Poglavje 6

Varnost v SUPB

Trije vidiki varnosti v SUPB

1. Transakcijska
   - kaj so transakcije
   - omogočanje istočasnega dela več uporabnikov nad istimi podatki

2. Dostopna
   - kdo sme dostopati do podatkovne baze
   - kdo sme kaj delati s katerimi podatki

3. Podatkovna
   - celovita skrb za varnost podatkov v podatkovni bazi
   - obnavljanje PB

Transakcijska varnost v SUPB

- Definicija transakcije
- Lastnosti transakcij
- Gradniki SUPB povezani z nadzorom sočasnosti ter obnovljivostjo podatkov

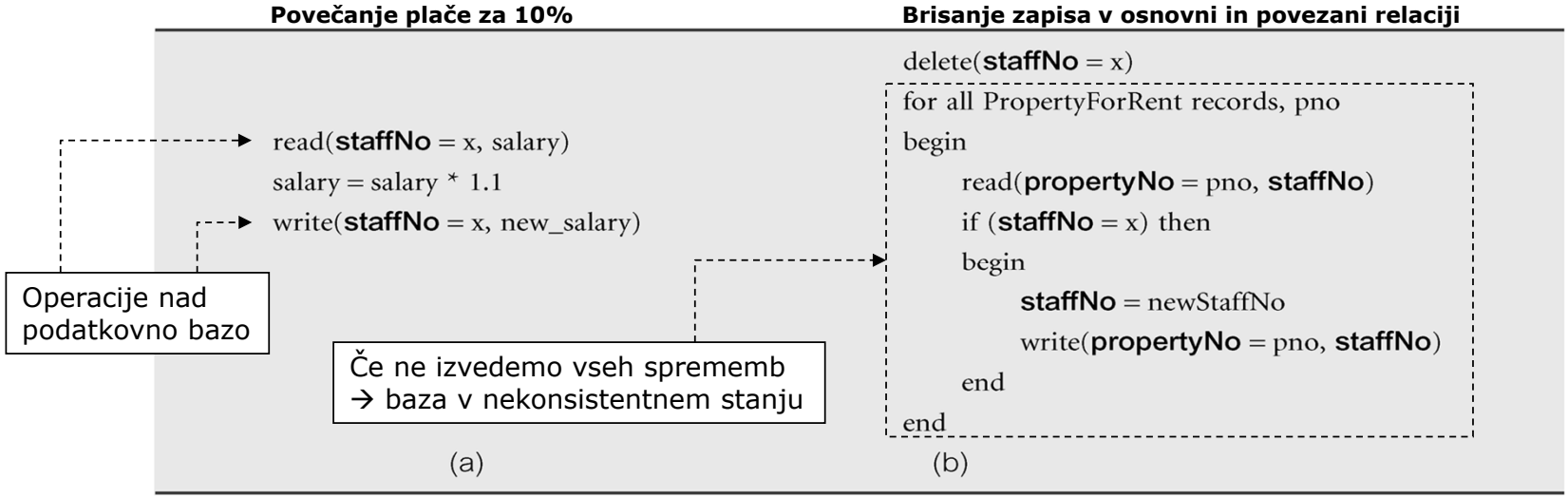
Transakcija - opredelitev oz. definicija

- Transakcija je operacija ali **niz operacij**, ki berejo ali pišejo v podatkovno bazo in so izvedene s strani enega uporabnika oziroma uporabniškega programa.
- Razvijalec določi, katere operacije tvorijo transakcijo (primer: bančni prenos sredstev med računi)
- Transakcija je logična enota dela – lahko je cel program ali samostojen ukaz (npr. INSERT ali UPDATE)
- Izvedba uporabniškega programa je s stališča podatkovne baze vidna kot ena ali več transakcij.

Opredelitev transakcije...

- Primeri transakcij

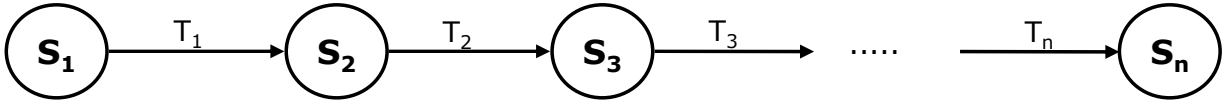
Staff(staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent(propertyNo, street, city, postcode, type, rooms, rent,
ownerNo, staffNo, branchNo)



Opredelitev transakcije...

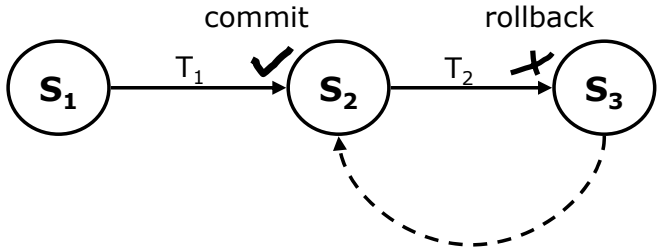
S_i; i=1 .. n ≈ konsistentna ali skladna stanja v podatkovni bazi

Med izvajanjem transakcije je lahko stanje v bazi neskladno!



Opredelitev transakcije…

- Transakcija se lahko zaključi na dva načina:
  - Uspešno ali
  - Neuspešno
- Če končana uspešno, jo potrdimo (commit), sicer razveljavimo (abort, rollback).
- Ob neuspešnem zaključku moramo podatkovno bazo vrniti v skladno stanje pred začetkom transakcije.

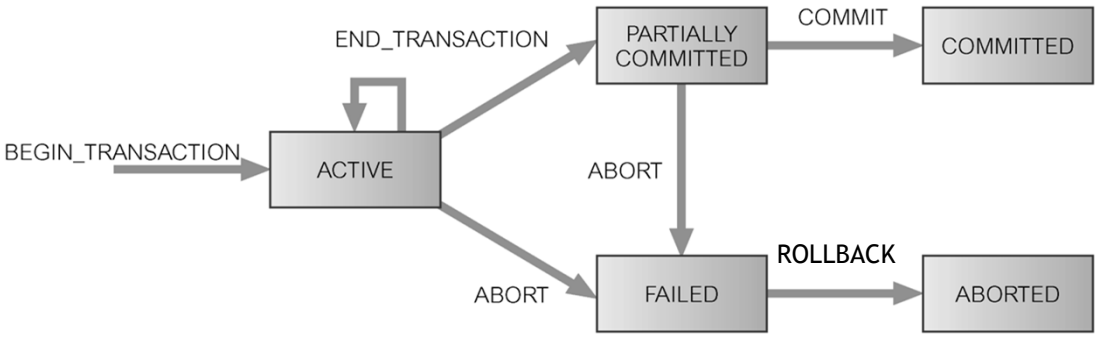


Opredelitev transakcije…

- Enkrat potrjene transakcije ni več moč razveljaviti.
  - Če smo s potrditvijo naredili napako, moramo za povrnitev v prejšnje stanje izvesti novo transakcijo, ki ima obraten učinek nad podatki v podatkovni bazi.
- Razveljavljene transakcije lahko ponovno poženemo.
- Enkrat zavrnjena transakcija je drugič lahko zaključena uspešno (odvisno od razloga za njeno prvotno neuspešnost).

Opredelitev transakcije

- SUPB se ne zaveda, kako so operacije logično grupirane. Uporabljamo eksplicitne ukaze, ki to povedo:
  - Po ISO standardu uporabljamo ukaz BEGIN TRANSACTION za začetek in COMMIT ali ROLLBACK za potrditev ali razveljavitev transakcije.
  - Če konstruktov za začetek in zaključek transakcije ne uporabimo, SUPB privzame cel uporabniški program kot eno transakcijo. Če se uspešno zaključi, izda implicitni COMMIT, sicer ROLLBACK.



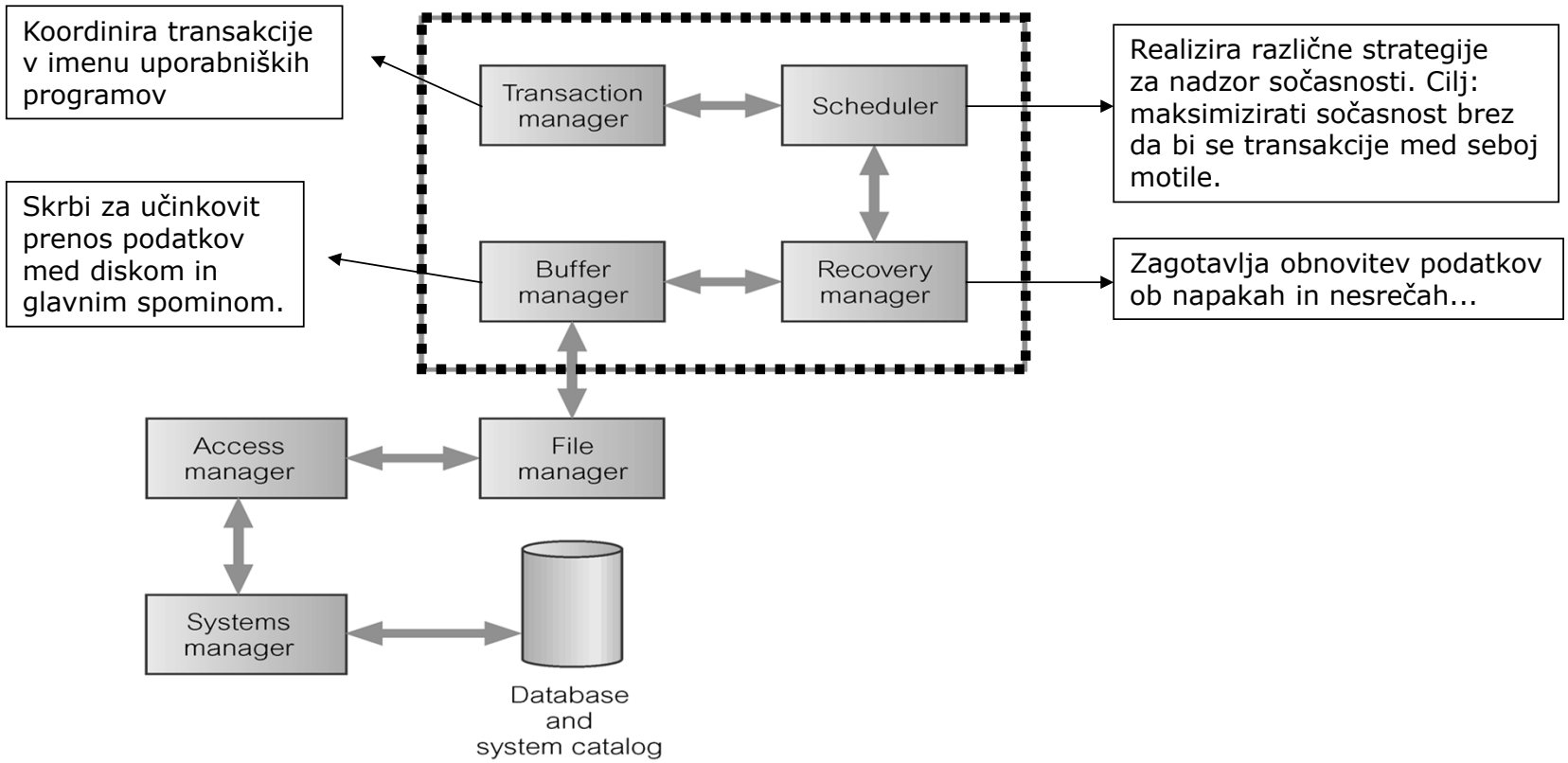
Lastnosti transakcij (ACID*)

- Vsaka transakcija naj bi zadoščala štirim osnovnim lastnostim:
  - Atomarnost: transakcija predstavlja atomaren sklop operacij. Ali se izvede vse ali nič. Atomarnost mora **zagotavljati SUPB**.
  - Konsistentnost: transakcija je sklop operacij, ki podatkovno bazo privede iz enega konsistentnega stanja v drugo. Zagotavljanje konsistentnosti je **naloga SUPB** (zagotavlja, da omejitve nad podatki niso kršene…) in programerjev (preprečuje vsebina neskladnosti).
  - Izolacija: transakcije se izvajajo neodvisno ena od druge → delni rezultati transakcije ne smejo biti vidni drugim transakcijam. Za izolacijo **skrbi SUPB**.
  - Trajnost: učinek potrjene transakcije je trajen – če želimo njen učinek razveljaviti, moramo to narediti z novo transakcijo, ki z obratnimi operacijami podatkovno bazo privede v prvotno stanje. Zagotavljanje trajnosti je **naloga SUPB**.

*__ACID__ – **A**tomicity, **C**onsistency, **I**solation and **D**urability

Obvladovanje transakcij – arhitektura

- Komponente SUPB za obvladovanje transakcij, nadzor sočasnosti in obnovitev podatkov:



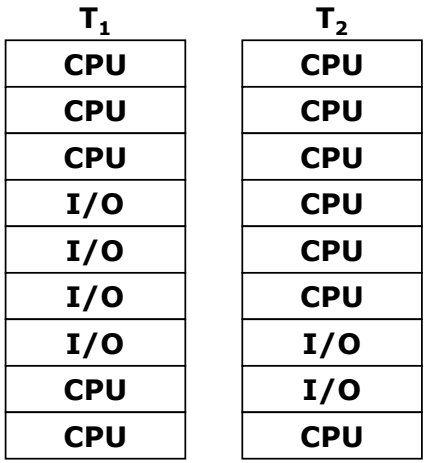
Zakaj sočasnost?...

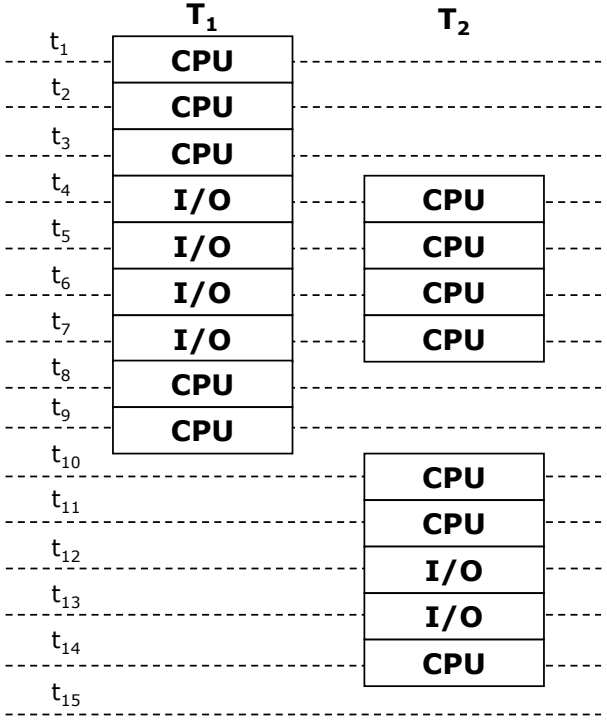
- Eden od ciljev in prednosti PB je možnost sočasnega dostopa s strani več uporabnikov do skupnih podatkov.

- Če vsi uporabniki podatke le berejo – nadzor sočasnosti trivialen;
- Če več uporabnikov sočasno dostopa do podatkov in vsaj eden podatke tudi zapisuje – možni konflikti.

Zakaj sočasnost?...

- Prepletanje operacij dveh transakcij...





Problemi v zvezi z nadzorom sočasnosti in konsistentnosti

- V *centraliziranem* SUPB zaradi sočasnosti dostopa različni problemi:
  - Izgubljene spremembe (lost update): uspešno izveden UPDATE se razveljavi zaradi istočasno izvajane operacije s strani drugega uporabnika.
  - Uporaba nepotrjenih podatkov (dirty read): transakciji je dovoljen vpogled v podatke druge transakcije, še preden je ta potrjena.
  - Neponovljivo branje (neskladnost analize) (non-repeatable read): transakcija prebere več vrednosti iz podatkovne baze. Nekatere izmed njih se v (po navadi daljšem) času izvajanja transakcije zaradi operacij neke druge transakcije spremenijo.
  - Branje fantomskih vrstic (phantom read): transakcija dvakrat izvede poizvedbo in dobi drugič različen rezultat – dodatne, fantomske vrstice, zaradi uspešne operacije neke druge transakcije
- *Decentralizirani* (porazdeljeni) SUPB pa imajo še dodatne probleme

Izgubljene spremembe (lost update)

- Primer:
  - T_1 dvig 10 € iz TRR, na katerem je začetno stanje 100 €.
    - T_2 depozit 100 € na isti TRR.
    - Po zaporedju T_1, T_2 končno stanje enako 190 €.
- Kaj je tu problem?

Time	T_1	T_2	$\mathbf{bal_x}$
t_1		begin_transaction	100
t_2	begin_transaction	read($\mathbf{bal_x}$)	100
t_3	read($\mathbf{bal_x}$)	$\mathbf{bal_x} = \mathbf{bal_x} + 100$	100
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	write($\mathbf{bal_x}$)	200
t_5	write($\mathbf{bal_x}$)	commit	90
t_6	commit		90

Uporaba nepotrjenih podatkov (dirty read)

- Primer:
  - T_3 dvig 10 € iz TRR.
  - T_4 depozit 100 € na isti TRR.
  - Po zaporedju T_3, T_4 končno stanje enako 190 €. Če T_4 preklicana, je pravilno končno stanje 90 €.

Time	T_3	T_4	$\mathbf{bal_x}$
t_1		begin_transaction	100
t_2		read($\mathbf{bal_x}$)	100
t_3		$\mathbf{bal_x} = \mathbf{bal_x} + 100$	100
t_4	begin_transaction	write($\mathbf{bal_x}$)	200
t_5	read($\mathbf{bal_x}$)	⋮	200
t_6	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	rollback	100
t_7	write($\mathbf{bal_x}$)		190
t_8	commit		190

Neponovljivo branje (non-repeatable read)

- Dvakrat izvedemo isti SELECT, dobimo različne vrednosti (UPDATE)
- Vrednost že prebranega podatka se na disku spremeni (UPDATE)
  - Začetno stanje: bal_x=100 €, bal_y=50 €, bal_z=25 €; seštevek je 175 €
  - T_5 prenos 10 € iz TRR_x na TRR_z.
  - T_6 izračun skupnega stanja na računih TRR_x, TRR_y in TRR_z.

Time	T_5	T_6	bal_x	bal_y	bal_z	sum
t_1		begin_transaction	100	50	25	
t_2	begin_transaction	sum = 0	100	50	25	0
t_3	read(bal_x)	read(bal_x)	100	50	25	0
t_4	$bal_x = bal_x - 10$	sum = sum + bal_x	100	50	25	100
t_5	write(bal_x)	read(bal_y)	90	50	25	100
t_6	read(bal_z)	sum = sum + bal_y	90	50	25	150
t_7	$bal_z = bal_z + 10$		90	50	25	150
t_8	write(bal_z)		90	50	35	150
t_9	commit	read(bal_z)	90	50	35	150
t_{10}		sum = sum + bal_z	90	50	35	185
t_{11}		commit	90	50	35	185

Fantomsko branje (phantom read)

- Konceptualno podobno kot neponovljivo pranje
- Dvakratno izvajanje iste poizvedbe znotraj transakcije
  - Lahko se pojavijo **nove vrstice**, ki izpolnjujejo pogoje za vključitev v rezultat, a jih ob prvi izvedbi ni bilo (posledica ukaza INSERT)
  - Lahko se izbrišejo **stare vrstice**, ki so izpolnjujevale pogoje za vključitev v rezultat, a jih ob drugi izvedbi ni več (posledica ukaza DELETE)
- Razlika med fantomskimi vrsticami in neponovljivim branjem: pri slednjem se med izvedbo spremenijo vrednosti v **že prebranih vrsticah** (posledica ukaza UPDATE)

Transakcije v SQL

- SQL vsebuje mehanizme za uporabo in (delno) nadzor upravljanja s transakcijami
- **Kako** uporabljamo SQL mehanizme za podporo transakcijam
  - Začetek in konec transakcije
  - Stopnje izolacije
  - Dodatki ukazom
  - Preverjanje omejitev

Transakcije v SQL

- Standardni ISO SQL definira transakcijski model z ukazoma COMMIT in ROLLBACK
  - Transakcija se začne na začetku programa ali neposredno za COMMIT/ROLLBACK
- Razširitve z vpeljavo dodatnih parametrov izvajanja:
  - PostgreSQL, MySQL: START TRANSACTION
  - Microsoft Transact-SQL: BEGIN TRANSACTION
  - Oracle: nima tega ukaza
  - START/BEGIN TRANSACTION implicitno izvede COMMIT predhodne transakcije
- Transakcija je logična enota dela z enim ali več SQL ukazi. S stališča zagotavljanja skladnega stanja je atomarna.
- Spremembe, ki so narejene znotraj poteka transakcije, niso vidne navzven drugim transakcijam, dokler transakcija ni končana.

Transakcije v SQL

- Transakcija se lahko zaključi na enega od štirih načinov:
  - Transakcija se uspešno zaključi s COMMIT; spremembe so permanentne.
  - Transakcija se prekine z ROLLBACK; spremembe, narejene s transakcijo, se razveljavijo.
  - Program, znotraj katerega se izvaja transakcija, se uspešno konča in zaključi sejo (session). Transakcija je potrjena implicitno (brez COMMIT).
  - Program, znotraj katerega se izvaja transakcija, se ne konča uspešno. Transakcija se implicitno razveljavi (brez ROLLBACK).

Transakcije v SQL

- Nova transakcija se začne z novim SQL stavkom, ki transakcijo začne (prvi stavek, za BEGIN/START TRANSACTION, za COMMIT ali ROLLBACK).
- SQL transakcij po standardu ne moremo gnezditi.
- Transakcijske nastavitve upravljamo s pomočjo ukaza SET TRANSACTION

```
SET TRANSACTION [READ ONLY | READ WRITE] |
    [ISOLATION LEVEL
        READ UNCOMMITTED    | READ COMMITTED |
        REPEATABLE READ     | SERIALIZABLE    ]
```

Transakcije v SQL

- READ ONLY – pove, da transakcija vključuje samo operacije, ki iz baze berejo.
  - SUPB bo dovolil INSERT, UPDATE in DELETE samo nad začasnimi tabelami.
- ISOLATION LEVEL – pove stopnjo interakcije, ki jo SUPB dovoli med to in drugimi transakcijami.

- Ukaz velja za naslednjo transakcijo (MySQL) ali za tekočo transakcijo (PostgreSQL, Oracle) neposredno ob začetku

Nastavljanje lastnosti za več kot eno transakcijo

- Različno od sistema do sistema, ni po ISO SQL standardu
- Oracle (za tekočo sejo):
  ALTER SESSION SET TRANSACTION …
- MySQL (za tekočo sejo ali globalno z ustreznimi pravicami):
  SET [GLOBAL | SESSION] TRANSACTION …
- PostgreSQL:
  SET SESSION CHARACTERISTICS AS TRANSACTION …
- Nadaljnja sintaksa je enaka kot pri SET TRANSACTION …
- V aplikaciji nam lahko pomaga knjižnica (npr. pri SQLAlchemy lahko pri `create_engine` navedemo parameter `isolation_level`)

Transakcije v SQL

- Učinek SET TRANSACTION ISOLATION LEVEL

	Branje neobsto-ječega podatka	**Neponovlji-vo branje**	**Fantomsko branje**	**Izgubljeno ažuriranje**
Read Uncommitted	Da	Da	Da	Da (eno- in dvodelni update)
Read Committed	Ne	Da	Da	Da (dvodelni update)
Repeatable Read	Ne	Ne	Da	Ne
Serializable	Ne	Ne	Ne	Ne

- Različne stopnje izolacije izbiramo zaradi različnega obsega želene sočasnosti (kompromis)