# Konfigurisanje i adminstracija baza podataka

## Transakcije

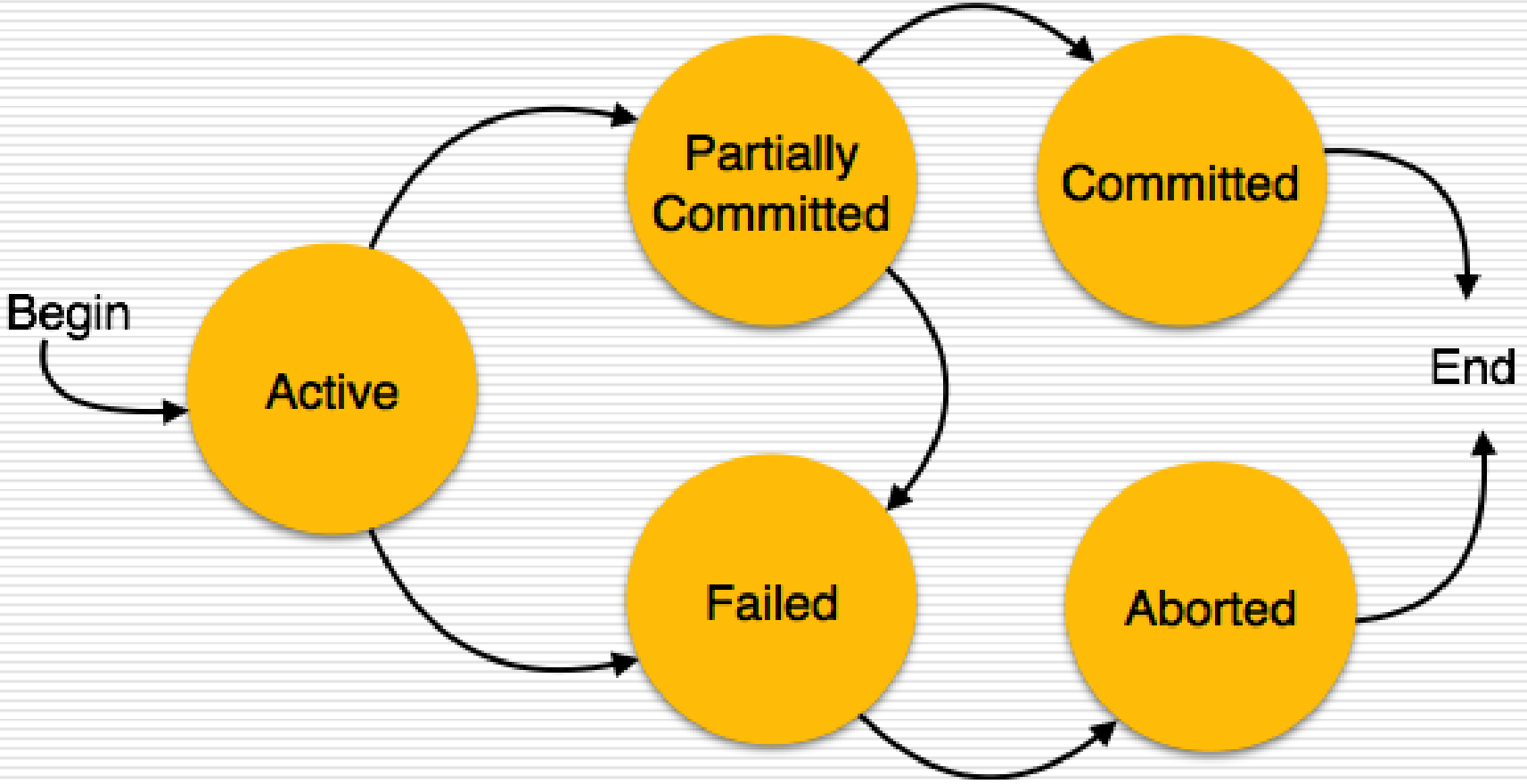
# Transakcije

- Transakcija je jedinica rada koja je izvršena nad bazom podataka.
- Transakcije mogu biti jedinice ili sekvence rada izvršene logičkim ledom, bilo ručno od strane korisnika ili automatski nekim programom koji upravlja bazom podataka.
- Transakcija je propagiranje jedne ili više promena nad bazom podataka.
- Na primer: Kada vršimo insert, update ili delete vrednosti iz baze, mi vršimo transakciju nad tabelom.
- Veoma je važno upravljati transakcijama kako bismo obezbedili integritet podataka i da bismo mogli da upravljamo greškama.
- Praktično ćemo više SQL upita grupisati i izvršiti ih kao jednu transakciju.

# Osobine transakcija: ACID

- Transakcije imaju 4 standardne osobine, obično opisane akronimom ACID.
- **Atomicity** (atomarnost) – obezbeđuje da su sve operacije unutar jedinice rada (transakcije) izvršene uspešno.
- U suprotnom transakcija se prekida u tački u kojoj je došlo do greske  i prethodno izvršene operacije se vraćaju na prethodno stanje, odnosno izvrši se rollback.
- Što znači da će transakcija ili biti izvršena u potpunosti ili ponistiti sva njena dejstva.
- Ne sme postojati stanje baze u kome je transakcija samo parcijalno izvršena.

# Osobine transakcija: ACID



# Osobine transakcija: ACID

- **Consistency** (konzistentnost) – obezbeđuje da će baza pravilno promeniti stanje nakon uspešno izvršene transakcije.
- Nijedna transakcija ne sme ostaviti negativan efekat na podatke koji se nalaze u bazi podataka.
- Ako je baza bila u konzistetnom stanju pre izvršene transakcije, mora ostati u konzistentnom stanju i nakon izvršene transakcije.

# Osobine transakcija: ACID

- **Isolation** (izolacija) – obezbeđuje da nijedna transakcija neće ugroziti izvrsavanje neke druge transakcije.
- U sistemu baza podataka, gde se jedna ili više transakcija izvršava redom, ili paralelno, izolacija omogućava da svaka transakcija bude izvršena kao da je jedina transakcija u sistemu.

# Osobine transakcija: ACID

- **Durability** (trajnost) – obezbeđuje da će rezultat izvršene transakcije postojati i u slučaju otkaza sistema.
- Baza podataka bi trebalo da bude trajna toliko da zadrži i poslednje napravljene izmene čak i u slučaju otkaza sistema ili ponovnog pokretanja.
- Ako transakcija promeni deo podataka u bazi i ako komituje, baza će sačuvati promene.
- U slučaju da transakcija komituje, ali da baza otkaže pre nego što se podaci upišu na disk, podaci će biti ažurirani čim se sistem vrati u operativno stanje.

# Kontrola transakcija

- Postoji nekoliko komandi za upravljanje transakcijama:
- COMMIT – da sačuvamo promene.
- ROLLBACK – da vratimo na prethodno stanje.
- SAVEPOINT – tačka u transakciji na koju možemo da se vratimo bez potrebe za rollback-om cele transakcije
- SET TRANSACTION – postavljamo ime transakciji.

# Kontrola transakcija

- Komande za kontrolu transakcija se koriste isključivo za naredbe INSERT, UPDATE i DELETE.
- Ne mogu se koristiti prilikom kreiranja ili brisanja tabele, zato što se ove operacije automatski komituju u bazu podataka.

# Commit

- **COMMIT** – komanda koja se koristi da sačuvamo izmene koje je napravila transakcija.
- Commit komanda čuva sve transakcije u bazi, od poslednje Commit ili Rollback naredbe.
- Sintaksa za Commit naredbu:
- COMMIT;

# Commit

| ID | Nam e | Age | Salar y |
|---|---|---|---|
| 1 | Milan | 25 | 3000 |
| 2 | Goran | 23 | 1450 |
| 3 | Marko | 25 | 2000 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

> DELETE FROM CUSTOMERS WHERE AGE = 25;
> COMMIT;

| ID | Name | Age | Salary |
|---|---|---|---|
| 2 | Goran | 23 | 1450 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

# Rollback

- **ROLLBACK** – je komanda kojom poništavamo akcije transakcije koje još uvek nisu sačuvane u bazi podataka.
- Rollback možemo koristiti da poništimo samo one transakcije koje su izvršene nakon poslednjeg commit-a ili rollback-a.
- Sintaksa za Rollback naredbu:
- ROLLBACK;

# Rollback

| ID | Name | Age | Salary |
|---|---|---|---|
| 1 | Milan | 25 | 3000 |
| 2 | Goran | 23 | 1450 |
| 3 | Marko | 25 | 2000 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

| ID | Name | Age | Salary |
|---|---|---|---|
| 1 | Milan | 25 | 3000 |
| 2 | Goran | 23 | 1450 |
| 3 | Marko | 25 | 2000 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

```
> DELETE FROM CUSTOMERS WHERE AGE = 25;
> ROLLBACK;
```

# Savepoint

- **SAVEPOINT** – je tačka u transakciji, na koju možemo da se vratimo bez potrebe za rollback-om cele transakcije.
- Sintaksa za savepoint:
- SAVEPOINT SAVEPOINT_NAME;
- Ova komanda služi samo za pravljenje ,,sigurne tačke“. Rollback-om ćemo poništiti transakciju.
- Sintaksa za rollback savepoint:
- ROLLBACK TO SAVEPOINT_NAME;

# Savepoint

| ID | Name | Age | Salary |
|---|---|---|---|
| 1 | Milan | 25 | 3000 |
| 2 | Goran | 23 | 1450 |
| 3 | Marko | 25 | 2000 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

- SAVEPOINT SP1;

Savepoint created.

- DELETE FROM CUSTOMERS WHERE ID = 1;

1 row deleted.

- SAVEPOINT SP2;

Savepoint created.

- DELETE FROM CUSTOMERS WHERE ID = 2;

1 row deleted.

- SAVEPOINT SP3;

Savepoint created.

- DELETE FROM CUSTOMERS WHERE ID = 3;

1 row deleted.

# Savepoint

- ROLLBACK TO SP2;

Rollback complete.

| ID | Name | Age | Salary |
|---|---|---|---|
| 2 | Goran | 23 | 1450 |
| 3 | Marko | 25 | 2000 |
| 4 | Petar | 27 | 5000 |
| 5 | Jovan | 29 | 3150 |

# Release savepoint

- **RELEASE SAVEPOINT** – koristimo da obrišemo savepoint koji smo prethodno napravili.
- Sintaksa za release savepoint:
- RELEASE SAVEPOINT SAVEPOINT_NAME;
- Onog trenutka kad obrišete savepoint, više ne možete da koristite rollback nad transakcijama izvršenim od tog savepoint-a.

# Set transaction

- **SET TRANSACTION** – komanda kojom započinjemo transakciju nad bazom podataka.
- Ovom komandom spefikujemo osobine određene transakcije.
- Na primer, možemo da kažemo da je transakcija read-only, ili da je transakcija read-write
- Sintaksa za set transaction:
- SET TRANSACTION [ READ ONLY | READ WRITE];

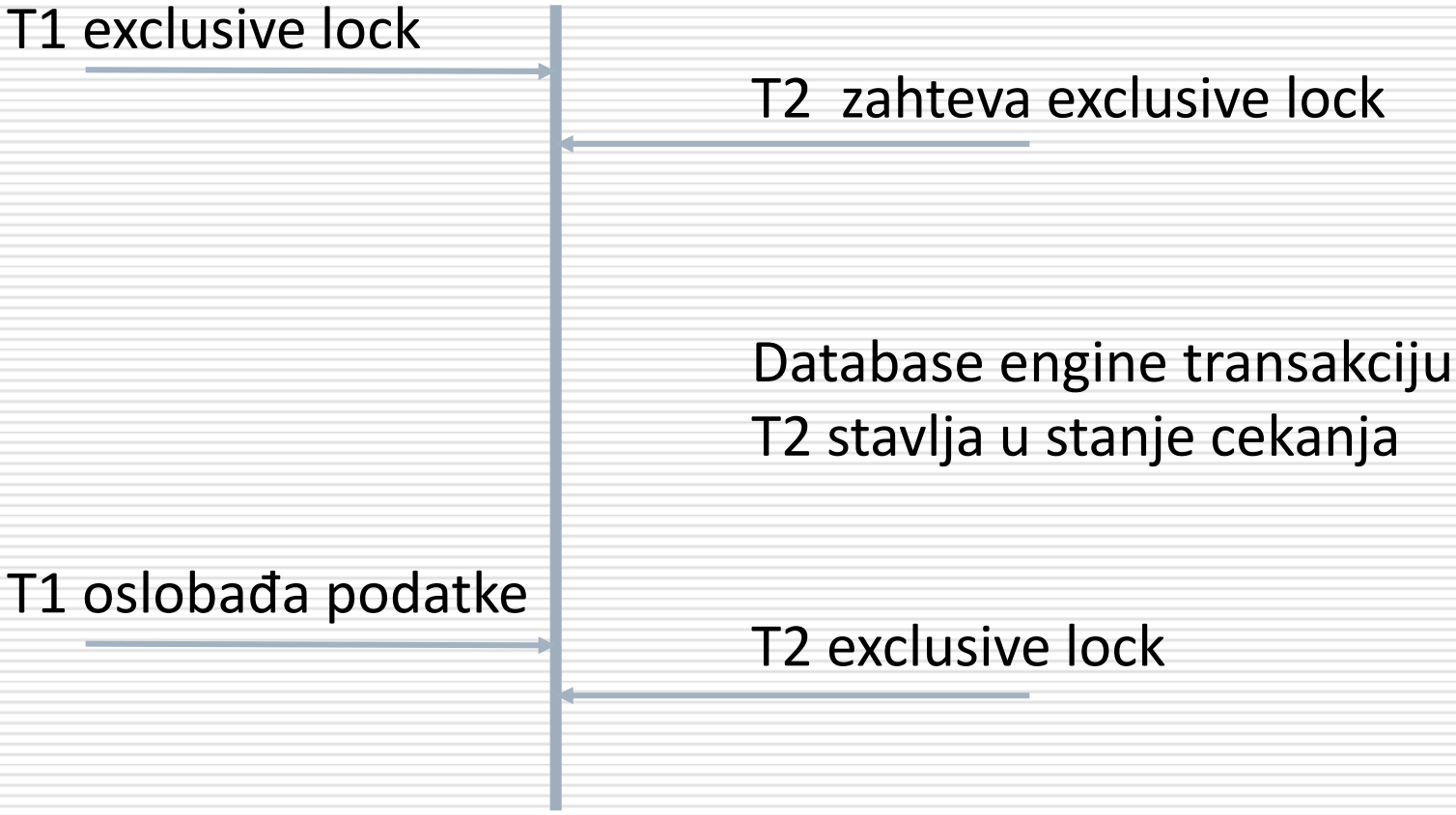
# Zaključavanje baze podataka (locking in)

- **Zaključavanje** je mehanizam koji se koristi od strane SUBP-a da sinhronizuje pristup više korisnika istim podacima u isto vreme.
- Pre nego što transakcija počne da menja stanje podataka, prilikom čitanja ili ažuriranja, mora da se zaštiti od efekata druge transakcije koja pokušava da menja podatke istovremeno.
- Transakcija ovo realizuje slanjem zahteva za zaključavanjem tog dela podataka.
- Zaključavanje ima različite režime, kao što su **shared** (deljen) ili **exclusive** (isključivo za sebe).
- Zaključavanje definiše nivo zavisnosti koji transakcija ima nad određenim podacima.

# Zaključavanje baze podataka

- Transakciji se ne može dodeliti pravo zaključavanja koje bi prouzrokovalo konflikt sa modom zaključavanja koji je druga transakcija već dobila.
- Ako transakcija zahteva mod zaključavanja koji će prouzrokovati konflikt sa zaključavanjem koje je već dodeljeno nad istim podacima, SUBP će zahtevanu transakciju staviti u status cekanja sve dok se ne oslobode podaci.
- Primer: T1 ima **exclusive** pravo nad podacima. T2 traži **exclusive** pravo nad istim podacima. Database engine transakciju T2 stavlja u stanje cekanja. T1 oslobađa podatke. Database engine dodeljuje **exclusive** pravo nad podacima za transakciju T2.

# Zaključavanje baze podataka



# Zaključavanje baze podataka

- Kada transakcija ažurira podatke, ona ih drži zaključanim štiteći ih od promena sve dok se transakcija ne zavrsi.
- Vreme koliko transakcija drži podatke zaključanim zavisi od podešavanja kojima je definisan nivo izolacije transakcije.
- Svako zaključavanje se oslobađa prilikom završetka transakcije (bilo u slučaju commit-a ili rollback-a).

# Zaključavanje baze podataka

- Aplikacije obično ne zahtevaju zaključavnje podataka. Zaključavanjima se upravlja interno, komponentom Database Engine koji se zove **lock manager**.
- Kada istanca Database Engine procesira SQL upit, Database Engine Query procesor određuje kojim reursima može da se pristupa.

# Zaključavanje baze podataka

- Query procesor odlučuje koji tipovi zaključavanja su neophodni da bi se zaštitio svaki resurs.
- Odlučivanje je zasnovano na tipu pristupa i nivou izolacije.
- Query procesor dalje zahteva određeno zaključavanje od lock manager-a.
- Lock manager odobrava zaključavanje, samo ako ne postoji mogućnost da dodje do konflikta zbog neke druge transakcije.

# Granularnost i hijerarhija zaključavanja

- Database Engine ima multigranularno zaključavanje koje omogućava različitim tipovima resursa da budu zaključani određenom transakcijom.
- Da bi smanjili troškove zaključavnaja, Database Engine zaključava resurse automatski na nivou koji je prikladan tom zadatku.
- Zaključavajući manju granularnost resursa, kao što su redovi u tabeli, povećava se konkurentnost ali se troškovi povećavaju takođe jer više zaključavanja moramo obraditi istovremeno.
- Zaključavanje većih granularnosti, kao što su tabele, smanjićemo opšte troškove ali ćemo zaključati celu bazu, odnostno niko neće moći da pristupi bilo kojem delu te baze, čime ćemo smanjiti konkurentnost.

# Granularnost i hijerarhija zaključavanja

- Primenljivo na: SQL Server.
- Database Engine mora da dobije multi level granularnosti da bi potpuno zaštitili podatke.
- Ove grupe zaključavanja na multi levelu granularnosti se zovu **hijerarhija zaključavanja**.
- Primer: Da bi potpuno zaštitili čitanje indeksa, Database Engine zahteva deljeno(shared) zaključavanje redova i (intent shared) deljeno zaključavanje stranica i tabela u bazi podataka.

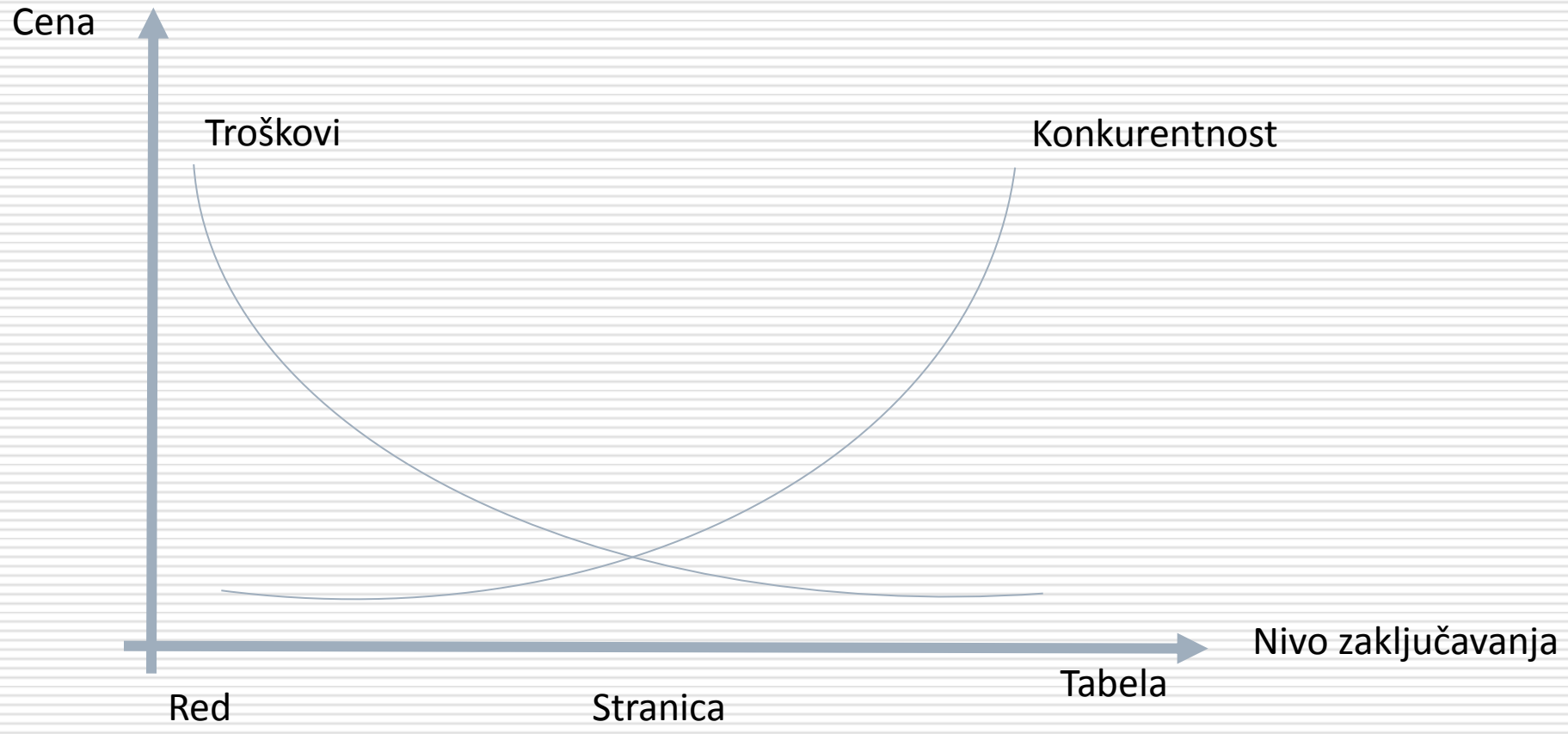
# Granularnost i hijerarhija zaključavanja

- Nivoi zaključavanja:
- RID (Row Identifier) – zaključavanje jednog reda.
- KEY – zaključavanje reda unutar indeksa da bi zaštitili ključeve prilikom serijabilnih transakcija.
- PAGE – 8KB stranica u bazi podataka.
- EXTENT – grupa od 8 susednih stranica.
- HoBT – a heap or B-tree. Zaključavanje štiti B-tree ili heap stranice u tabeli koje još uvek nemaju grupisane indekse.

# Granularnost i hijerarhija zaključavanja

- TABLE – zaključavanje cele tabele, uključujući podatke i indekse.
- FILE – fajl baze podataka.
- APPLICATION – zaključavanje na nivou aplikacije.
- METADATA – zaključavanje meta podataka.
- ALLOCATION_UNIT – jedinica alokacije (veličina klastera)
- DATABASE – zaključavanje cele baze podataka.

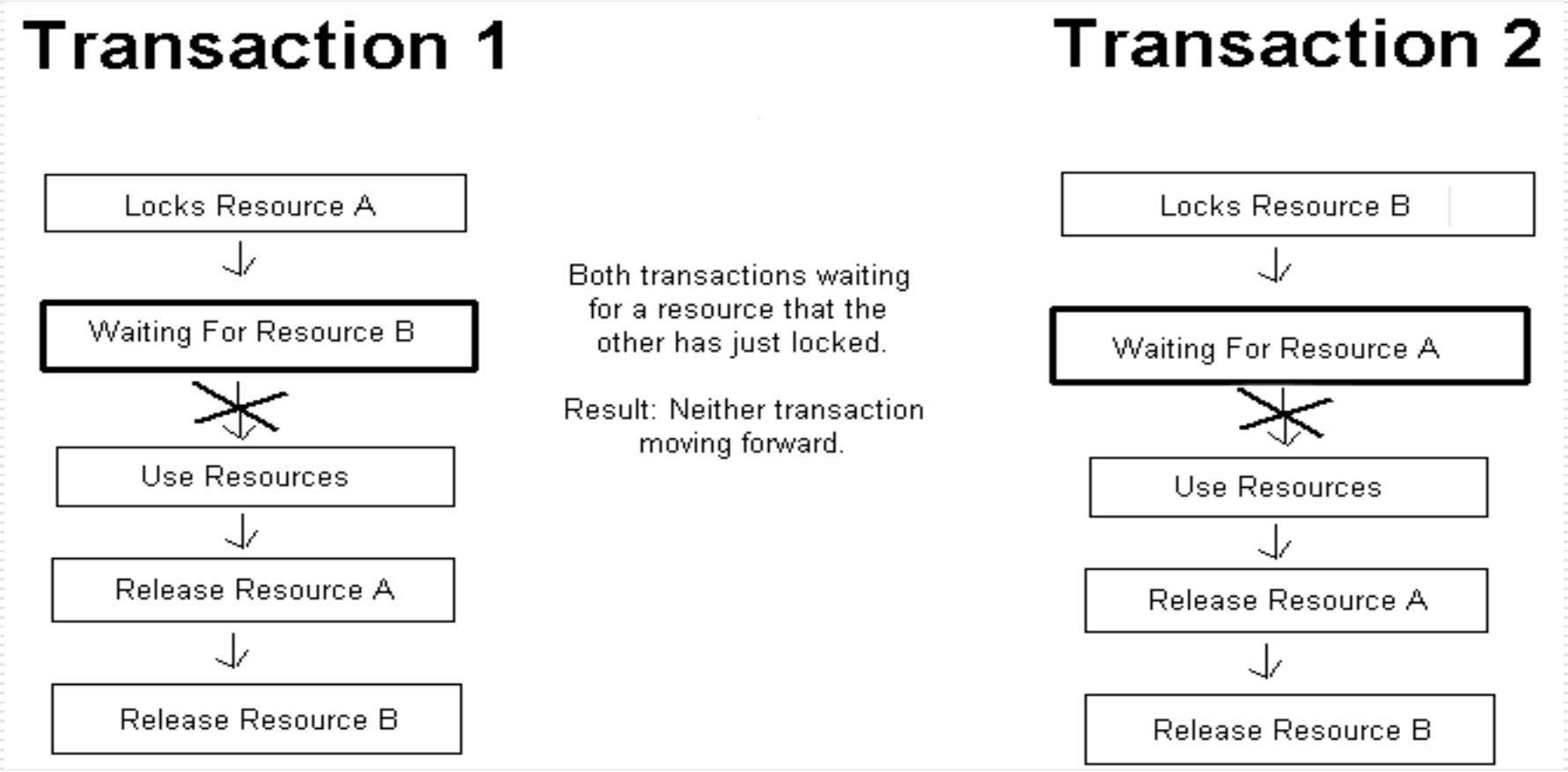
# Granularnost i hijerarhija zaključavanja



# Modovi zaključavanja

- Microsoft SQL Server Database Engine zaključava resurse koristeći različite modove za zaključavanje kojima odlučuje na koji način se može pristupati podacima od strane konkurentnih transakcija.
- Modovi zaključavanja koje Database Engine koristi:
- **Shared** (S) – koristi se za operacije čitanja koje ne menjaju podatke, kao sto je SELECT upit.
- **Update** (U) – koristi se za resurse koje možemo promeniti. Štiti od uobičajene forme deadlock-a koji se dešava kada više sesija čita, zaključava i potencijalno menja podatke istovremeno.

# Modovi zaključavanja - deadlock



# Modovi zaključavanja

- **Exclusive** (X) – koristi se za operacije koje ažuriraju podatke kao što su INSERT, UPDATE ili DELETE. Obezbeđuje da ne možemo vršiti višestruko ažuriranje istih podataka istovremeno.
- **Intent** – koristi se da uspostavi hijerarhiju zaključavanja. Tipovi intent zaključavanja su: intent shared (IS), intent exclusive (IX) i shared intent exclusive (SIX).
- **Schema** – koriste se kada se operacije zavisne od šeme tabele izvršavaju. Tipovi schema zaključavanja su: schema modification (Sch-M) i schema stability (Sch-S).

# Modovi zaključavanja

- **Bulk Update** (BU) – koristi se prilikom kopiranja podataka u tabele i kada je TABLOCK hint naveden.
- **Key-Range** - štiti redove koji se čitaju od strane upita prilikom serijabilnih transakcija sa određenim nivoom izolacije. Obezbeđuje da druge transakcije ne mogu da dodaju redove koji će biti kvalifikovani za upit serijabilnih transakcija ako su upiti pokrenuti ponovo.

# Modovi zaključavanja - kompatibilnost

| Mode | NL | IS | IX | S | SIX | X |
|---|---|---|---|---|---|---|
| NL | Yes | Yes | Yes | Yes | Yes | Yes |
| IS | Yes | Yes | Yes | Yes | Yes | No |
| IX | Yes | Yes | Yes | No | No | No |
| S | Yes | Yes | No | Yes | No | No |
| SIX | Yes | Yes | No | No | No | No |
| X | Yes | No | No | No | No | No |