Poglavje 9

Sočasni dostop do podatkovne baze

Povzeto po [4]

Sočasni dostop do PB..

- Sodobni OS omogočajo izvajanje več programov hkrati
- Poleg SUPB se tako lahko izvaja več uporabniških programov, ki dostopajo do PB hkrati
- Transakcije, ki jih izvajajo uporabniški programi se izvajajo prepletajoče, zato tudi SUPB izvaja ukaze prepletajoče
- Prepletajoče izvajanje transakcij pa skriva dve pasti:
  - podatkovna baza lahko zaide v nekonsistentno stanje,
  - rezultati povpraševanj v podatkovni bazi so lahko napačni

Sočasni dostop do PB..

- Prepletajoča izvedba transakcij lahko bazo pusti v nekonsistentnem stanju, čeprav sta obe transakciji pravilno izvedli svoja ažuriranja. Primera:
  - "**izgubljeno ažuriranje**": na osnovi istega prebranega zapisa se izvede ažuriranje istega zapisa s strani dveh transakcij, kar pomeni, da obvelja samo zadnje ažuriranje,
  - **branje "neobstoječega zapisa"**: predpostavimo, da neka transakcija prebere zapis, ki ga je ravnokar ažurirala druga transakcija, ki se je takoj zatem ponesrečila in zato razveljavila. Prva transakcija je torej prebrala zapis (oz. njegovo stanje), ki ni nikoli veljavno obstajal v PB!!!

Sočasni dostop do PB..

- Naloga nadzora nad sočasno uporabo PB (*concurrency control*) je:
  - ohraniti podatkovno bazo v konsistentnem stanju,
  - dopustiti čimvečjo sočasnost izvajanja transakcij
- V ta namen se uporabljata dve tehniki:
  - zaseganje zapisov (*locking*),
  - časovno označevanje (*timestamping*)

Sočasni dostop do PB

- Sočasno izvajanje transakcij bomo obravnavali na primeru centralizirane PB, ki jo upravlja en sam SUPB
- Do problemov sočasnega izvajanja transakcij prihaja tudi pri porazdeljenih PB. Ker se tam transakcije zares izvajajo sočasno, so problemi sočasnosti izvajanja transakcij še težje rešljivi. Niso pa nerešljivi!

Sočasno izvajanje transakcij..

- Z vidika nadzora nad sočasnim izvajanjem transakcij so pomembni predvsem tisti ukazi, ki spreminjajo vsebino PB
- Omejili se bomo na naslednje ukaze:
  - Pomni,
  - Pozabi,
  - PoiščiPreberi(X, x),
  - Ažuriraj(X, x) (Ta ukaz zamenjuje oz. pokriva ukaze: Dodaj(X, x), Spremeni(X, x), Izbriši(X)).

Sočasno izvajanje transakcij..

- Zaporedje izvajanja ukazov v okviru (več) transakcij se imenuje razpored (*schedule*)
- Zaporeden razpored: pri sočasnem izvajanju transakcij se najprej izvedejo vsi ukazi ene transakcije, nato pa vsi ukazi druge transakcije (primeri na naslednjih straneh)

Sočasno izvajanje transakcij..

Zamislimo si banko, ki vodi račune svojih komitentov. Komitenti med seboj kupčujejo, zato je potrebno pri vsaki kupčiji izvesti transakcijo - prenos sredstev z računa enega komitenta na račun drugega komitenta. Po izvedbi ene ali večih transakcij mora vsota sredstev na vseh računih skupaj ostati nespremenjena.

Omejimo se na dve transakciji T0 in T1. Izvedeta se lahko po dveh različnih zaporednih razporedih, in po celi vrsti izmeničnih razporedov. Pri tem se lahko posamezna transakcija zaključi uspešno (z ukazom Pomni) ali pa neuspešno (z ukazom Pozabi), pri čemer pa se vsa njena že izvedena ažuriranja v podatkovni bazi razveljavijo. Predpostavili bomo tudi, da se ažuriranja izvajajo kot sprotna ažuriranja.

T0	*T1*
PoiščiPreberi(R1,a) *a := a - 10* *Ažuriraj(R1,a)* *PoiščiPreberi(R2,b)* *b := b + 10* *Ažuriraj(R2,b)* *Pomni*	*PoiščiPreberi(R2,c)* *c := c - 20* *Ažuriraj(R2,c)* *PoiščiPreberi(R3,d)* *d := d + 20* *Ažuriraj(R3,d)* *Pomni*

T0	*T1*
PoiščiPreberi(R1,a) *a := a - 10* *Ažuriraj(R1,a)* *PoiščiPreberi(R2,b)* *b := b + 10* *Ažuriraj(R2,b)* *Pomni*	*PoiščiPreberi(R2,c)* *c := c - 20* *Ažuriraj(R2,c)* *PoiščiPreberi(R3,d)* *d := d + 20* *Ažuriraj(R3,d)* *Pomni*

Sočasno izvajanje transakcij..

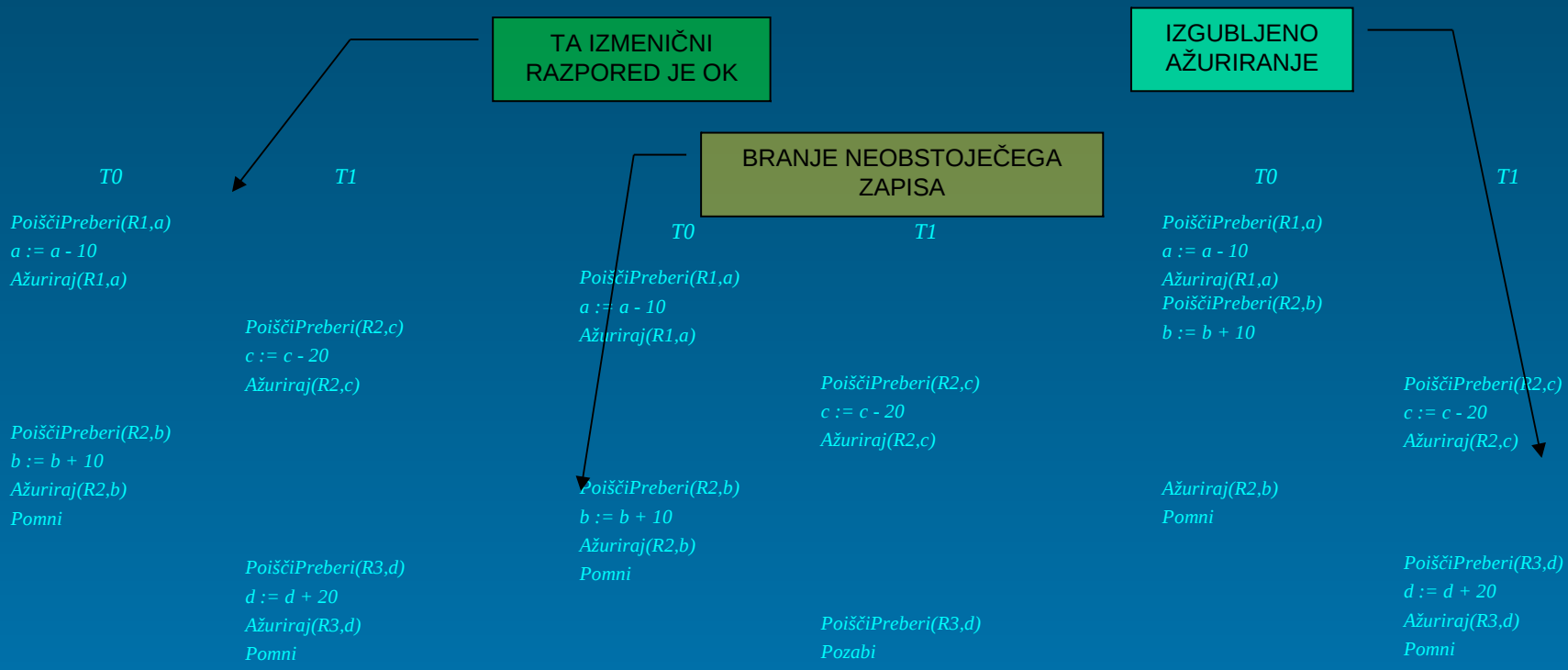
- Ukaz transakcije T1 - PoiščiPreberi(R3,d) se ni uspešno izvršil
- Transakcijski program je izdal ukaz Pozabi, SUPB pa je nato transakcijo T1 razveljavil in s tem obnovil podatek R2 v prvotno stanje
- Stanje v podatkovni bazi po izvedenih transakcijah je R1 = 90, R2 = 210 in R3 = 300. Merilo uspešnosti je ohranjanje vsote 600
- Razpored je zaporeden

Zaporedni razpored B2

T0	*T1*
	PoiščiPreberi(R2,c)
	c := c - 20
	Ažuriraj(R2,c)
	PoiščiPreberi(R3,d)
	Pozabi
PoiščiPreberi(R1,a)	
a := a - 10	
Ažuriraj(R1,a)	
PoiščiPreberi(R2,b)	
b := b + 10	
Ažuriraj(R2,b)	
Pomni	

Sočasno izvajanje transakcij..

- **Prepletajoč razpored: med ukazi ene transakcije se izvajajo tudi ukazi drugih transakcij.**



Sočasno izvajanje transakcij..

- Transakcije so med seboj neodvisne, če se lahko izvajajo v poljubnem vrstnem redu: zaporedno ali prepletajoče, pri čemer se kakšne izmed transakcij lahko zaključijo tudi neuspešno
- Velja naslednje: Če vsaka izmed transakcij ohranja konsistentnost podatkovne baze, potem jo ohranja tudi vsak njihov zaporedni razpored

Sočasno izvajanje transakcij..

- Nekateri prepletajoči razporedi ne ohranjajo konsistentnosti PB
- Tisti prepletajoči razpored, ki učinkuje na PB enako kot kak izmed zaporednih razporedov, se imenuje zaporedniški razpored
- Velja naslednje: Če vsaka izmed transakcij ohranja konsistentnost podatkovne baze, potem jo ohranja tudi vsak njihov zaporedniški razpored

Sočasno izvajanje transakcij..

- Nadzor nad sočasno uporabo PB mora zagotoviti prepletajoče izvajanje transakcij po enem izmed možnih zaporedniških razporedov
- Pri sočasni uporabi PB se včasih pripeti, da se konsistentnost podatkov ohrani, nekonsistentni pa so rezultati povpraševanja, kar kaže primer na desni
- **RAZLOG**: vsota po vseh računih se je ohranila in je enaka 600, transakcija T2 pa jo je izračunala napačno: v = 610. Razlog je v tem, ker je T2 prebrala podatek R1 pred ažuriranjem, podatek R2 pa po ažuriranju

T0	*T2*
	v := 0
PoiščiPreberi(R1,a)	
a := a - 10	
	PoiščiPreberi(R1,s)
	v := v + s
Ažuriraj(R1,a)	
PoiščiPreberi(R2,b)	
b := b + 10	
Ažuriraj(R2,b)	
Pomni	
	PoiščiPreberi(R2,s)
	v := v + s
	PoiščiPreberi(R3,s)
	v := v + s
	Pomni

Sočasno izvajanje transakcij

- Rezultat povpraševanja transakcije T2 iz primera lahko ocenimo kot približno točen ali pa kot napačen
- Če so zahtevani popolnoma natančni rezultati, mora nadzor nad sočasno uporabo podatkovne baze zagotoviti tudi konsistentnost povpraševanj v podatkovni bazi

Zaseganje zapisov..

- Možna rešitev predstavljenih problemov: možnost, da si transakcijski program pridobi izključno pravico dostopa do zapisov, do katerih transakcija dostopa
- S tem se prepreči vmešavanje sočasnih transakcij v njen postopek ažuriranja

Zaseganje zapisov..

- Pravilo po katerem se mora ravnati SUPB, da zaščiti konsistentnost PB, je poimenovano kot “Piščevo pravilo”:
  - Ko se v okviru transakcij izvaja ažuriranje dela PB, mora biti celotno zaporedje operacij ažuriranja zaščiteno pred vmešavanjem s strani transakcij, ki žele sočasno ažurirati isti del PB

Zaseganje zapisov..

- Pravilo po katerem se mora ravnati SUPB, da zaščiti konsistentnost rezultatov povpraševanja, je poimenovano kot “Bralčevo pravilo”:
  - Ko se v okviru transakcije izvaja le povpraševanje v PB, potem je lahko (ali pa tudi ne) celotno zaporedje operacij povpraševanja zaščiteno pred vmešavanjem s strani transakcij, ki bi žele ažurirati isti del PB
- Ali želi “bralec” zaščito ali ne, je odvisno od tega, ali želi priti do točnih ali samo do približnih rezultatov
- Zaščita pred vmešavanjem sočasnih transakcij se izvede s pomočjo zaseganja zapisov

Zaseganje zapisov..

- Zaradi tega se ukazom za upravljanje s podatki dodajo še ukazi za zaseganje:
  - E (exclusive)-zaseži(X),
  - D (shared)-zaseži(X),
  - Sprosti(X).
- Način delovanja: transakcijski program izda zahtevo po zaseženju. Če je zaseženje mogoče, to SUPB takoj odobri. Če to ni možno, mora transakcija na odobritev počakati

Zaseganje zapisov..

- 2 pravili glede zaseganja podatkov:
  - ekskluzivno zaseženi podatek se ne more še dodatno zaseči, niti ekskluzivno, niti delno,
  - deljeno zaseženi podatek se lahko dodatno deljeno zaseže, ne more pa se zaseči ekskluzivno.

Zaseganje zapisov

- Kompatibilnostna matrika o tem katera zaseženja podatkov so možna, glede na trenutno zaseženje teh podatkov:

		Zahteva po zaseženju		
		E	**D**	**-**
Trenutno zaseženje	**E**	ne	ne	da
	D	ne	da	da
	-	da	da	da

Zaseganje zapisov - PXC, PSC..

- Pri izvajanju transakcij je potrebno pri zaseganju podatkov upoštevati določena pravila, ki jim pravimo protokol
- Protokol zagotavlja, da se prepletajoče izvajanje transakcij izvaja po enem izmed zaporedniških razporedov
- 2 protokola:
  - PXC (Protocol eXclusive Commit): temelji na ekskluzivnem zaseganju podatkov,
  - PSC (Protocol Shared Commit): temelji na ekskluzivnem in deljenem zaseganju

Zaseganje zapisov - PXC, PSC..

- Pravila, ki jih uvaja PXC protokol:
  - transakcija, ki želi podatek ažurirati, ga mora najprej ekskluzivno zaseči,
  - če zahteva po zaseženju ne more biti takoj odobrena, preide transakcija v stanje čakanja, njeno izvajanje se nadaljuje po odobritvi zaseženja,
  - vsa zaseženja se smejo sprostiti šele po zaključku transakcije (uspešnem ali neuspešnem),
  - transakcija, ki želi le prebrati podatek in ji ni mar za sočasno ažuriranje tega podatka s strani kake druge transakcije, ga sme prebrati ne glede na to, ali je podatek zasežen ali ne

Zaseganje zapisov - PXC, PSC..

- Dodatno vsebuje protokol PSC še naslednje pravilo:
  - Transakcija, ki želi ekskluzivno zaseči podatek, mora imeti pred tem odobreno njegovo deljeno zaseženje
- Razlike med protokoloma so v obsegu dopuščenega sočasnega izvajanja transakcij:
  - PSC dopušča sočasno izvajanje dveh zgolj povpraševalnih transakcij, ki zahtevata zaščito pred ažuriranjem; PXC pa ne,
  - PSC veliko bolj dopušča nastop mrtve zanke (medsebojno blokiranje transakcij)

Zaseganje zapisov - PXC, PSC..

- Da se protokol za zaseganje podatkov lahko izvaja, mora SUPB za podatke voditi evidenco o tem:
  - ali podatek je ali ni zasežen,
  - kako je zasežen,
  - kdo vse ga je zasegel,
  - kdo vse ga želi zaseči in na kakšne način.
- Zato se za vsak podatek vzdržujeta 2 listi:
  - lista odobrenih zaseženj, ki vsebuje pare: (Oznaka transakcije, Vrsta Odobrenega Zaseženja),
  - lista zahtevanih zaseženj, ki vsebuje pare: (Oznaka transakcije, Vrsta Zahtevanega Zaseženja).

Zaseganje zapisov - PXC, PSC

- SUPB mora med izvajanjem transakcij ves čas vzdrževati obe listi
- Do sprememb na listah prihaja:
  - pri sprejetju zahteve po zaseženju,
  - pri odobritvi zaseženja,
  - ob zaključku transkacije

Zaseganje zapisov – Objekti zaseženja..

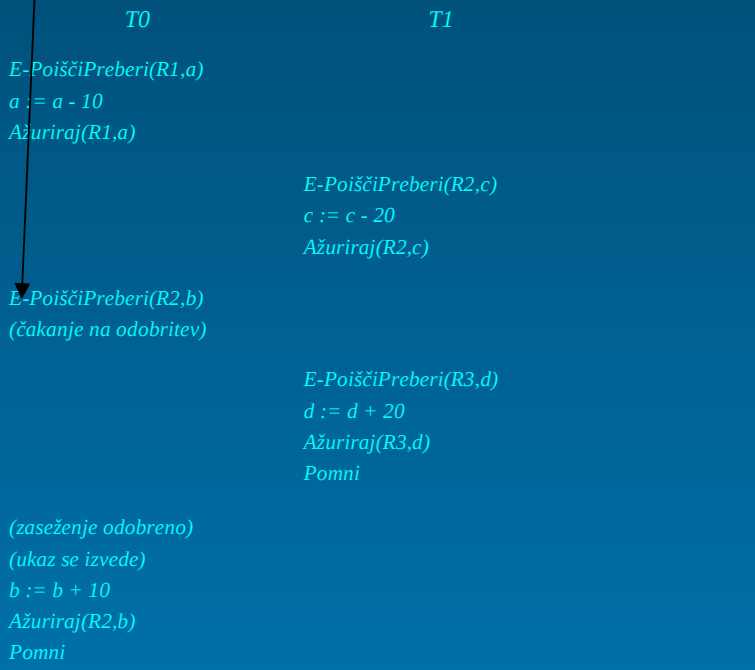
- Objekti zaseženja so lahko različni:
  - logični,
  - fizični.
- Logični objekti zaseženja:
  - relacije,
  - n-terice v relacijah,...
- Fizični objekti zaseženja:
  - celotna fizična podatkovna baza,
  - tabele,
  - fizični bloki oz. strani,
  - fizični zapisi v tabeli.

Zaseganje zapisov – Objekti zaseženja

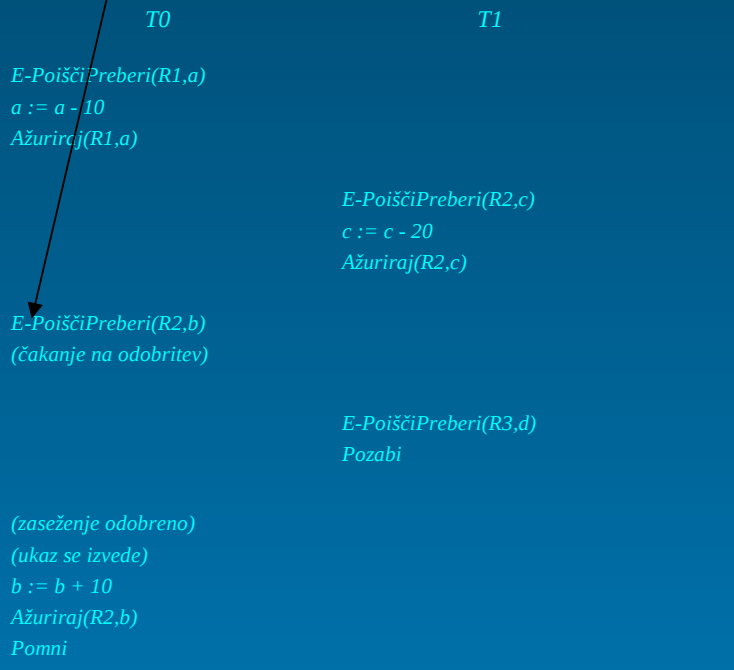
- Granulacija objektov zaseženja vpliva na:
  - obseg sočasnoti pri izvajanju transakcij,
  - obseg podatkov o odobrenih in zahtevanih zaseženjih,
  - stopnjo dodatne obremenitve SUPB z izvajanjem nadzora nad zaseženji.
- Ukaza “pomni” in “pozabi” pomenita tudi sprostitev zaseženj podatkov
- Protokol PXC uvaja še ukaz:
  - E-PoiščiPreberi(X,x),
- Protokol PSC pa ukaza:
  - D-PoiščiPreberi(X,x) in
  - E-Ažurirarj(X,x). Najprej zahteva razširitev zaseženja, potem izvede ažuriranje

Zaseganje zapisov – Primeri..

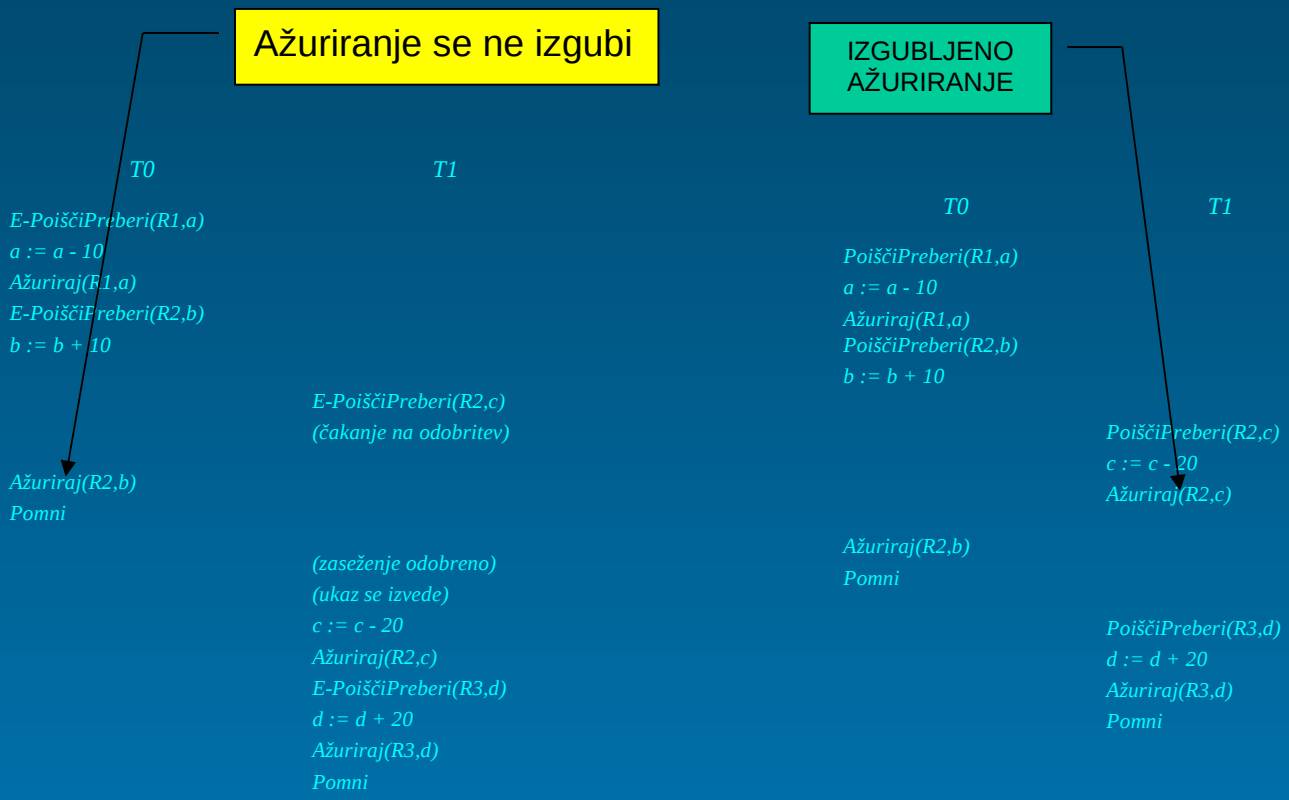
Ker hoče T0 zaseči podatek, ki ga je pred tem zasegla že T1, mora čakati.



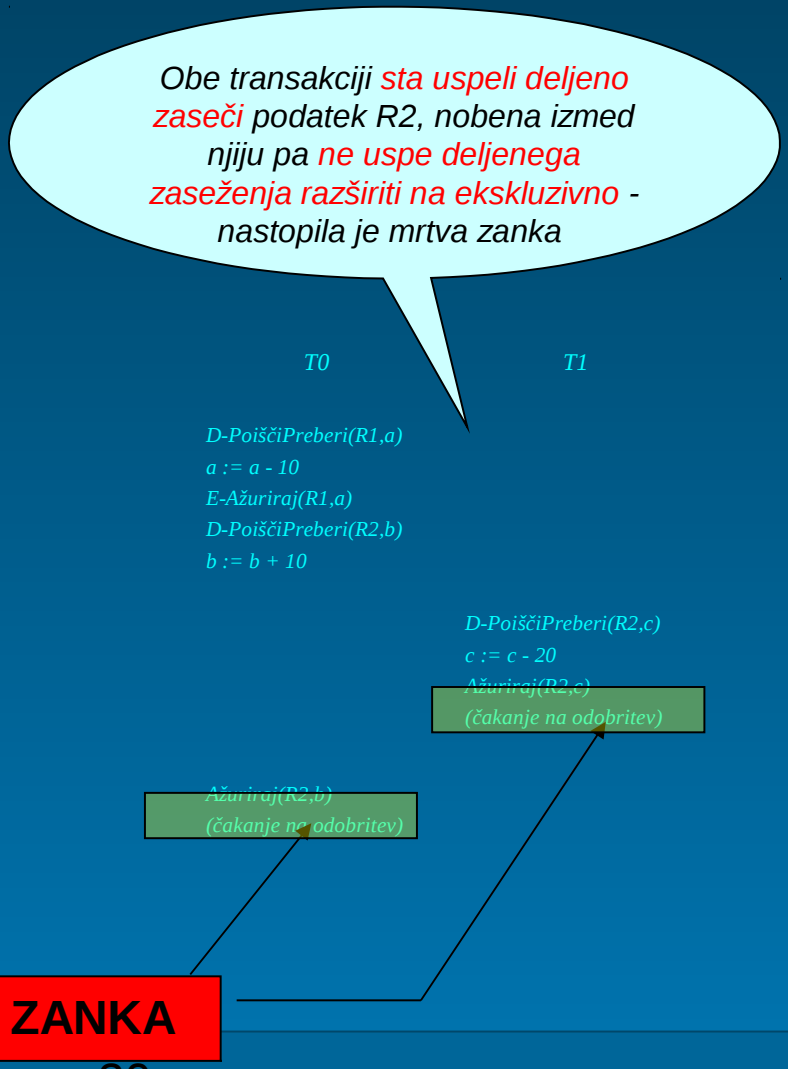
Tukaj bi se izvedlo branje neobstoječega podatka. Z zaseganjem je razpored zaporedniški.



Zaseganje zapisov – Primeri..



Zaseganje zapisov – Primeri

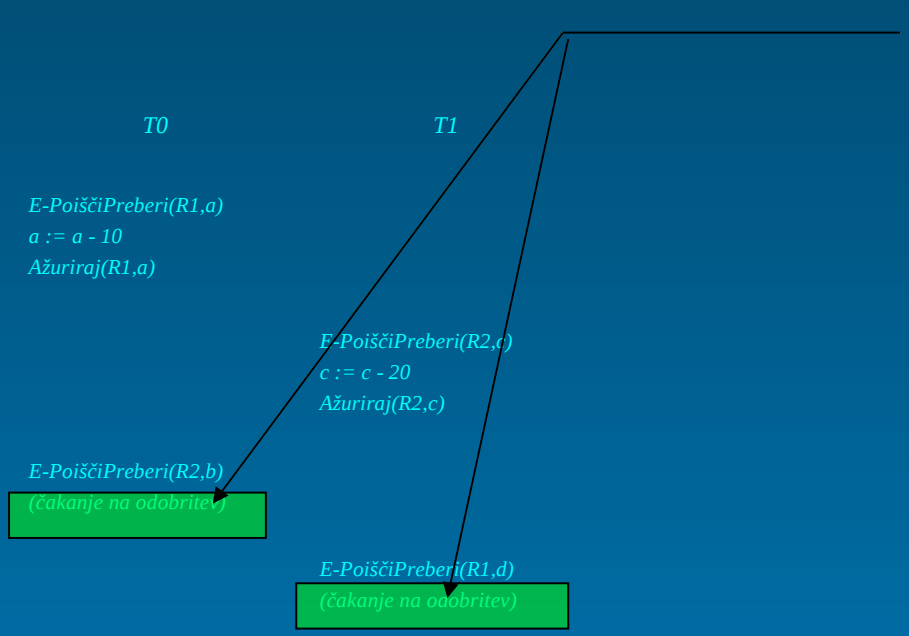


Mrtva zanka – Uvod..

- Pri sočasnem izvajanju transakcij se pojavljata naslednja problema:
  - livelock: Posamezno transakcijo lahko pri odobritvi zaseženja določenega podatka prehitevajo vse ostale transakcije in se s tem njeno čakanje raztegne na nedoločen čas (problem je rešljiv z algoritmi za dodeljevanje zaseženj),
  - mrtva zanka (deadlock): ko dve (ali več transakcij) zaseže vsaka svoj podatek, vsaka od njiju pa želi še podatek, ki ga je že zasegla tekmica. Zaradi tega transakciji čakata druga na drugo, in ker se po protokolu sprostijo zaseženi podatki le ob zaključku transakcije, ni čakanja nikoli konec.

Mrtva zanka – Uvod..

- Primer nastopa mrtve zanke, ki lahko nastopi tudi pri PXC:



Vsaka transakcija hoče zaseči podatek, ki ga je zasegla že druga transakcija.

Mrtva zanka – Uvod

- Problem mrtve zanke se razreši na dva načina:
  - preprečimo, da se mrtva zanka sploh lahko pojavi, kar rešujemo z enim ali več od naslednjih prijemov: urnikom izvajanja transakcij, vnaprejšnja zahteva po zaseženjih zapisov, ureditev objektov zaseganja, odloča transakcijski program in prekinitev in ponovno izvajanje transakcij
  - odpravimo mrtvo zanko potem, ko je ta že nastopila

Preprečevanje nastopa mrtve zanke..

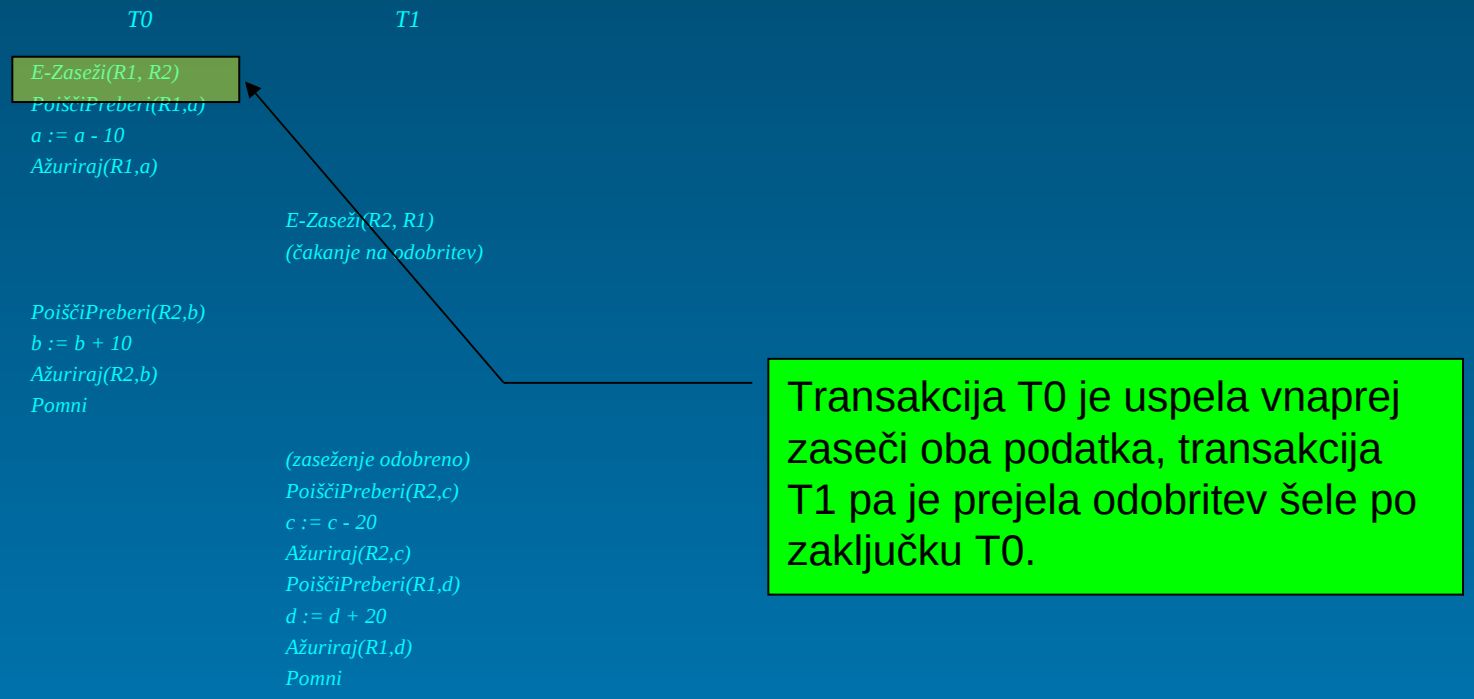
- **Urnik izvajanja transakcij**:
  - Z izvajanjem transakcij po urniku ne dopustimo sočasnega izvajanja takih transakcij oziroma transakcijskih programov, ki bi utegnili imeti konfliktne podatkovne zahteve
  - V tem primeru tudi ni potrebno izvajati zaseganja podatkov. Pravimo, da se na ta način poveča "propustnost" podatkovne baze
  - Ker praviloma ni vnaprej znano katere podatke bo transakcija ažurirala, se možna sočasnost (prepletajoči način) izvajanja transakcij zelo omeji
  - V najneugodnejšem primeru je urnik transakcij ekvivalenten zaseženju celotne podatkovne baze s strani posamezne transakcije

Preprečevanje nastopa mrtve zanke..

- **Vnaprejšnja zahteva po zaseženju podatkov**:
  - Transakcija postavi pred svojim prvim ažuriranjem zahtevo po zaseženjih vseh podatkov, ki jih namerava ažurirati:
    - Če je možno vsa zahtevana zaseženja takoj odobriti, jih tudi SUPB odobri in transakcija se lahko nemoteno izvaja vse do svojega zaključka.
    - Če kakšnega izmed zaseženj ni možno odobriti, se ji ne odobri nobenega zaseženja in transakcija preide v stanje čakanja.
  - Pri taki rešitvi ne sme biti izbor podatka, ki ga bo transakcija ažurirala, pogojen z uspehom ali neuspehom kakšnega predhodnega ažuriranja, zato ker v takem primeru transakcija:
    - bodisi ne more vedeti, katere podatke naj zaseže,
    - bodisi zaseže preventivno bistveno večje število podatkov, kot bi bilo potrebno, zato se zmanjšuje možnost sočasnega izvajanja drugih transakcij. (primer na naslednji strani)

Preprečevanje nastopa mrtve zanke..

- Primer vnaprejšnje zahteve po zaseženju podatkov:

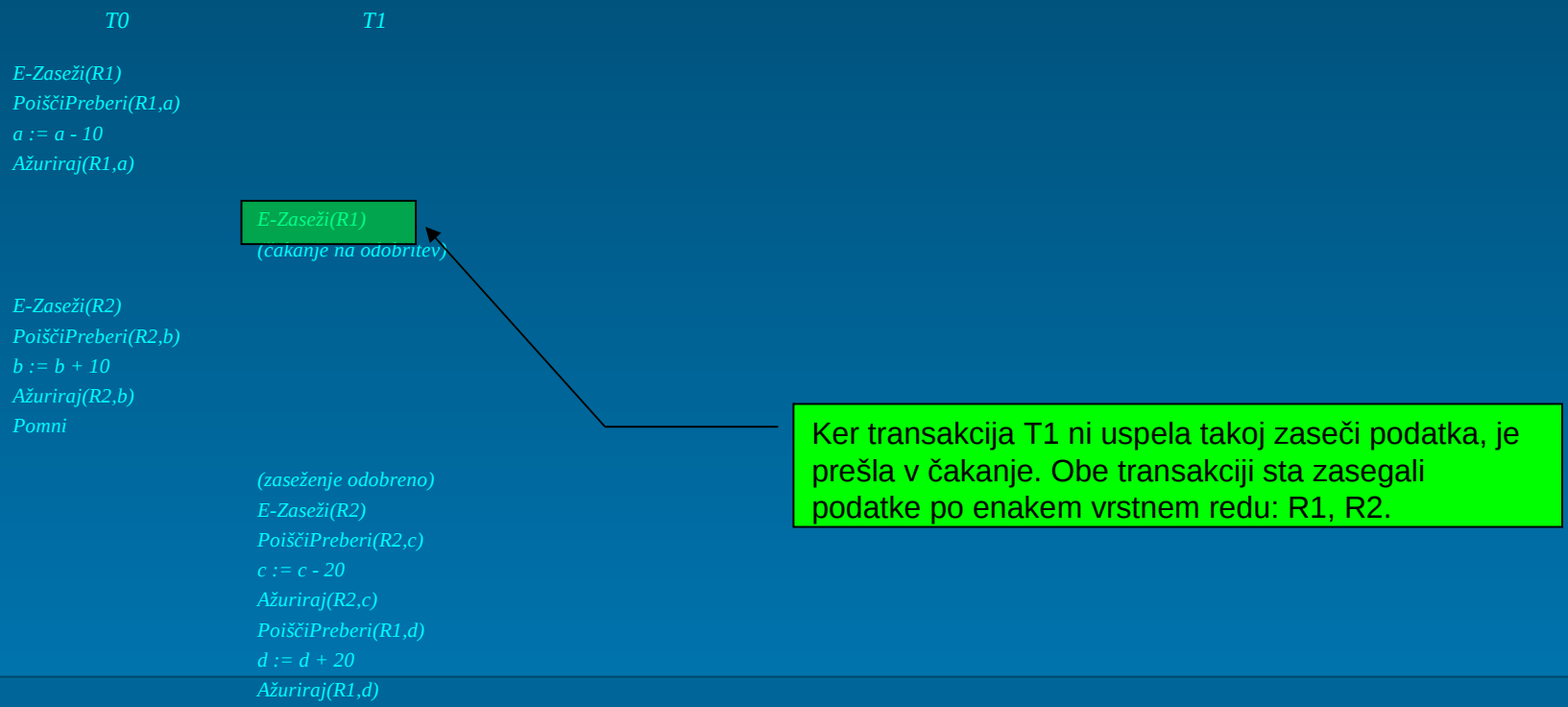


Preprečevanje nastopa mrtve zanke..

- **Ureditev objektov zaseganja**:
  - Objekte zaseganja, ki so lahko zapisi, strani zapisov, n-terice, tabele, se sme zaseči samo po določenem vrstnem redu, ki mora biti znan vnaprej vsem transakcijskim programom
  - Če je objektov zaseganja veliko (npr. zapisi), je določanje njihovega vrstnega reda in tudi nadzor nad pravilnim vrstnim redom zaseganja dokaj zamudno opravilo
  - poleg tega pa je potrebno vnaprej poznati podatke, ki se bodo v okviru transakcije ažurirali. (primer na naslednji strani)

Preprečevanje nastopa mrtve zanke..

- Primer ureditve objektov zaseganja:
  - Predpostavimo, da je potrebno zaseči najprej R1 in šele nato R2



Preprečevanje nastopa mrtve zanke..

- **Odloča transakcijski program**:
  - Če zahteva po odobritvi zaseženja podatka v okviru transakcije ni takoj odobrena, odloči transakcijski program, kako naprej:
    - izvajanje transakcije lahko program takoj prekine in ko jo SUPB razveljavi, prične z njenim ponovnim izvajanjem; lahko pa jo uvrsti na začetek čakalne vrste in prične izvajati transakcijo, ki je naslednja na vrsti;
    - v presledkih lahko izvede nekaj poskusov zaseganja podatka in če ne uspe, prekine izvajanje transakcije.

Preprečevanje nastopa mrtve zanke..

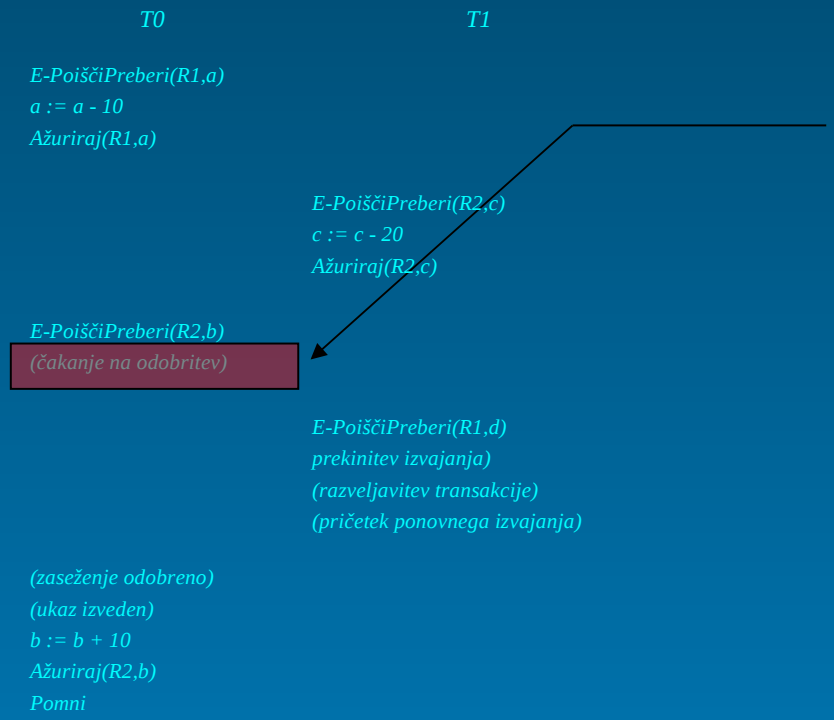
- **Prekinitev in ponovno izvajanje transakcij**:
  - Uporaba protokolov:
    - **Čakaj ali izdihni** (Wait Die) in
    - **Rani ali čakaj** (Wound-Wait).
- Vsaki izmed transakcij pripiše SUPB ob njenem pričetku časovno oznako - njen startni čas. Na ta način je za poljubni dve transakciji možno ugotoviti, katera je "starejša" in katera "mlajša"
- Starejša transakcija je tista, ki se že dlje časa izvaja in je zato njen startni čas manjši

Preprečevanje nastopa mrtve zanke..

- Ko transakcija TA zahteva zaseženje podatka, ki je že zasežen s strani transakcije TB, in se njeni zahtevi zaradi nekompatibilnosti zaseženj ne da takoj ugoditi, se po protokolu “Čakaj ali izdihni” izvede naslednje:
  - če je transakcija TA starejša od TB, preide TA v stanje čakanja na odobritev,
  - če je mlajša, pa se njeno izvajanje prekine, transakcija se razveljavi in posreduje transakcijskemu programu v ponovno izvajanje

Preprečevanje nastopa mrtve zanke..

- Razpored ažuriranj pri uporabi protokola “Čakaj ali izdihni”:



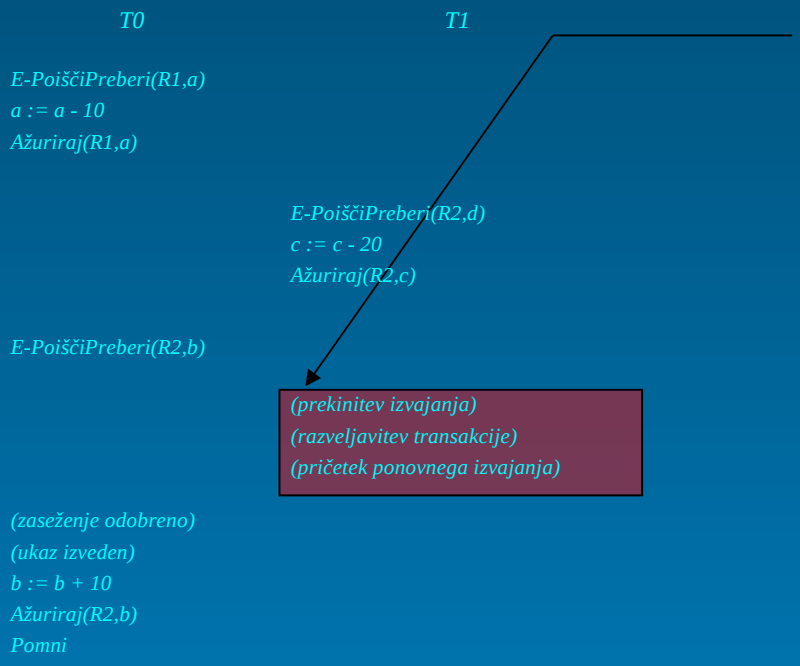
Protokol Čakaj ali izdihni prepreči nastop mrtve zanke. T0 je starejša, zato preide v stanje čakanja na odobritev.

Preprečevanje nastopa mrtve zanke..

- Ko transakcija TA zahteva zaseženje podatka, ki ga je že zasežen s strani transakcije TB, in se njeni zahtevi zaradi nekompatibilnosti zaseženj ne da takoj ugoditi, se po protokolu “Rani ali čakaj” izvede naslednje:
  - če je transakcija TA starejša od TB, se prekine transakcija TB, razveljavi in vrne v ponovno izvajanje. Po razveljavitvi TB se transakciji TA odobri zaseženje podatka,
  - če je TA mlajša, pa preide v stanje čakanja.

Preprečevanje nastopa mrtve zanke

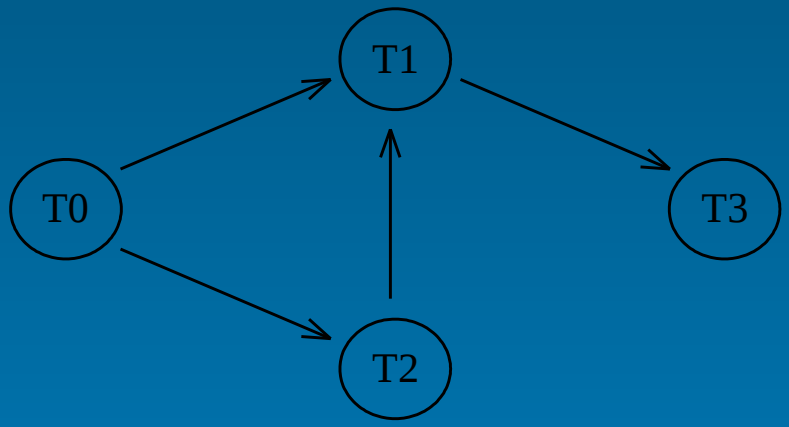
- Razpored ažuriranj pri uporabi protokola “Rani ali čakaj”:



Isti problem kot v prejšnjem primeru se s protokolom Rani ali čakaj izvede na naslednji način. Ker je T0 starejša od T1, se T1 prekine, razveljavi in vrne v ponovno izvajanje.

Odpravljanje mrtve zanke..

- Mrtvo zanko je potrebno najprej odkriti, nato pa pristopimo k njenemu razreševanju
- V ta namen lahko uporabimo čakalni graf
- Primer čakalnega grafa brez cikla:

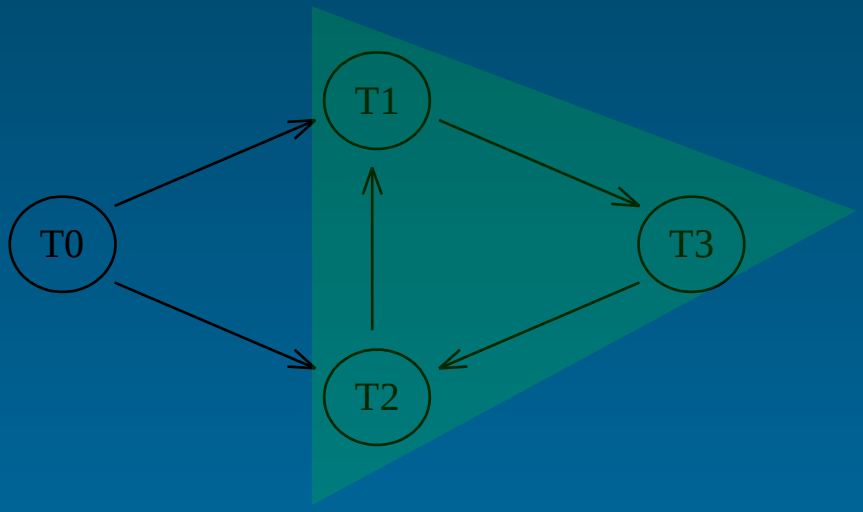


Odpravljanje mrtve zanke..

- **Čakalni graf** je usmerjni graf **G = < V, P >**
- “**V**” predstavlja množico vozlišč, v kateri vsak predstavlja po eno aktivno transakcijo T_i
- “P” predstavlja množico usmerjenih povezav (T_i, T_j), ki predstavljajo čakanje transakcije T_i na odobritev zaseženja podatka, ki ga ima zaseženega transakcija T_j
- Mrtva zanka se v grafu kaže kot cikel. V njem se lahko nahajata dve ali več transakcij, ki druga drugo čakajo na sprostitev zaseženj

Odpravljanje mrtve zanke..

- Primer mrtve zanke, v kateri se nahajajo tri transakcije: T1, T2, T3.



- Za odpravljanje mrtve zanke je zadolžen SUPB, ki ves čas vzdržuje čakalni graf
- Spremembe v grafu nastopijo:
  - ob neodobrenih zahtevah po zaseženjih in
  - ob zaključkih transakcij.

Odpravljanje mrtve zanke..

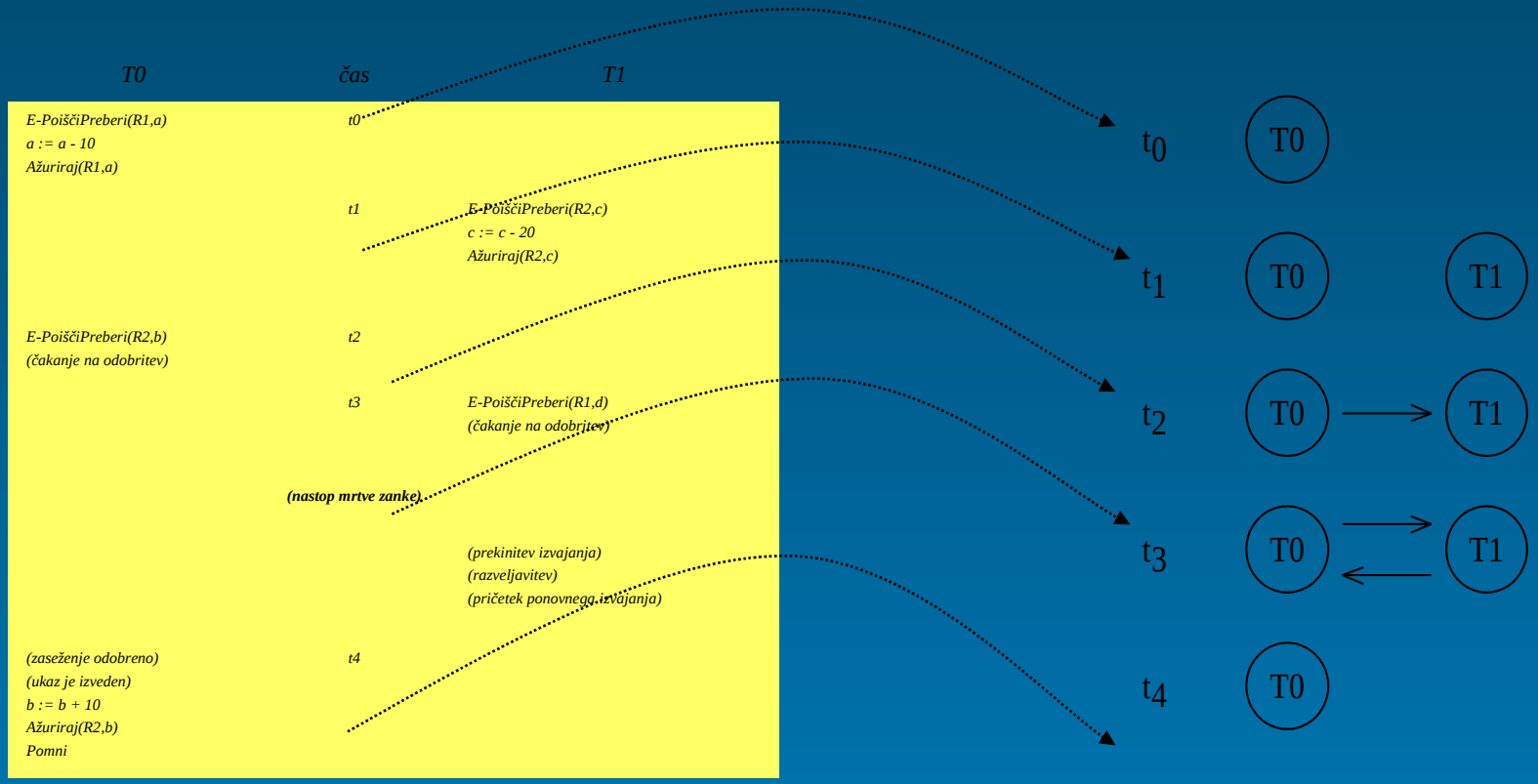
- Ko SUPB zasledi v grafu cikel, se mora lotiti odpravljanja mrtve zanke
- Odpravi jo tako, da eno izmed transakcij v mrtvi zanki izbere za žrtev in prekine njeno izvajanje
- S tem, ko se ena izmed transakcij v mrtvi zanki prekine, izgine iz čakalnega grafa tudi njeno vozlišče in vse pripadajoče povezave, kar pomeni, da cikla več
- Po razveljavitvi ažuriranj prekinjene transakcije, eni izmed preostalih transakcij odobri zahtevano zaseženje, prekinjena transakcija pa se vrne v ponovno izvajanje

Odpravljanje mrtve zanke..

- SUPB izbere žrtev preko enega od naslednjih kriterijev:
  - čas izvajanja transakcij ("starost") do nastopa mrtve zanke,
  - predvideni čas izvajanja transakcij do njihovega zaključka,
  - število odobrenih zaseženj,
  - število izvedenih ažuriranj.
- Ne glede na izbrani kriterij je potrebno poskrbeti, da ne bo vedno znova razveljavljena ena in ista transakcija

Odpravljanje mrtve zanke

- Razvoj čakalnega grafa:



Časovno označevanje..

- Postopek za nadzor nad sočasnim izvajanjem transakcij, ki ne temelji na zaseganju, zato predstavlja manjšo dodatno obremenitev SUPB
- Postopek je uporaben tudi pri porazdeljenih PB, kjer ne obstaja centraliziranega nadzora nad izvajanjem transakcij
- Pri časovnem označevanju se podatki ne zasegajo, zato tudi ne more priti do mrtve zanke

Časovno označevanje..

- Pri časovnem označevanju se transakcije izvajajo tako, da je njihov zaporedniški razpored ekvivalenten takemu zaporednemu razporedu, po katerem se najprej izvede starejša transakcija in nato mlajša
- Konfliktne oziroma potencialno konfliktne zahteve po branju ali ažuriranju podatkov se razrešujejo s prekinitvijo in ponovnim izvajanjem transakcije

Časovno označevanje..

- Protokol obsega naslednja pravila:
  - Vsaki transakciji Ti priredi SUPB časovno oznako **to(Ti),** ki je enaka njenemu startnemu času, in enolično označuje posamezno transakcijo.
    - V primeru centralizirane podatkovne baze se dve transakciji tako ne moreta pričeti ob istem času,
    - pri porazdeljenih podatkovnih bazah pa se časovna oznaka kombinira še z oznako lokalnega SUPB, kar nato skupaj omogoča enolično identifikacijo.
  - Transakciji, katere izvajanje se po pravilih protokola prekine, se pri ponovnem izvajanju priredi nova časovna oznaka.
  - Pri izvajanju transakcij se mora kot zaščita pred transakcijskimi nesrečami uporabljati *odloženo ažuriranje*. Vse spremembe, ki jih je transakcija izvedla, se v podatkovni bazi uveljavijo šele ob njenem uspešnem zaključku. Tako je preprečena možnost branja neobstoječega podatka (neobstoječega zapisa).

Časovno označevanje..

- Vsakemu podatku P oz. objektu zaseganja P, priredi SUPB dve časovni oznaki:
  - **tp(P)** je enak časovni oznaki to(Ti) najmlajše transakcije doslej, ki je podatek P uspešno prebrala;
  - **ta(P)** je enak časovni oznaki to(Ti) najmlajše transakcije doslej, ki je podatek P uspešno ažurirala.
- Ob zahtevi transakcije Ti po branju podatka P:
  - če je **to(Ti) < ta(P),** se Ti razveljavi in vrne v ponovno izvajanje;
  - v nasprotnem primeru se branje izvede, časovna oznaka povezana z branjem podatka P pa se spremeni v: **tp(P) = max(tp(P), to(Ti)).**
- Ob zahtevi transakcije Ti po ažuriranju podatka P:
  - če je **to(Ti) < tp(P)** ali **to(Ti) < ta(P),** se Ti razveljavi in vrne v ponovno izvajanje;
  - v nasprotnem primeru se ažuriranje izvede, časovna oznaka povezana z ažuriranjem podatka P, pa se spremeni v: **ta(P) = to(Ti).**

Časovno označevanje..

- Protokol s peto točko zagotavlja, da starejša transakcija ne more prebrati podatka, ki ga je pred tem ažurirala mlajša transakcija. S točko šest pa, da starejša transakcija ne more ažurirati podatka, ki ga je pred tem prebrala ali ažurirala mlajša transakcija
- Transakcija TX se lahko ob nekem branju ali ažuriranju nadaljuje le, če je bilo zadnje ažuriranje (ali branje) tega podatka s strani starejše transakcije TY
- Če transakciji podatek le bereta, je zaporedje branj seveda nepomembno

Časovno označevanje

- Transakcije se ob uporabi časovnega označevanja praviloma prekinjajo pogosteje, kot bi bilo potrebno oz. bi bilo ob uporabi prej obravnavanih protokolov
- Ob konfliktni situaciji se vedno prekine delovanje starejše transakcije
- Daljše transakcije postanejo pogosto prekinjene zaradi ažuriranja, ki ga izvedejo mlajše. Protokol daje implicitno prednost krajšim transakcijam pred daljšimi

Izkušnje iz prakse

- Sodobni SUPB izvajajo avtomatsko zaseganje in avtomatsko odkrivanje mrtve zanke
- S posebnimi kazi dosežemo implicitni E-PoiščiPreberi. Oracle: SELECT … FOR UPDATE

Vaja

Ugotovi, ali je izmenični razpored ukazov transakcij T1 in T2 zaporedniški. Če ni, napiši razlog (detaljno argumentiraj!) in napiši zaporedniški vrstni red izvajanja ukazov transakcij T1 in T2 z uporabo protokola PXC. X, Y in Z so naslovi v podatkovni bazi, x, y in z pa lokalne spremenljivke.

t	T1: Odštej	T2: Prištej
1	Začetek	
2	PoišČiPreberi(X,x)	
3		Začetek
4		PoišČiPreberi (Y,y)
5	x=x-10	
6		y=y+10
7	PoišČiPreberi (Y,y)	
8		PoišČiPreberi (Z,z)
9	y=y-10	
10		z=z+10
11	Ažuriraj(X,x)	
12		Ažuriraj(Z,z)
13	Ažuriraj(Y,y)	
14		Ažuriraj(Y,y)
15	Pomni	
16		Pomni

Razpored ukazov transakcij NI zaporedniški. Razlog: izgubljeno ažuriranje iz vrstice 13 transakcije T1, ki se izgubi z ažuriranjem iz vrstice 14 transakcije T2.

Rešitev z uporabo protokola PXC:

t	T1: Odštej	T2: Prištej
1	Začetek	
2	E-PoišČiPreberi(X,x)	
3		Začetek
4		E-PoišČiPreberi (Y,y)
5	x=x-10	
6		y=y+10
7	E-PoišČiPreberi (Y,y) (čakanje na odobritev)	
8		E-PoišČiPreberi (Z,z)
9		z=z+10
10		Ažuriraj(Z,z)
11		Ažuriraj(Y,y)
12		Pomni (vsa zaseženja se sprostijo)
13	(zaseženje je odobreno, ukaz se izvede)	
14	y=y-10	
15	Ažuriraj(X,x)	
16	Ažuriraj(Y,y)	
17	Pomni	