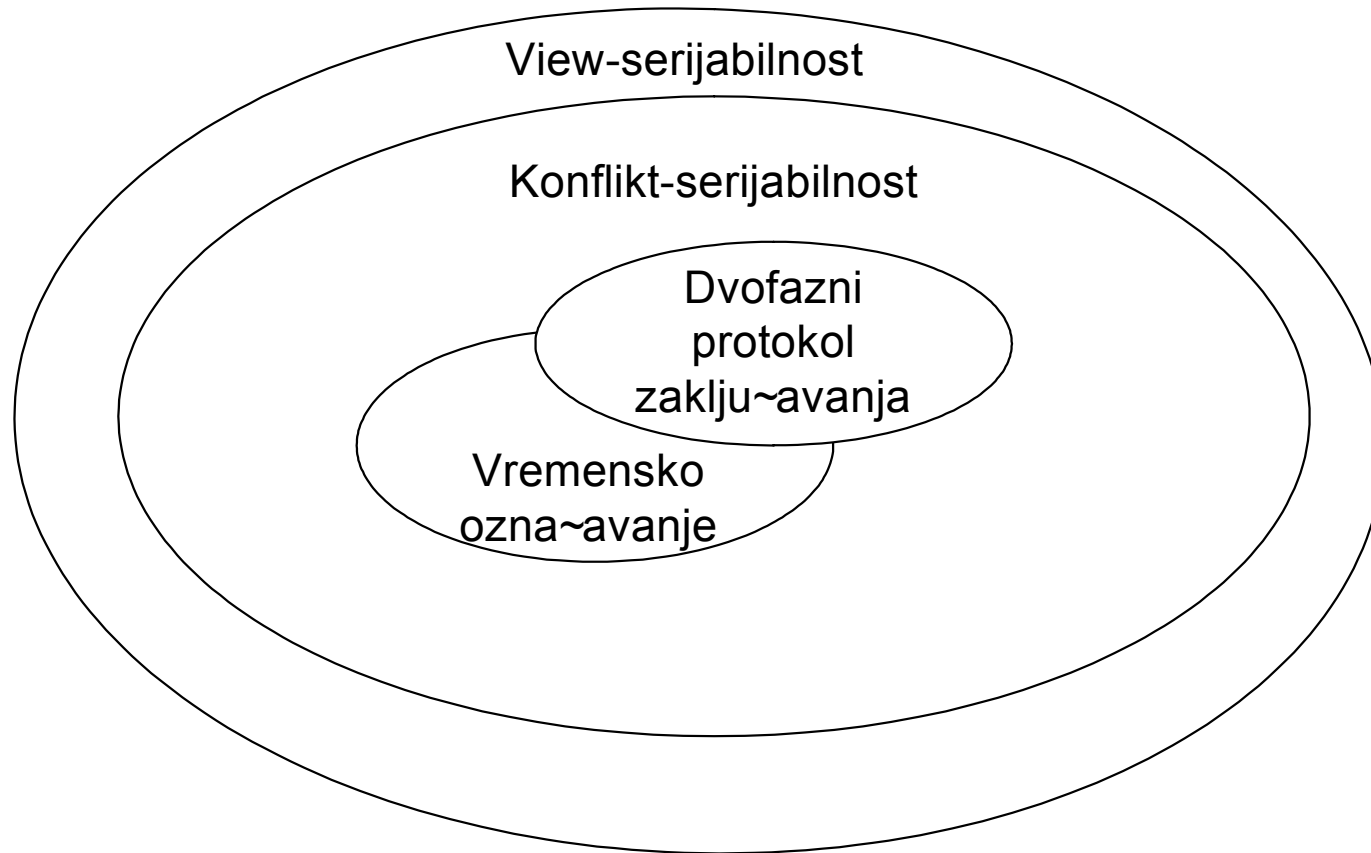
Svakom objektu baze podataka pridružuju se dve oznake:

- RMAX, najveći identifikator transakcije koja je uspešno izvršila čitanje posmatranog objekta i
- UMAX, najveći identifikator transakcije koja je uspešno izvršila ažuriranje posmatranog objekta.

Algoritam vremenskog označavanja

```
read(R)
  if t >= UMAX
  then { operacija se prihvata}
  RMAX := MAX(t, RMAX);
else { konflikt}
restart T;
write(R) { zajedno sa COMMIT}
  if t >= UMAX and t >= RMAX
  then { operacija se prihvata}
  UMAX := t;
else { konflikt}
restart T;
```

Nivoi izolovanosti transakcija



Načini narušavanja serijabilnosti izvršenja skupa transakcija:

- ***Prljavo čitanje (“dirty read”)***. Ovako izvršenje skupa transakcija je već ranije objašnjeno. Ako transakcija T1 ažurira neki podatak u bazi, zatim transakcija T2 pročita taj podatak, a nakon toga se transakcija T1 završi sa ROLLBACK, tada je očigledno transakcija T2 pročitala nepostojeći podatak, odnosno izvršila “prljavo čitanje”.
- ***Neponovljivo čitanje (Nonrepeatable read)***. U jednoj transakciji, ponovno čitanje istih podataka mora dati isti rezultat. Naprotiv, ako transakcija T1 pročita neki podatak baze, zatim ga transakcija T2 promeni, ponovno čitanje istog podatka u transakciji T1 neće dati isti rezultat.
- ***Fantomsko čitanje (Fantoms)***. Ako transakcija T1 preuzme upitom skup n-torki koje zadovoljavaju zadati uslov, a posle toga transakcija T2 doda novu n-torku koja zadovoljava isti uslov, novo izvršenje istog upita u transakciji T1 sadržaće i novu “fantomsku” n-torku.

Odnos nivoa izolovanosti i ponašanje izvršenja skupa transakcija

Ponašanje Nivo izolovanosti	PRLJAVO ČITANJE	NEPONOVLJIVO ČITANJE	FANTOMSKO ČITANJE
READ UNCOMMITTED	DA	DA	DA
READ COMMITTED	NE	DA	DA
REPEATABLE READ	NE	NE	DA
SERIALIZABLE	NE	NE	NE

Upravljanje zaključavanjem

Osnovu upravljanja zaključavanjem čine tri procedure
menadžera lokota:

- (1) `r_lock` – postavljanje ekskluzivnog lokota,
- (2) `w_lock` – postavljanje deljivog lokota i
- (3) `unlock` – oslobađanje zaključanog elementa baze podataka.

```
r_lock (T,X, errcode, timeout)
```

```
w_lock (T,X, errcode, timeout)
```

```
unlock (T,X)
```

T - identifikator transakcije,

X – elemenat baze podataka koji se zaključava,

errcode – daje kod statusa zahteva (0-zahtev zadovoljen, ≠0 - nezadovoljen)

timeout – maksimalni interval vremena koji transakcija “čeka” da bi dobila lokot
nad objektom X.

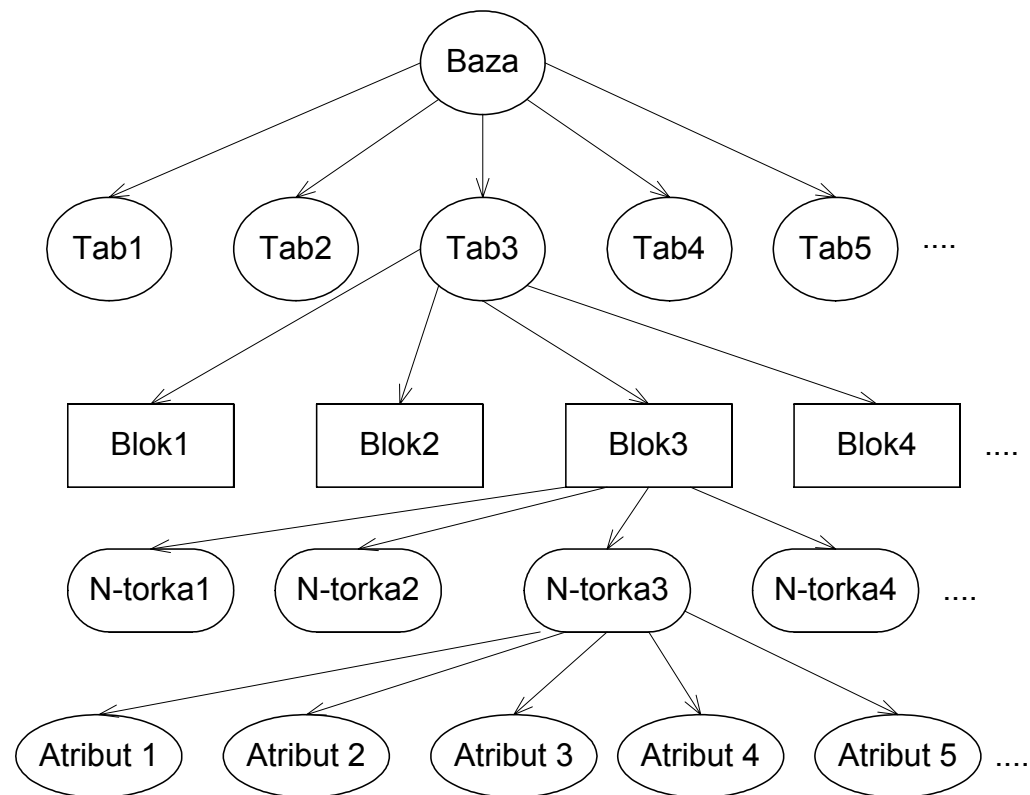
Granularnost zaključavanja i hijerarhijsko zaključavanje

Pored ekskluzivnog lokota (XL) i deljivog lokota (SL) za ovaj protokol se definišu i sledeće vrste lokota:

- **ISL** – nameravani deljivi lokot: transakcija namerava da postavi deljivi lokot na element baze podataka;

- **IXL** - nameravani ekskluzivni lokot: transakcija namerava da postavi ekskluzivni lokot na elementa baze podataka;

- **SIXL** – nameravano deljivo-ekskluzivno zaključavanje: transakcija namerava da postavi deljivi lokot na element baze podataka X i ekskluzivni lokot na neki element na nižem nivou granularnosti koji je deo elementa X.



Granularnost zaključavanja i hijerarhijsko zaključavanje

U hijerarhijskom protokolu zaključavanje se obavlja po sledećim pravilima:

1. Lokoti se zaključavaju počev od vrha, niz hijerarhiju elemenata;
2. Lokoti se oslobađaju počev od elementa sa najmanjom granularnošću, naviše, uz hijerarhiju elemenata.
3. Da bi se postavio SL ili ISL lokot na neki elemenat X, transakcija mora prethodno da postavi ISL ili IXL lokot na element “roditelj” elementu X u stablu;
4. Da bi zahtevala IXL, XI ili SIXL lokot na elemenat X, transakcija mora prethodno da postavi SIXL ili IXL lokot na element “roditelj” elementu X u stablu;
5. Matrica kompatibilnosti ovi lokota za transakcije A i B data je na slici

B A	ISL	IXL	SL	SIXL	XL
ISL	DA	DA	DA	DA	NE
IXL	DA	DA	NE	NE	NE
SL	DA	NE	DA	NE	NE
SIXL	DA	NE	NE	NE	NE
XL	NE	NE	NE	NE	NE

Hijerarhijsko zaključavanje indeksnih struktura

Ako bi se na B-stablima primenjivale standardne vrste lokota (ekskluzivni i deljivi) i standardni protokoli zaključavanja (Dvofazni protokol zaključavanja) tako da se celo B-stablo zaključava, jer se pretpostavlja da će biti potrebno i njegovo ažuriranje, paralelizam u izvršenju skupa transakcija bi se gotovo potpuno eliminisao.

Zbog toga se definišu specijalni protokoli za hijerarhijske, odnosno indeksne strukture. Jedan takav protokol je:

- Transakcija može postaviti svoj prvi lokot na bilo koji čvor stabla;
- Sledeći lokoti se mogu dobiti samo ako je data transakcija postavila lokot i na čvor “roditelj”;
- Lokoti se mogu osloboditi u bilo kom trenutku;
- Transakcija ne može ponovo da postavi lokot na element sa koga je skinula lokot, čak i ako još uvek drži lokot na čvoru “roditelju”.

“Živi” i “mrtvi” lokoti

Kao što je u nekim od prethodnih primera pokazano, u toku izvršenja skupa transakcija, moguće je da dve ili više transakcija formiraju “mrtvi lokot”, odnosno da lokoti koje su one postavile na objekte baze podataka sve njih dovode u stanje čekanja.

Pored pojma “mrtvog lokota”, ponekad se definiše i pojam “živog lokota”. To je situacija u kojoj je neka transakcija stalno u stanju čekanja na neki objekat baze podataka zbog toga što druge transakcije uvek pre nje postave lokot na taj objekat. Problem “živog lokota” jednostavno se rešava uvođenjem nekog redosleda zaključavanja objekta (na primer FIFO).

“Živi” i “mrtvi” lokoti - 2

Za razrešavanje problema mrtvih lokota koriste se tri tehnike:

- 1. *Prekidanje transakcije posle isteka intervala vremena*** predviđenog za čekanje na lokot za neki elemenat. Ovo pravilo koristi parametar “timeout” koji menadžer lokota dodeljuje transakciji pri pokušaju zaključavanja nekog objekta. Kada ovo vreme istekne transakcija se poništava i ponovo startuje, očekujući da tada neće proizvesti “mrtvi lokot”.

“Živi” i “mrtvi” lokoti - 3

2. **Prevenција lokota.** Mogu se definisati različiti protokoli koji će sprečiti da do “mrtvog čvora” dođe. Jedan od takvih protokola se zasniva na uređenju elemenata baze podataka. Na primer, blokovi se mogu urediti po njihovim adresama na spoljnoj memoriji ($A1 < A2 < A3 < \dots < A_n$). Ako se zahteva da svaka transakcija zaključa elemente u njihovom definisanom redosledu, očigledno je da do “mrtvog čvora” ne može da dođe. Na primer ako T1 zahteva A1 pa A2, transakcija T2 neće moći da traži lokote u redosledu A2 pa A1, koji bi doveo do “mrtvog čvora”, jer ovakvo zaključavanje nije po definisanom protokolu. Drugi protokol prevencije zasniva se na dodeljivanju, za ovaj protokol specifične, vremenske oznake transakciji. Ako je transakcija koja zahteva lokot starija (manja vremenska oznaka) od transakcije koja drži lokot na elementu baze, dozvoljava joj se čekanje na dobijanje lokota, u protivnom se prekida. Moguće je primeniti i obrnuti uslov prekidanja.

“Živi” i “mrtvi” lokoti - 4

3. **Detekcija “mrtvog čvora.** Dozvoljava da “mrtvog lokota” dođe, pa se tada neka od transakcija koja ga je izazvala "ubije", njeni efekti na bazu podataka ponište, a ona sama, eventualno, ponovo startuje, nadajući se da tada neće izazvati mrtvi lokot. Za otkrivanje “mrtvih lokota” u sistemu koristi se tzv. **graf čekanja** (Wait-For graf) čiji su čvorovi transakcije T u izvršenju, a grana $T_i - T_j$ postoji ako transakcija T_i zahteva neki objekat koji je transakcija T_j zaključala. Na Slici 11.15 prikazan je jedan graf čekanja. Transakcija T1 čeka na objekat koji je zaključala transakcija T2, ova čeka na objekat koji je zaključala T3, a T3 čeka na objekat koji je zaključala T1. Ispitivanje mrtvih lokota se može vršiti bilo svaki put kada neka transakcija prelazi u stanje "čekaj", bilo periodično, ili se može smatrati da je mrtvi lokot nastao ako transakcija ništa nije uradila u predefinisanim periodu vremena. Očigledno je da se nalaženje “mrtvog lokota” svodi na nalaženje ciklusa u grafu čekanja. Kriterijumi za izbor transakcije koja će se poništiti ("žrtve") mogu biti različiti, na primer, transakcija koja je poslednja startovala ili transakcija koja zahteva najmanje lokota. "Ubijanje" transakcije i poništavanje njenih efekata oslobađa i sve lokote koje je ona postavila.

Dugačke transakcije

Prethodno diskutovane metode upravljanja izvršenjem skupa transakcija pogodne su za poslovne sisteme sa jednostavnim kratkim transakcijama, kada je prihvatljivo da jedna transakcija drži zaključanim neki elemenat baze podataka za neko relativno kratko vreme, reda veličine desetinke sekunde, sekunde ili čak i minuta, zavisno od vrste aplikacija.

Međutim, postoje aplikacije u kojima su transakcije mnogo duže, reda desetine minuta, sati ili čak nekoliko dana. Prikazani mehanizmi zaključavanja u kome bi neka transakcija ovako dugo ekskluzivno zaključala elemente baze podataka koje koristi je, očigledno, neprihvatljivo.

Primeri aplikativnih sistema sa dugim transakcijama:

- Neke transakcije u poslovnim aplikacijama (npr. ukupno stanje računa banke)
- Projektni sistemi (CAE, CAD, CASE)
- Workflow sistemi

Sigurnost baze podataka

Sigurnost baze podataka

Pod sigurnošću baze podataka podrazumeva se zaštita baze od neovlašćenog pristupa.

Osnovni koncepti sigurnosti baze podataka

Sigurnost celokupnog sistema. Jedan od problema sigurnosti je i zaštita celokupnog sistema, odnosno sprečavanje neovlašćenoj osobi da pristupi sistemu bilo sa ciljem da dobije neke informacije ili da ošteti deo ili celu bazu podataka.

Model sigurnosti baze podataka. Osnovni model sigurnosti baze podataka se može definisati preko skupa ***autorizacija*** odnosno skupa četvorki,

Autorizacija: <korisnik, objekat, operacija, uslov>

Osnovni koncepti sigurnosti baze podataka

Podsistem sigurnosti (security subsystem) ili ***podsystem autorizacije (authorization subsystem)*** je deo SUBP-a koji treba da omogući da se model sigurnosti uskladišti i da se operacije nad bazom podataka obavljaju na osnovu definisanog modela.

Diskrecioni mehanizam sigurnosti (Discretionary mechanism) podržava osnovni model u kome se pojedinim korisnicima daju specifična prava pristupa (privilegije) za različite objekte.

Mehanizam ovlašćenja (mandatory mechanism) podržava specifičan model sigurnosti u kome je, s jedne strane svakom objektu baze podataka dodeljen neki stepen “tajnosti”, a sa druge korisnicima dato pravo da pristupe objektima definisanog stepena tajnosti.

Osnovni koncepti sigurnosti baze podataka

Statističke baze podataka se, sa tačke gledišta sigurnosti, posebno analiziraju. Mehanizam sigurnosti statističkih baza mora da onemogući da se iz zbirnih podataka izvuku zaključci o vrednostima pojedinačnih podataka za neki entitet.

Enkripcija podataka. Da bi se osigurali posebno osetljivi podaci u bazi podataka primenjuje se enkripcija podataka, specifičan način kodiranja podataka koji je veoma teško “razbiti”.

Registar i revizija korisnikovih akcija (Audit Trail). Neophodno je voditi poseban log koji se naziva “Registar korisnikovih akcija” i povremeno vršiti pregled (reviziju) ovog registra. Zapis u registru sadrži: (1) izvorni tekst korisnikovog zahteva, (2) identifikator terminala sa kojeg je zahtev postavljen, (3) identifikator korisnika koji je postavio zahtev, (4) datum i vreme, (5) relacije, n-torke i atributi kojima je pristupljeno, (6) novu i staru vrednost

Diskrecioni mehanizam sigurnosti

SQL standard podržava samo diskrecioni mehanizam sigurnosti preko naredbe GRANT za kreiranje autorizacije i naredbe REVOKE kojom se autorizacija ukida. Sintaksa GRANT naredbe je:

```
GRANT < lista privilegija>  
ON <objekat baze>  
TO < lista identifikatora korisnika>  
[WITH GRANT OPTION];
```

SQL ne podržava postavljanje uslova u autorizaciji. Taj nedostatak se za tabele može rešiti prethodnim definisanjem pogleda sa datim uslovom, nad kojim se primenjuje prikazani GRANT iskaz.

Diskrecioni mehanizam sigurnosti

Privilegija data nekom korisniku se može opozvati korišćenjem naredbe REVOKE koja ima sledeću sintaksu:

```
REVOKE [GRANT OPTION FOR] <lista privilegija>  
      ON <objekat baze>  
      FROM <lista identifikatora      korisnika>  
      <opcija>;
```

Primer

Neka je Miloš vlasnik šeme StudentskiS koja ima sledeće tabele:

Student (BI, Ime, Starost, Semestar, Smer)

Predmet (IPred, NazivPred, BrojČas)

Miloš daje privilegiju SELECT i INSERT Ani za tabelu Student, a Aci privilegije SELECT i UPDATE za tabelu Predmet, dozvoljavajući im da te privilegije prenesu i na druge korisnike.

```
GRANT SELECT, INSERT ON Student TO Ana  
WITH GRANT OPTION;  
GRANT SELECT, UPDATE ON Predmet TO Aca;  
WITH GRANT OPTION;
```

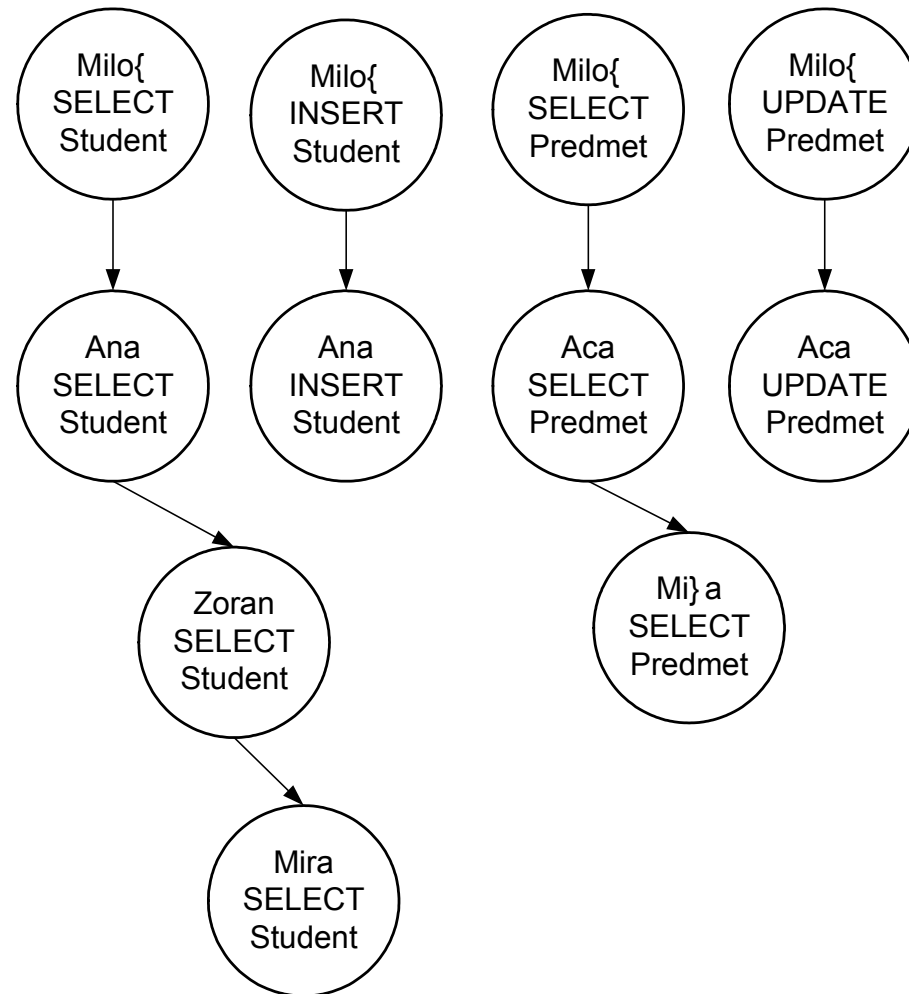
Primer

Ana prenosi privilegiju SELECT za tabele Student na Zorana sa opcijom prenosa na druge, a ovaj prenosi privilegiju SELECT za tabelu Student na Miru. Aca prenosi privilegiju SELECT za tabelu Predmet na Miću.

```
GRANT SELECT Student TO Zoran  
WITH GRANT OPTION;  
GRANT SELECT Student TO Mira;  
GRANT SELECT Predmet TO Mića;
```

Davanje i prenos privilegija prikazaćemo preko tzv. "Grant" dijagrama (grafa). Čvor ovoga grafa predstavlja jednu privilegiju korisnika, a usmerena grana prikazuje prenos privilegija na drugog korisnika.

Primer



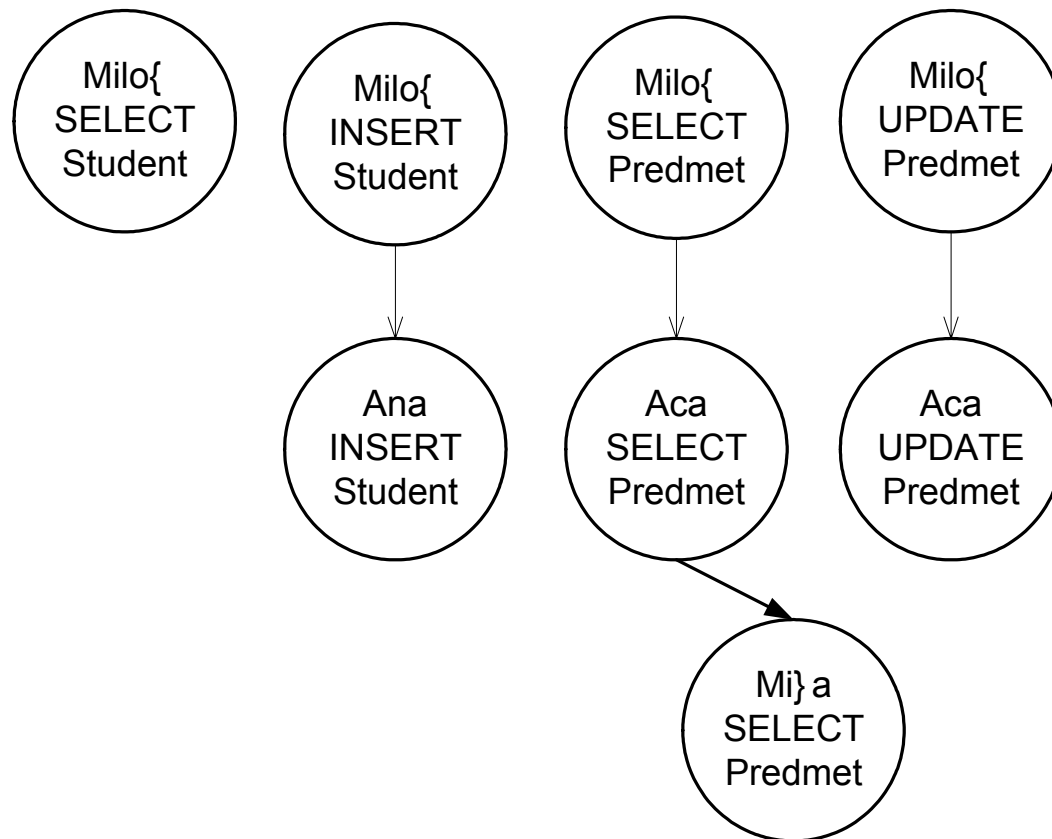
Grant dijagram za prikazani primer

Primer

Na sledecoj slici prikazan je Grant dijagram za isti primer, posle povlačenja privilegija Zoranu i Ani za SELECT Student i Aci za SELECT Predmet preko naredbi:

```
REVOKE GRANT SELECT ON Student  
FROM Ana  
CASCADES;  
REVOKE GRANT SELECT ON Predmet  
FROM Aca  
RESTRICT;
```

Primer



Grant diagram posle povla~enja privilegija

Katalog baze podataka

Katalog baze podataka

Svaki SUBP mora da poseduje svoj ***katalog*** ili ***rečnik podataka***.

Pored termina ***katalog***, koriste se i termini ***rečnik podataka (data dictionary)***, ***direktorijum podataka (data directory)***, ***repozitorijum podataka (data repository)*** ili ***enciklopedija***.

Uobičajena je podela na **aktivne** i **pasivne** rečnike podataka.

SQL okru`enje i katalog baze podataka

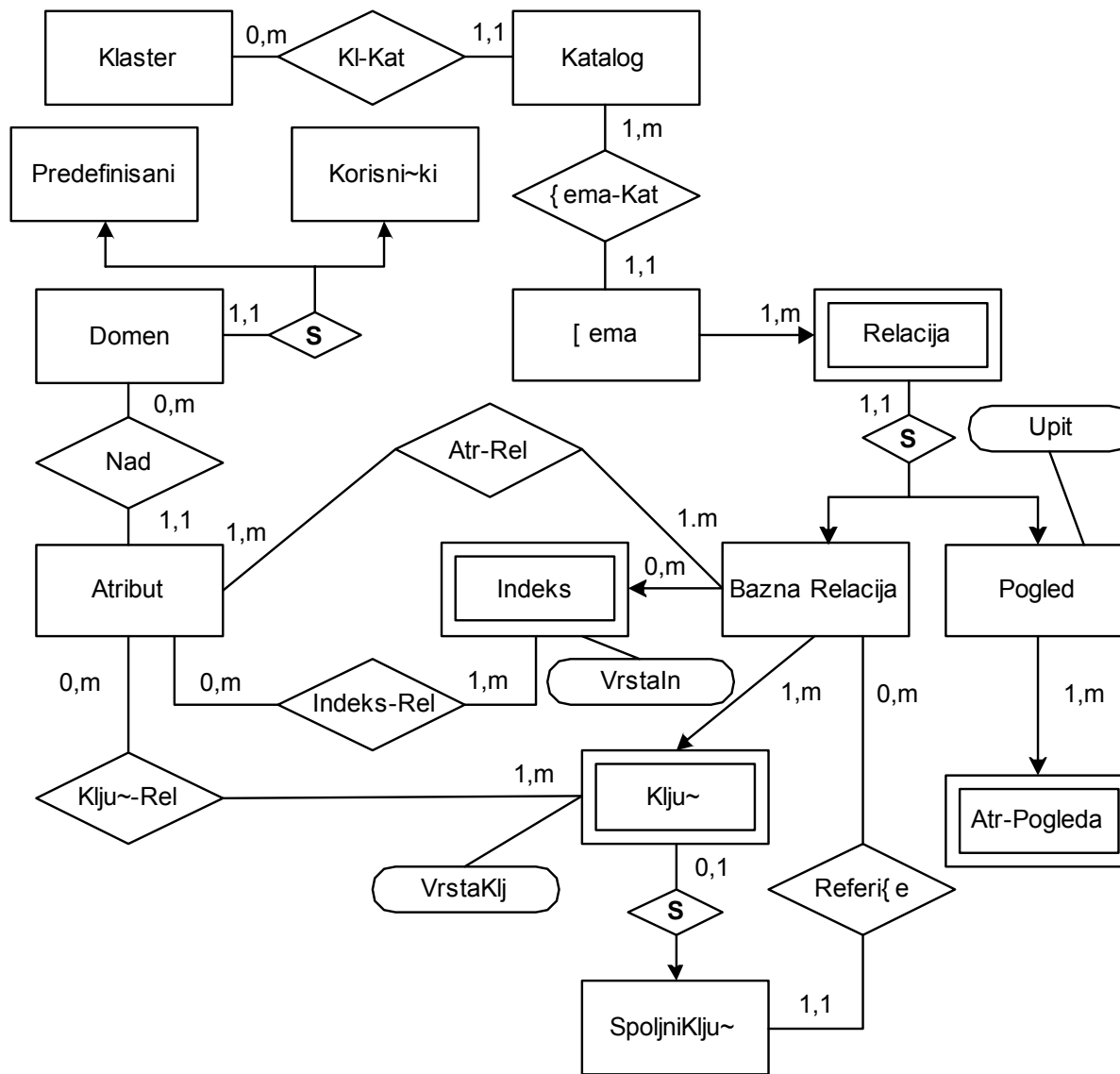
Pod SQL okruženjem podrazumeva se okvir u kome se mogu čuvati podaci i operisati sa njima.

SQL okruženje predstavlja u osnovi neki relacioni SUBP.

SQL okruženje definiše i sledeće:

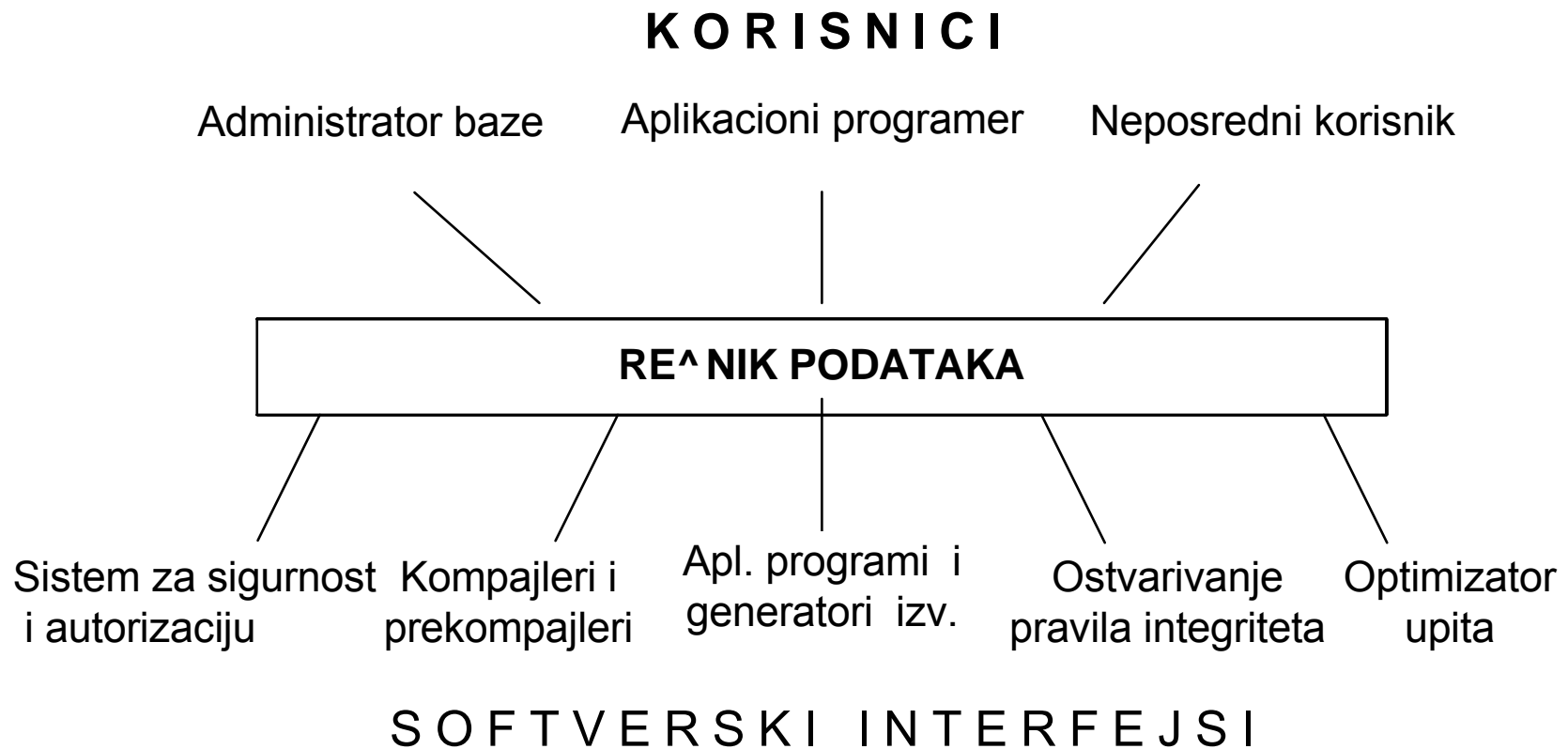
- **Šema** je kolekcija tabela, pogleda, domena, tvrdnji (assertions)
- **Katalog** je kolekcija šema. U katalogu postoji i informaciona šema (INFORMATION_SCHEMA), katalog u užem smislu reči, u kojoj se daju informacije o svim šemama u katalogu.
- **Klaster** je kolekcija kataloga.

SQL okru`enje i katalog baze podataka



Model objekti veze za relacioni katalog

SQL okru`enje i katalog baze podataka



Korisnici i softverski interfejsi za rečnik podataka

Primer rečnika podataka za Oracle SUBP

Dat je primer kreiranja jedne Oracle baze, a zatim slede neki upiti nad katalogom te šeme i njihovi rezultati.

Naziv šeme je MINIB.

Prikazane su relacije o studentima, predmetima, prijavama i profesorima i pogled u kome se daju položeni ispiti za svakog studenta.

```
CREATE TABLE MINIB.STUDENT (  
BRIND          VARCHAR2(6) NOT NULL UNIQUE,  
IME            VARCHAR2(15),  
PREZIME        VARCHAR2(30),  
CONSTRAINT BROJ_IND PRIMARY KEY (BRIND));
```

```
CREATE TABLE MINIB.PREDMET (  
SIFRAPRED      INTEGER NOT NULL UNIQUE,  
NAZIV          VARCHAR2(30),  
SEMESTAR       INTEGER,  
CONSTRAINT KLJUC PRIMARY KEY (SIFRAPRED));
```

Primer rečnika podataka za Oracle SUBP

```
CREATE TABLE MINIB.PRIJAVA (  
BRIND          VARCHAR2(6) ,  
SIFRAPRED      INTEGER,  
DATUM          DATE NOT NULL,  
OCENA          INTEGER NOT NULL,  
CONSTRAINT PR PRIMARY KEY (SIFRAPRED,BRIND) ,  
CONSTRAINT PRED FOREIGN KEY(SIFRAPRED) REFERENCES  
    MINIB.PREDMET(SIFRAPRED) ,  
CONSTRAINT STD FOREIGN KEY(BRIND) REFERENCES  
    MINIB.STUDENT(BRIND) ) ;  
  
CREATE TABLE MINIB.PROFESOR (  
SIFRAPROF      INTEGER NOT NULL,  
IME            VARCHAR2(15) ,  
PREZIME        VARCHAR2(30) ,  
SIFRAPRED      INTEGER,  
CONSTRAINT KLJUCPROF PRIMARY KEY (SIFRAPROF) ,  
CONSTRAINT PREDAJE FOREIGN KEY(SIFRAPRED) REFERENCES 53  
    MINIB.PREDMET(SIFRAPRED) ) ;
```

Primer rečnika podataka za Oracle SUBP

```
CREATE VIEW POLOZIO AS
SELSELECT STUDENT.IME, STUDENT.PREZIME, PREDMET.NAZIV,
          PRIJAVA.DATUM, PRIJAVA.OCENA
FROM      STUDENT, PREDMET, PRIJAVA
WHERE     PRIJAVA.BRIND = STUDENT.BRIND
AND       PRIJAVA.SIFRAPRED = PREDMET.SIFRAPRED;
```

Prikaz svih tabela vlasnika MINIB:

```
SQL>  SELECT *
        FROM  ALL_CATALOG
        WHERE  OWNER = 'MINIB';
```

OWNER	TABLE_NAME	TABLE_TYPE
-----	-----	-----
MINIB	POLOZIO	VIEW
MINIB	PREDMET	TABLE
MINIB	PRIJAVA	TABLE
MINIB	PROFESOR	TABLE
MINIB	STUDENT	TABLE

Primer rečnika podataka za Oracle SUBP

Za objektno-relacione konstrukcije ORACLE katalog poseduje USER_TYPES, USER_OBJECT_TABLES i USER_DEPENDENCIES tabele. Osnovne kolone tabele USER_TYPES su:

- TYPE_NAME – naziv tipa
- ATTRIBUTES – broj atributa u tipu
- METHODS – broj metoda u tipu

Osnovne kolone u tabeli USER_DEPENDENCIES koja pokazuje kako je jedan objektni tip definisan preko drugog su:

- NAME – naziv zavisnog objekta šeme
- TYPE - tip zavisnog objekta (TYPE, TABLE i slično)
- REFERENCED_NAME – naziv nadređenog objekta
- REFERENCED_TYPE – tip nadređenog objekta (TYPE, TABLE i slično)
- DEPENDENCY_TYPE – vrsta zavisnosti ('HARD', 'REF').

Primer rečnika podataka za Oracle SUBP

Rezultat sledećeg upita predstavlja podatke o svim kolonama jedne tabele. Kolone NUM_DISTINCT (broj različitih vrednosti), LOW_VALUE (najmanja vrednost), HIGH_VALUE (najveća vrednost) daju značajne podatke za optimizator upita. Ove kolone se ne ažuriraju direktno sa ažuriranjem odgovarajuće tabele već tek kada se postavi zahtev preko Oracle iskaza

```
ANALYZE TABLE STUDENT  
COMPUTE STATISTICS;
```

Prikaz svih informacija o jednoj tabeli:

```
SELECT COLUMN_NAME, DATA_TYPE, DATA_LENGTH, NUM_DISTINCT,  
       HIGH_VALUE, LOW_VALUE  
FROM USER_TAB_COLUMNS  
WHERE TABLE_NAME = 'STUDENT';
```

COLUMN_NAME	DATA_TYPE	DATA_LENGTH	NUM_DISTINCT	HIGH_VALUE	LOW_VALUE
-----	-----	-----	-----	-----	-----
BRIND	VARCHAR2	6	5	3530312F3935	3133372F3936
IME	VARCHAR2	15	4	564C414441	414E41
PREZIME	VARCHAR2	30	5	56554B4F564943	4A41524943