TRANSAKCIJE

B. Lazarević, Z. Marjanović, N. Aničić, S. Babrogić, *Baze podataka*, FON, Beograd, 2003.

KONCEPT TRANSAKCIJE

```
Start
    Read ( Poruka sa terminala )
    Read ( Stanje na računu )
Write ( Stanje na računu )
Write ( Dnevnik izvršenih transakcija )
    Read ( Stanje ATM uređaja )
Write ( Stanje ATM uređaja )
    Read ( Stanje ekspoziture )
Write ( Stanje ekspoziture )
Write ( Poruka na terminal )
Commit
```

Program za podizanje i ulaganje novca sa ATM uređaja se sastoji od više operacija mora zadovoljiti više zahteva:

- Program se mora izvršavati konkurentno sa drugim programima
- Ako program počne sa izvršavanjem, mora se i završiti
- Rezultati rada programa, moraju biti trajni i dokumentovani
- Program se mora izvršavati, saglasno specifikacijama, i u distribuiranom okruženju

Transakcija je niz operacija nad bazom podataka koji odgovara jednoj logičkoj jedinici posla u realnom sistemu.

IZVRŠAVANJE TRANSAKCIJA

- Pretpostavimo da u multiprogramskom računarskom sistemu postoji skup od n aktivnih transakcija, T={T1 ,…, Tn}.
- Transakcije iz ovog skupa se mogu izvršavati :
  - Serijski
  - Konkurentno (Neserijski) – uporedno izvršavanje

IZVRŠAVANJE TRANSAKCIJA

A = 100, B = 100

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

A = 45, B = 165

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	

A = 40, B = 160

T_1	T_2
read(A) 100	
$A := A - 50$	
write(A) 50	
	read(A) 50
	$temp := A * 0.1$
	$A := A - temp$ 5
	write(A) 45
read(B) 100	
$B := B + 50$	
write(B) 150	
	read(B) 150
	$B := B + temp$ 5
	write(B) 165

A = 45, B = 165

T_1	T_2
read(A) 100	
$A := A - 50$	
	read(A) 100
	$temp := A * 0.1$
	$A := A - temp$ 10
	write(A) 90
	read(B) 100
write(A) 50	
read(B) 100	
$B := B + 50$	
write(B) 150	
	$B := B + temp$ 10
	write(B) 110

A = 90 B = 110

Ovaj scenario izvršavanja neće dati stanje koje se može dobiti serijskim izvršavanjem

IZVRŠAVANJE TRASAKCIJA

- Osnovna Pretpostavka – Bez obzira na način izvršavanja, po završetku kompletne transakcije stanje baze mora biti **konzistetno**.
- Neki neserijski redosled je serijabilan (linearan) ako je ekvivalentan serijskom redosledu.
  Rezultat konkurentnog izvršavanja transakcija mora biti ekvivalentan razultatu koji se dobija serijskim izvođenjem tih istih transakcija.

ACID

Transakcija u izvršenju mora da ima tzv. **ACID** osobine:

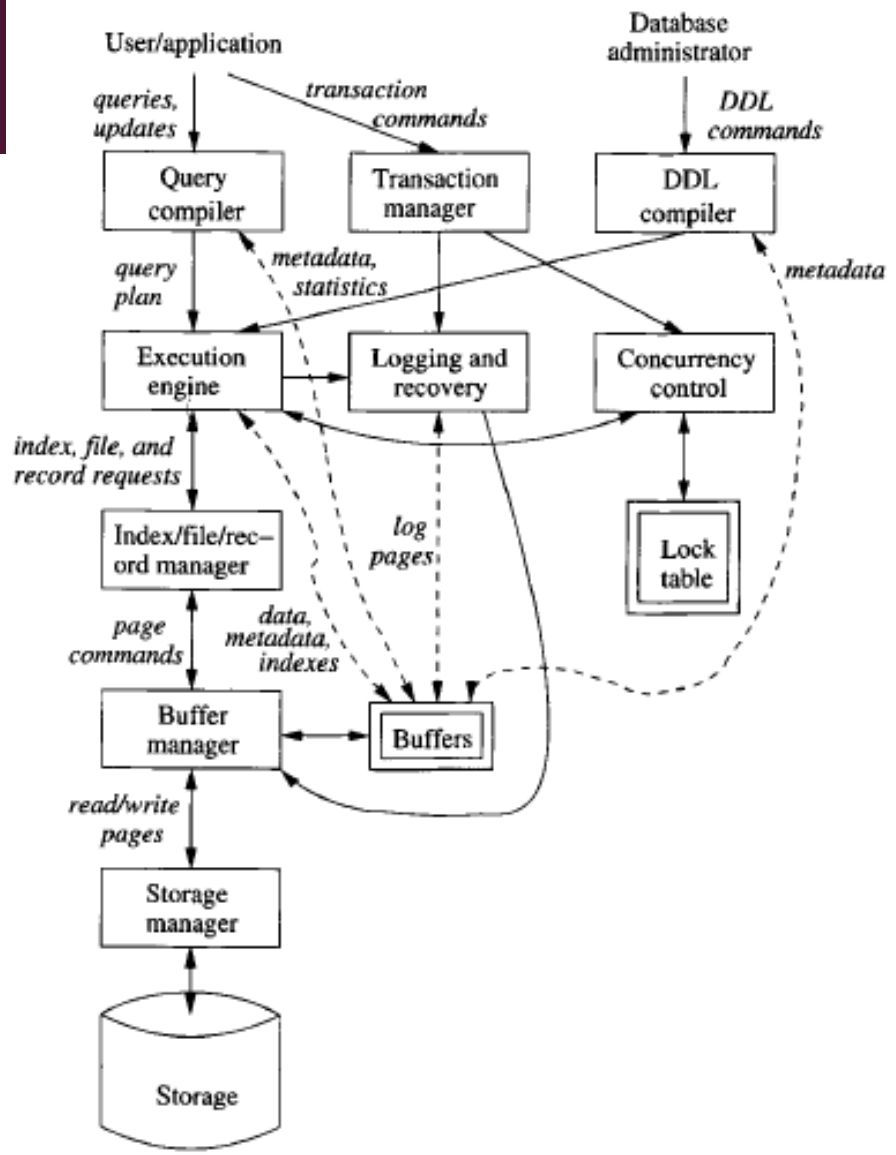
- **Atomičnost (Atomicity)**. Zahteva se da se sve operacije nad bazom podataka predviđene transakcijom uspešno obave ili da se ne prihvati nijedna, tj. ako se bar jedna operacija ne obavi kako je očekivano, onda se poništavaju efekti i svih ostalih.
- **Konzistentnost (Consistency)**. Očigledno je da se transakcija može definisati i kao "jedinica konzistentnosti" baze podataka: pre početka i posle okončanja transakcije stanje baze podataka mora da zadovolji uslove konzistentnosti. Za vreme obavljanja transakcije konzistentnost baze podataka može da bude narušena.
- **Izolacija (Isolation)**. Kada se dve ili više transakcija izvršavaju istovremeno, njihovi efekti moraju biti međusobno izolovani. Drugim rečima efekti koje izazovu transakcije koje se obavljaju istovremeno moraju biti jednaki efektima nekog njihovog serijskog (jedna posle druge) izvršenja.
- **Trajnost (Durability)**. Kada se transakcija završi njeni efekti ne mogu biti izgubljeni, čak i ako se neposredno po njenom okončanju desi neki ozbiljan otkaz sistema.

ACID - PRIMER

- Prenos $50 sa računa A na račun B:
  1. **read(*A*)**
  2. *A := A – 50*
  3. **write(*A*)**
  4. **read(*B*)**
  5. *B := B + 50*
  6. **write(*B*)**
- **Atomska** – ako se transakcija prekine posle koraka 3 i pre koraka 6, sistem mora obezbediti da ažuriranja nisu upisana u bazu, jer će baza preći u nekonzistentno stanje.
- **Konzistentna** – izvršenjem transakcije suma A i B nije promenjena.
- **Izolovana / Nezavisna** – ako se između koraka 3 i 6 dozvoli drugoj transakciji pristup delimično ažuriranoj bazi, ona će videti nekonzistentnu bazu (suma *A + B će biti* manja nego što treba).
- **Trajna** – posle obaveštavanja korisnika da je transakcija izvršena (to jest, prenos $50 izvršen), ažurirana baza mora zadržati stanje uprkos mogućim kvarovima.

OBEZBEĐIVANJE SERIJABILNOSTI

- Transaction Manager - DBMS poseduje komponenta koja upravlja celokupnim izvršenjem transakcija.
- Kontrolu izvršavanja konkurentnih trasakcija vrši Concurency control komponenta, tj. planer (Scheduler).
- Zadatak planera – sprečava neserijabilna rešenja primenom nekog od protokola za utvrđivanje i/ili obezbeđivanje serijabilnosti.
- Protokolima se obezbedjuje serijalizovanost i to NAMETANJEM DISCIPLINE kojom se izbegavaju neserijabilni scenariji izvodjenja.



PROTOKOL ZAKLJUČAVANJA

- Forisarno ostvarivanje serijabilnosti primenom mehanizma zaključavanja.
- Tokom izvršenja transakcije se postavlja lokot (lock) na objekat baze kojem je pristupila.
- Osnovne vrste zaključavanja podataka

	S	X
S	true	false
X	false	false

  - **Ekskluzivno zaključavanje** (**exclusive** ili write lock) – ni jedna druga transakcija ne može postaviti novi lokot, bilo koje vrste.
  - **Deljivo zaključavanje** (**shared** (S) ili read lock) – druge transakcije mogu dobiti deljivi lock nad istim objektom, ali ne mogu dobiti ekskluzivni.
- Zahtevi za zaključavanjem se prosleđuju concurrency-control manageru, a transakcija može nastaviti sa izvršavanjem samo ako je lokot odobren.

PROTOKOL ZAKLJUČAVANJA

- 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.

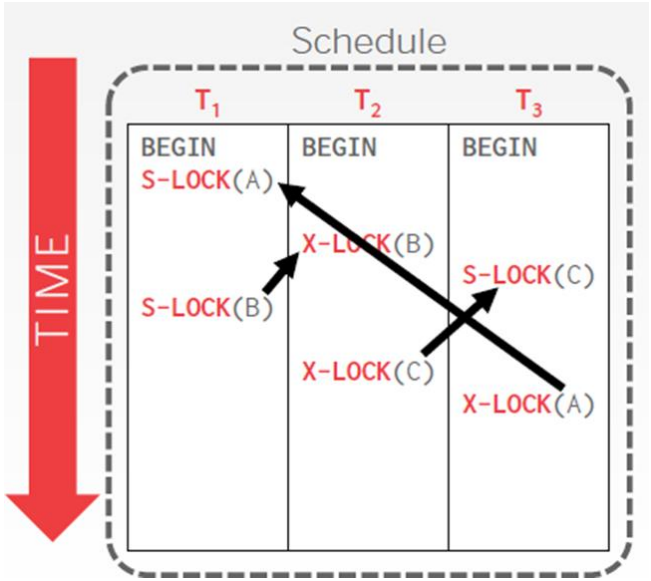
DVOFAZNI PROTOKOL ZAKLJUČAVANJA

- Transakcija prolazi kroz dve faze izvršavanja:
  - Faza širenja – kada broj lokota raste, lokoti se zahtevaju, ali se ne otpuštaju.
  - Faza skupljanja – lokoti se oslobađaju, a nije dozvoljeno postavljanje novih.
- Transakcije koje poštuju dvofazni protokol su uvek serijabilne.

Transakcija A	vreme	Transakcija B
read(X); {postavlja se S lokot na objekat X}	t1	
X := X – N;	t2	
write(X); {postavlja se X lokot za objekat X}	t3	
	t4	**read(X);** {zahteva se X lokot za objekat X}
		čekanje
Rollback; {oslobađa se X lokot sa objekta X}		čekanje
	t5	**read(X);** {postavlja se S lokot na objekat X}
		X := X + M;
		write (X); {postavlja se X lokot za objekat X}
		Commit; {oslobađa se X lokot sa objekta X}

DEADLOCK – MRTVI ČVOR

- Transakcije čekaju jedna na drugu.
- Da bi deadlock bio razrešen jedna od ove dve transakcije mora biti poništena, a lokoti oslobođeni.
- Ukoliko concurency-control manager nije dobro projektovan dolazi do izgladnjivanja (starvation)

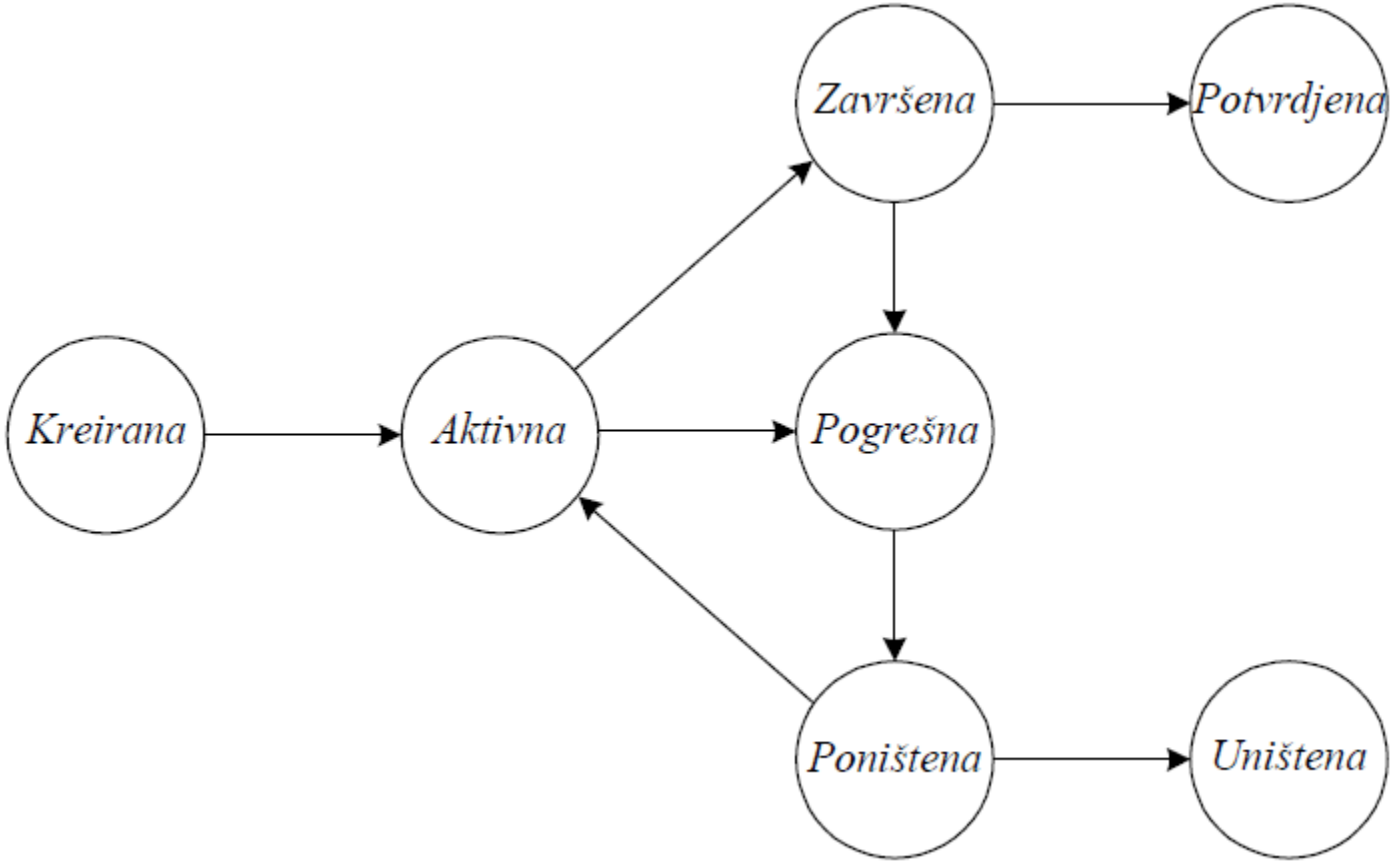


Transakcija A	vreme	Transakcija B
read(X); {postavlja se S lokot na objekat X}	t1	
X := X – N;	t2	
		read(X); {postavlja se S lokot na objekat X}
	t3	X := X + M;
write(X); {zahteva se X lokot za objekat X}	t4	
čekanje	t5	
čekanje		
čekanje	t6	**write (X);** {zahteva se X lokot za objekat X}
čekanje		
čekanje		čekanje
čekanje		čekanje

STANJA TRANSAKCIJE

- **Kreirana** – spremna za aktiviranje
- **Aktivna** – počne da se izvršava
- **Završena** – pošto se izvrši zadnja instrukcija.
- **Pogrešna** – ako se inicirani upisi u bazu ne završe usled kvara sistema. U ovo stanje, transakcija može preći i iz aktivnog stanja, ukoliko u toku izvršavanja transakcije dođe do kvara sistema, bilo hardverskog, bilo softverskog.
- **Poništena** – Pošto se nije uspešno završila, transakciju je potrebno prevesti u stanje Poništena, i vratiti bazu u prethodno konzistentno stanje. Posle restauracije baze, transakciju treba ponovo startovati, ako je hardverski ili softverski kvar, zbog koga se transakcija nije mogla završiti, u potpunosti otklonjen. Ukoliko kvar ne može biti u potpunosti otklonjen, ponovno startovanje transakcije nema smisla, već se transakcija prevodi u stanje Uništena.
- **Uništena** – Tipičan slučaj je prisustvo logičke greške u transakciji. Takva greška može biti ispravljena samo ponovnim pisanjem programa transakcije.
- **Potvrđena** – svi upisi, inicirani u toku izvršenja transakcije, u bazu završeni i očuvana konzistentnost baze.

STANJA TRANSAKCIJE

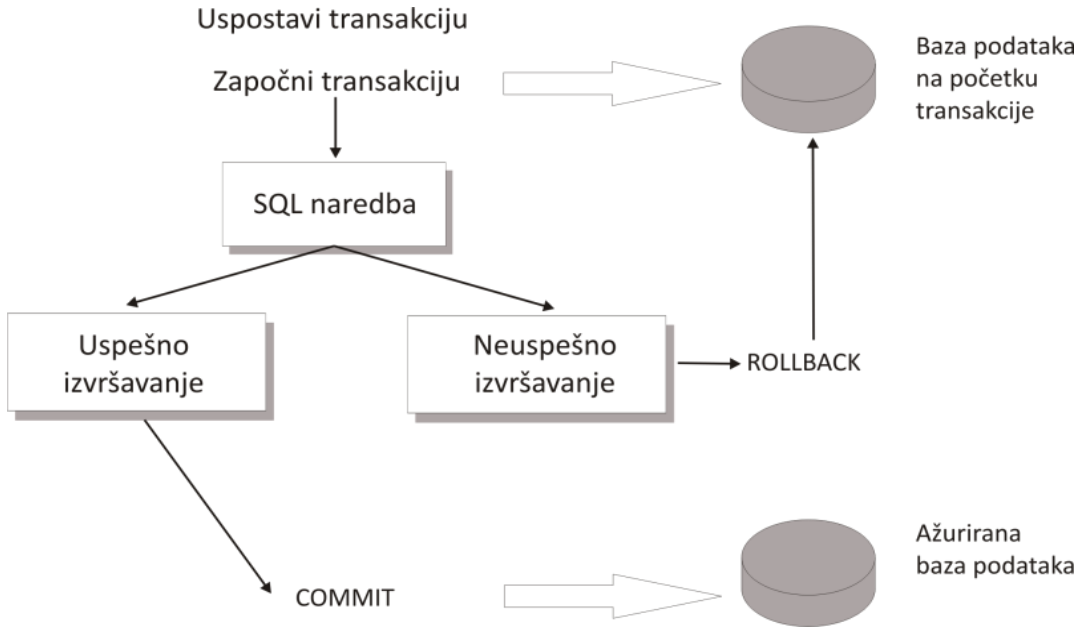


SQL PODRŠKA REALIZACIJI TRANSAKCIJA

- SQL obezbeđuje sledeće naredbe za upravljanje transakcijama:
  - SET TRANSACTION,
  - START TRANSACTION (BEGIN TRANSACTION – T-SQL),
  - SAVEPOINT,
  - ROLLBACK i
  - COMMIT.
- **BEGIN TRANSACTION -** deklariše eksplicitnu transakciju.
- **SET TRANSACTION -** kontroliše karakteristike promene podataka, najpre read/write karakteristike i **nivo izolacije** (izdvajanja) transakcija.
- **COMMIT** - Naredba COMMIT eksplicitno završava otvorenu transakciju i čini promene stalnim u bazi podataka.
- **ROLLBACK** - Naredba ROLLBACK poništava transakcije do njihovog početka ili do prethodno definisane SAVEPOINT tačke.
- **SAVEPOINT -** Ova naredba prekida transakciju u logičkim tačkama prekida. U okviru jedne transakcije može biti specificirano više tačaka čuvanja. Glavna korist od naredbe SAVEPOINT jeste ta da efekti komandi unutar transakcije mogu da budu parcijalno povraćeni do označene tačke čuvanja korišćenjem naredbe ROLLBACK

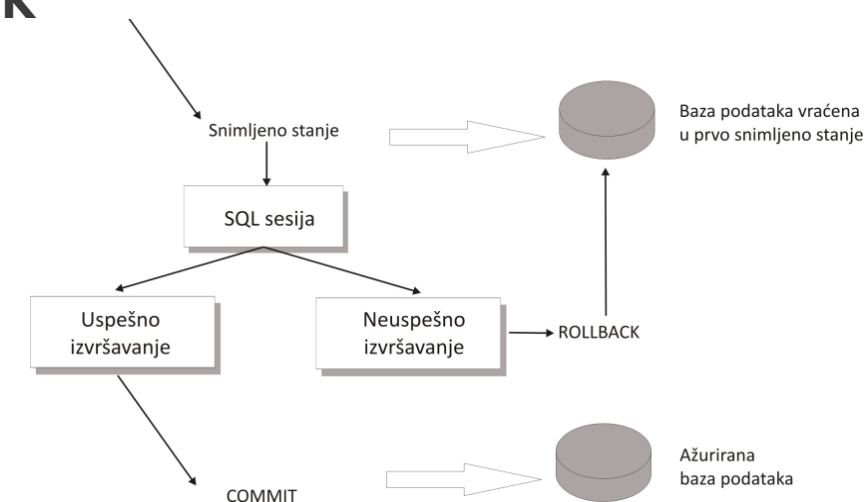
POTVRDA I PONIŠTAVANJE

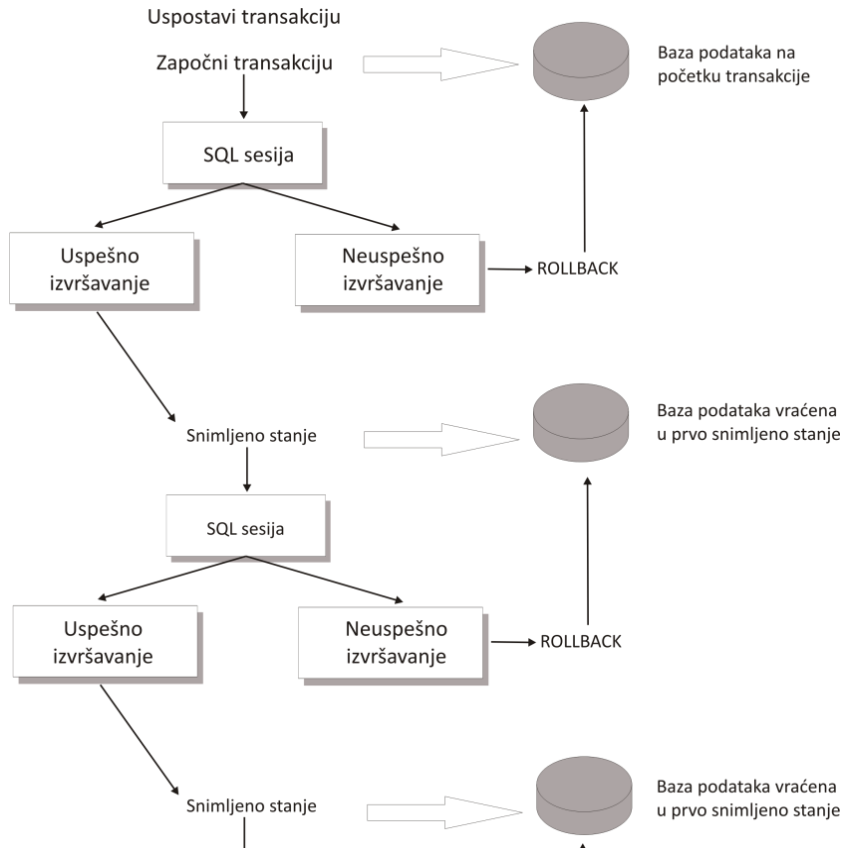
- Svaka transakcija koja je eksplicitno deklarisana može da bude učinjena stalnom samo izvršavanjem naredbe COMMIT.
- Slično, bilo koja transakcija koja je neuspešna ili je potrebno da bude odbačena mora da bude eksplicitno poništena naredbom ROLLBACK.



EKSPLICITNA I IMPLICITNA KONTROLA TRANSAKCIJE

```
BEGIN TRANSACTION t1
INSERT INTO magacin
VALUES(100, 'proizvod1', 100.50, 40)
IF @@ERROR <> 0 ROLLBACK
save transaction tr1
INSERT INTO magacin VALUES(101, 'proizvod2', 110.50, 50);
IF @@ERROR <> 0 ROLLBACK TRANSACTION tr1
save transaction tr2
INSERT INTO magacin VALUES(102, 'proizvod3', 120.50, 50)
IF @@ERROR <> 0 ROLLBACK
TRANSACTION tr2
COMMIT TRANSACTION t1
```





KONKURENTNOST - PROBLEMI

Nekontrolisano konkurentno izvršenje transakcija može uzrokovati:

- Gubljenje rezultata ažuriranja – pisanje preko ažuriranih podataka
- Problem nekorektne analize podataka – upotreba pre ažuriranja
- Problem nepotvrđenih promena (čitanja)

SQL I NIVOI IZOLACIJE

- Serijabilnost zahteva najviši nivo međusobne izolovanosti transakcija, što usporava izvršavanje i upotrebljivost paralelizma.
- DBMS-ovi dozvoljavaju i niže nivoe izolovanosti.
- Nivoom izolacije se definiše politika postavljanja lokota.
- SQL satndard predviđa:
  - SERIALIZABLE
  - REPEATABLE READ
  - READ COMMITED
  - READ UNCOMMITED
- Nivo izoalcije je karakteristika transakcije i postavlja se

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

SQL I NIVOI IZOLACIJE

- Različiti nivoi izolacije dozvoljavaju/ne dozvoljavaju određene načine narušavanja integiteta baze ili izlaza-
- ***Prljavo citanje (dirty read).***
  Ako transakcija T1 ažurira neki podatak u bazi, zatim transakcija T2 procita taj podatak, a nakon toga se transakcija T1 završi sa ROLLBACK, tada je ocigledno transakcija T2 procitala nepotvrđen i nepostojeći podatak, odnosno izvršila "prljavo citanje".
- ***Neponovljivo citanje (Nonrepeatable read).***
  U jednoj transakciji, ponovno citanje istih podataka mora dati isti rezultat. Ako transakcija T1 procita neki podatak baze, zatim ga transakcija T2 promeni, ponovno citanje istog podatka u transakciji T1 nece dati isti rezultat.
- ***Fantomsko citanje (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.

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

SIGURNOST BAZA PODATAKA

B. Lazarević, Z. Marjanović, N. Aničić, S. Babrogić, *Baze podataka*, FON, Beograd, 2003.
D. Stefanović, SQL i programiranje u relacionim bazama podataka, PMF, Kragujevac, 2009.

SIGURNOST BAZE PODATAKA

- Dva aspekta sigurnosti:
  - Sigurnost celog sistema - sprečavanje neovlašćenih lica da pristupe sistemu
    uključuje proces logovanja korisnika, koliko je prostora na disku dodeljeno svakom korisniku i koje akcije može svaki korisnik da izvršava
  - Sigurnost baze podataka – zaštita baze od neovlašćenog pristupa
    uključuje upravljanje i dodeljivanje dozvola za korisnike za različite objekte u bazi podataka
- Ciljevi:
  - SUBP mora da garantuje da neautorizovani korisnici ne mogu da vide ili promene podatke.
  - U isto vreme, autorizovanim korisnicima treba omogućiti da pristupaju bilo kojoj informaciji koja bi trebala da im bude na raspolaganju.
- U skladu sa ovim ciljevima SQL definiše model bezbednosti koji omogućava da se odredi koji korisnici mogu da pristupaju određenim podacima i šta mogu da rade sa njima

SIGURNOST BAZE PODATAKA

Osnovni model sigurnosti baze podataka se može definisati preko skupa autorizacija, koje predstavljaju uređene četvorke

<korisnik, objekat, operacija, uslov>

- Podsistem sigurnosti ili podsistem autorizacije (security/autorization subsystem)
  Komponenta DBMS-a koja omogućava skladištenje modela sigurnosti, kao i obavljanje operacija nad bazom u skaldu sa definisanim modelom.

SQL

- SQL definiše model bezbednosti kojim definiše koji korisnici mogu da pristupaju određenim podacima i šta mogu da rade sa njima.
- Centralni objekat modela - autorizacioni identifikator

  Autorizacioni identifikator korisnika - objekat u SQL okruženju, predstavlja korisnika ili grupu korisnika kojima su dozvoljene određene privilegije nad objektima i podacima u SQL okruženju.

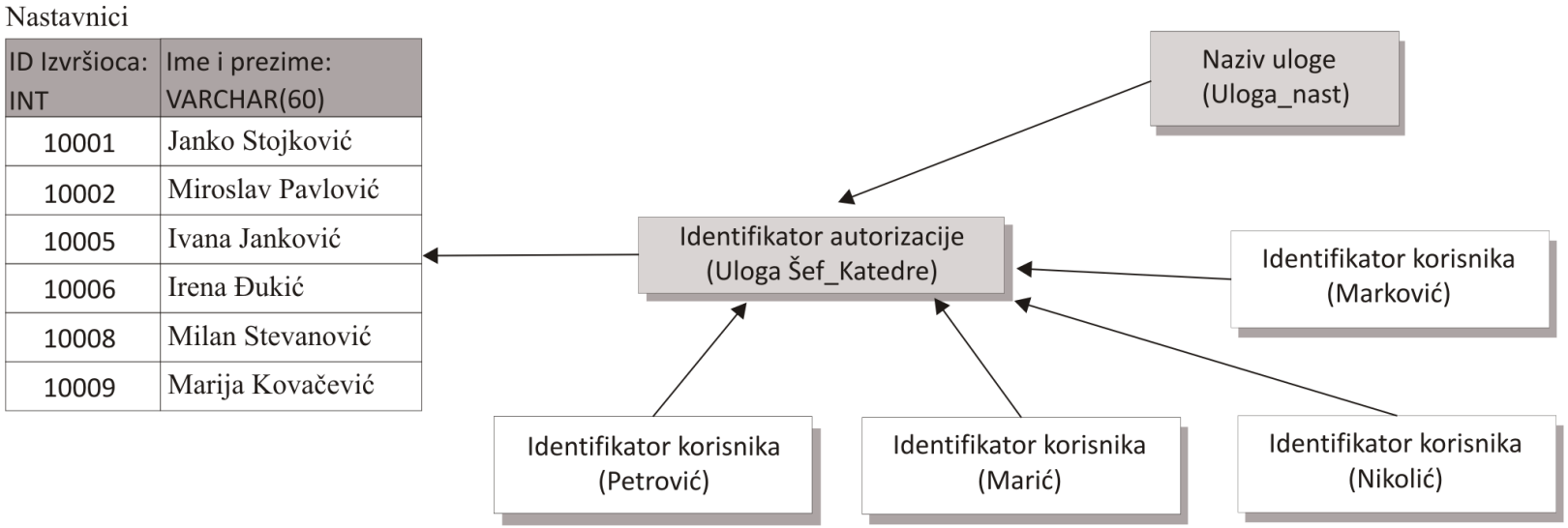
  Privilegije su dodeljene autorizacionim identifikatorima u šemi objekata.

  Tip privilegije koji je dodeljen određuje tip pristupa.
- Dva tipa autorizacionih identifikatora:
  identifikatori korisnika (user) i
  imena uloga (role)

Napomena: Korisnik je poseban bezbednosni nalog koji može da predstavlja pojedinca, aplikaciju ili sistemski servis.

ULOGA

- Definiše skup privilegija koje mogu da se dodele korisniku ili drugoj ulozi.
- Ako je ulozi dodeljen pristup šemi objekta, onda svi korisnici i uloge kojima je bila dodeljena specificirana uloga imaju pristup istom objektu kao i data uloga.



SQL SESIJA

- SQL sesija je veza između neke vrste klijentske aplikacije i baze podataka.
- Svaka SQL sesija je povezana sa korisničkim identifikatorom (korisnikom i imenom uloge (ulogom))
- Sesija obezbeđuje kontekst u kome autorizacioni identifikator izvršava SQL naredbe tokom jedne konekcije.
- SQL sesija održava tokom ove konekcije vezu sa parom korisnik/uloga.

PRIVILEGIJE

Privilegije se mogu dodeljivati nad različitim objektima baze – tabela, view, stored procedure, ...

Neke privilegije nad tabelama

- SELECT
- INSERT – može se primeniti i na atribute
- DELETE
- UPDATE – može se primeniti i na atribute

KREIRANJE IDENTIFIKATORA I DODELA PRIVILEGIJA

```
LOGIN SavicMilos
    WITH PASSWORD = 'Ailo3ksz';
USE Studije;
```

```
CREATE USER Savic FOR LOGIN SavicMilos
    WITH DEFAULT_SCHEMA = NastOdr;
CREATE ROLE SefKatedreAUTHORIZATION
Savic;
```

Naredba GRANT dodeljuje privilegije korisnicima i ulogama, koje im omogućavaju da pristupe i koriste objekte baze podataka (to jest, privilegije objekata).

```
GRANT INSERT, UPDATE, DELETE ON NastOdr.Nastavnici TO SefKatedre;
GRANT SELECT ON Nastavnici TO nastavnici WITH GRANT OPTION
GRANT SELECT ON Nastavnici TO nastavnici
GRANT SELECT ON Nastavnici TO Ana AS nastavnici
```

REVOKE

- Naredba *REVOKE* uzima dva glavna oblika (forme). Prvi oblik naredbe oduzima (odstranjuje) specificirane dozvole naredbi od korisnika, grupa ili uloga. Drugi oblik naredbe oduzima dozvole pristupa specificiranim objektima baze podataka ili resursima.

REVOKE GRANT OPTION FOR

SELECT, INSERT, UPDATE, DELETE ON Prijave TO Milan CASCADE

REVOKE ALL PRIVILEGES ON TABLE Nastavnici FROM Dejan

REVOKE SELECT(Indeks, Upisan, Mesto) ON Studenti FROM Marko

DINAMIČKI SQL I SQL INJECTION

SQLi

DINAMIČKI SQL

```
DECLARE @g INT
SET @g = 2001

SELECT *
from studenti
where upisan = @g

EXECUTE sp_executesql N'SELECT *
                        from studenti
                        where upisan = @god',
                        N'@god int',@god = 2001;

declare @tekst nvarchar(100)
set @tekst = N'SELECT * FROM studenti WHERE upisan=2001'
EXECUTE sp_executesql @tekst
```

SQLI

SQLi je vrsta napada na aplikacije u kojoj napadač preoblikuje SQL upite kako bi izvukao podatke iz baze ili izmenio sadržaj baze.

PRIMER

```
select * into copystudenti from Studenti
select * from copystudenti

declare @komanda nvarchar(100)
declare @param nvarchar(100)
set @komanda = N'SELECT * FROM studenti WHERE upisan='
set @param = N'2001; drop table copystudenti'

set @komanda = @komanda + @param
EXECUTE sp_executesql @komanda

select * from copystudenti
```

ZAŠTITA

- Parametrizacija upita
- Poseban tretman specialnih karaktera (insistiranje na duplim i sl.)
- Provera paterna
- Ograničavanje dozvolama