Napredni modeli i baze podataka

Predavanja
prosinac 2015.

NoSQL
1/3

NoSQL - teme

1. Baze podataka: ACID podsjetnik
2. Uvod
3. Distribuirane baze podataka
4. NoSQL modeli podataka
5. Distribucijski modeli
6. Konzistencija, verzije
7. MapReduce
8. Primjeri
   - Key Value
   - Document
   - Column family
   - Graph DBs

Transakcije i obnova baze podataka

Ova tema se detaljnije radi na predmetima:

- **Baze podataka** http://www.fer.unizg.hr/predmet/bazepod
- **Sustavi baza podataka** http://www.fer.unizg.hr/predmet/sbp

sljedeći slajdovi su preuzeti (i pojednostavljeni)
iz predavanja predmeta Baza podataka

Sustav za upravljanje bazama podataka

- skriva od korisnika detalje fizičke pohrane podataka
- omogućuje definiciju i rukovanje s podacima
- obavlja optimiranje upita

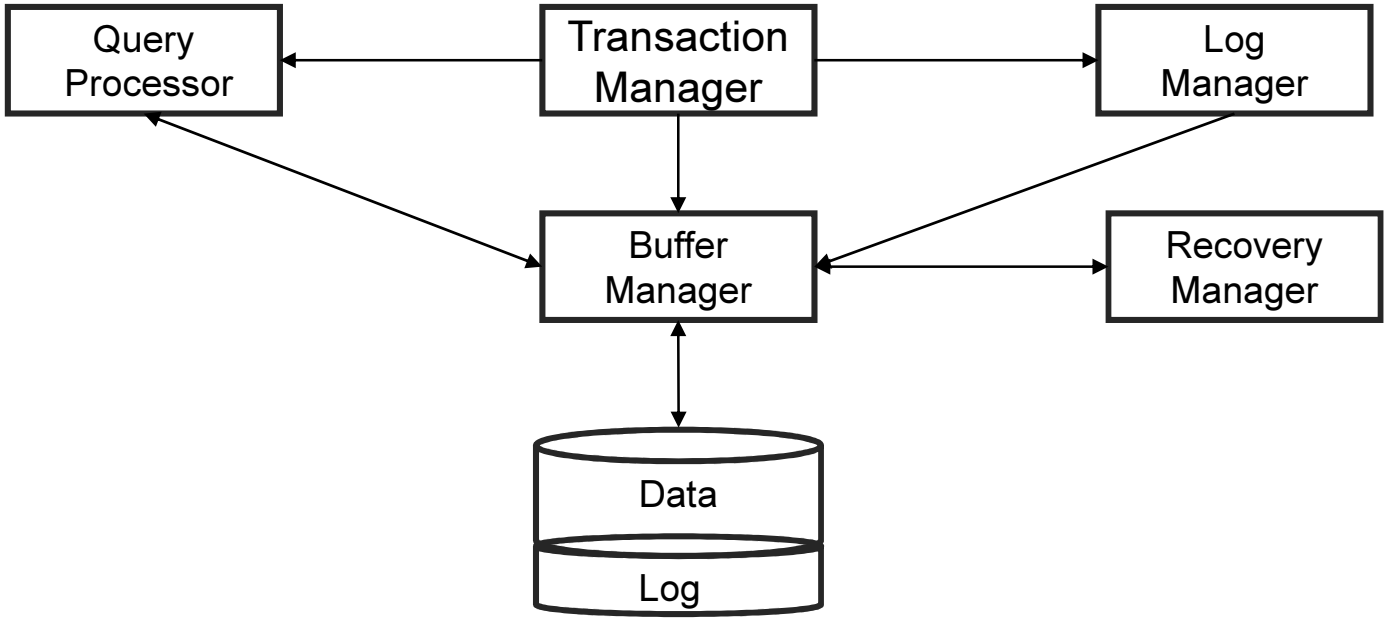
- obavlja funkciju zaštite podataka
  - integritet podataka
  - pristup podacima - autorizacija, sigurnost
  - osigurava potporu za upravljanje transakcijama
    - obnova u slučaju pogreške ili uništenja baze podataka
    - kontrola paralelnog pristupa

TRANSAKCIJA

- jedinica rada nad bazom podataka
- sastoji se od niza logički povezanih izmjena
- početak transakcije - **BEGIN WORK**
- završetak transakcije:
  - **COMMIT WORK** - uspješan završetak - potvrđivanje transakcije
  - **ROLLBACK WORK** - neuspješan završetak - poništavanje transakcije - poništavanje svih izmjena koje je transakcija obavila

Upravljanje transakcijama

- *Transaction Management*
- Upravljač transakcijama (*transaction manager, transaction processing monitor – TP monitor*) - dio sustava koji brine o obavljanju transakcija i osigurava zadovoljavanje svih poznatih pravila integriteta.



Terminologija

- *Transaction Management*
- *Transaction Manager*
- *Query Processor*
- *Log*
- *Log Manager*
- *Recovery Manager*
- *Buffer Manager*

- Upravljanje transakcijama
- Upravljač transakcijama
- Procesor upita
- Dnevnik
- Upravljač dnevnicima
- Upravljač obnovom
- Upravljač međuspremnicima

Primjer transakcije

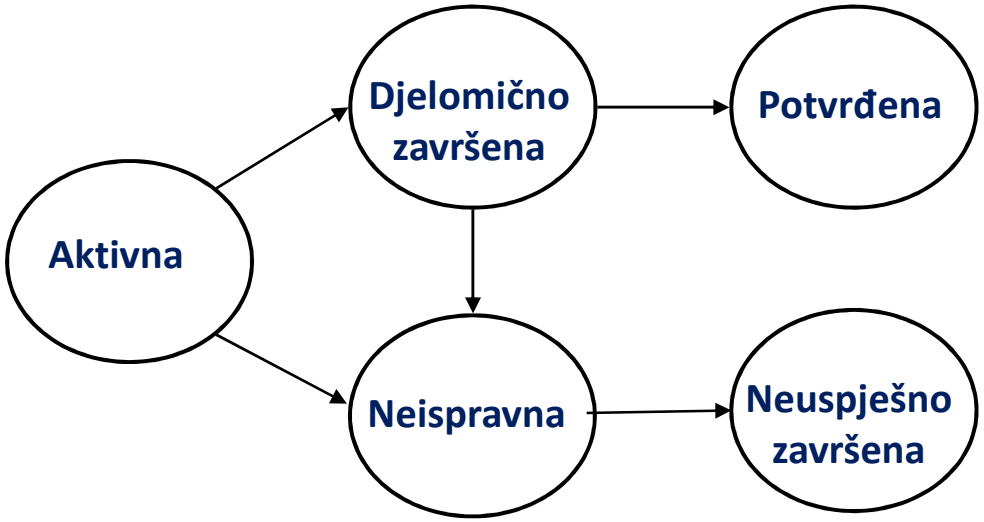
```
CREATE PROCEDURE prijenos (s_racuna INTEGER
                         , na_racun INTEGER
                         , iznos DECIMAL (8,2))
   DEFINE pom_saldo DECIMAL (8,2);
   BEGIN WORK;
     UPDATE racun SET saldo = saldo - iznos
        WHERE br_racun = s_racuna;
     UPDATE racun SET saldo = saldo + iznos
        WHERE br_racun = na_racun;
     SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
     IF pom_saldo < 0 THEN
        ROLLBACK WORK;
     ELSE
        COMMIT WORK;
     END IF
END PROCEDURE
```

Implicitne granice transakcija

- Ako granice transakcije nisu eksplicitno definirane naredbama BEGIN/COMMIT/ROLLBACK, tada se granice transakcije određuju implicitno:
  - svaka SQL naredba se smatra transakcijom za sebe
    - naročito važno: UPDATE, DELETE, INSERT u slučajevima kada djeluju nad skupom n-torki
- Neki SUBP-ovi (npr. Oracle, SQL Server, …) podržavaju način rada u kojem nije potrebno eksplicitno zadati početak transakcije
  - tada se početkom transakcije smatra prva naredba izvedena u okviru sjednice
  - potrebno je eksplicitno zadati COMMIT ili ROLLBACK - nakon čega opet implicitno počinje nova transakcija

Stanja transakcije

- Aktivna (*active*) – tijekom izvođenja
- Djelomično završena – (*partially committed*) – nakon što je obavljena njezina posljednja operacija
- Neispravna (*failed*) – nakon što se ustanovi da nije moguće nastaviti njezino normalno izvođenje
- Neuspješno završena (*aborted*) – nakon što su poništeni njezini efekti i baza podataka vraćena u stanje kakvo je bilo prije nego što je započela
- Potvrđena (*committed*) – uspješno završena



Dijagram stanja transakcije

Potvrđivanje transakcije

- Točka potvrđivanja (*commit point*) – trenutak u kojem sve izmjene koje je transakcija napravila postaju trajne
- Sve izmjene koje je transakcija načinila prije točke potvrđivanja mogu se smatrati tentativnima (*tentative* = privremeno, provizorno)
- U točki potvrđivanja otpuštaju se svi ključevi
- Potvrđena izmjena nikad ne može biti poništena - sustav garantira da će njezine izmjene biti trajno pohranjene u bazi podataka, čak i ako kvar nastane neposredno nakon njezinog potvrđivanja

11

Svojstva transakcije - ACID

- ***Atomicity*** - nedjeljivost transakcije (atomarnost) - transakcija se mora obaviti u cijelosti ili se uopće ne smije obaviti
- ***Consistency*** - konzistentnost - transakcijom baza podataka prelazi iz jednog konzistentnog stanja u drugo konzistentno stanje
- ***Isolation*** - izolacija - kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge
- ***Durability*** - izdržljivost - ako je transakcija obavila svoj posao, njezini efekti ne smiju biti izgubljeni ako se dogodi kvar sustava, čak i u situaciji kada se kvar desi neposredno nakon završetka transakcije

Atomarnost

```
CREATE PROCEDURE prijenos (s_racuna INTEGER, na_racun INTEGER
                         , iznos DECIMAL (8,2))
   DEFINE pom_saldo DECIMAL (8,2);
   BEGIN WORK;
     UPDATE racun SET saldo = saldo – iznos
        WHERE br_racun = s_racuna;
     UPDATE racun SET saldo = saldo + iznos
        WHERE br_racun = na_racun;
     SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
     ...
```

Kvar sustava

- Kvar se dogodio za vrijeme obavljanja druge UPDATE naredbe
  - sustav mora osigurati poništavanje efekata prve UPDATE naredbe!
- Sa stanovišta krajnjeg korisnika transakcija je **nedjeljiva**
  - nije bitno što se moraju obaviti dvije ili više zasebnih operacija nad bazom podataka
- Korisnik mora biti siguran da je zadatak **obavljen potpuno i samo jednom** (ili ništa nije obavljeno)

Durability

```
...
BEGIN WORK;
  UPDATE racun SET saldo = saldo - iznos
     WHERE br_racun = s_racuna;
  UPDATE racun SET saldo = saldo + iznos
     WHERE br_racun = na_racun;
  SELECT saldo INTO pom_saldo FROM racun
     WHERE br_racun = s_racuna;
  IF pom_saldo < 0 THEN
     ROLLBACK WORK;
  ELSE
     COMMIT WORK;

  END IF
```

Kvar sustava

- Kvar se dogodio nakon potvrđivanja transakcije
  - efekti transakcije ne smiju biti izgubljeni
- Bez obzira u kojem se trenutku nakon potvrđivanja transakcije dogodio kvar, sustav mora osigurati da su njezini efekti trajno pohranjeni

OBNOVA BAZE PODATAKA (Database Recovery)

- Dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno

- Velike baze podataka – dijeljene, višekorisničke – nužno moraju posjedovati mehanizme obnove
- Male, jednokorisničke baze podataka obično imaju malu ili uopće nemaju potporu obnovi – obnova se prepušta korisnikovoj odgovornosti – podrazumijeva se da korisnik periodički stvara arhivsku (*backup*) kopiju pomoću koje u slučaju potrebe obnavlja bazu podataka

Uzroci pogrešaka

- pogreške opreme
- pogreške operacijskog sustava
- pogreške sustava za upravljanje bazama podataka
- Pogreške aplikacijskog programa
- pogreške operatera
- kolebanje izvora energije
- požar, sabotaža, ...

Općenito pravilo koje omogućuje obnovu

- **Redundancija** - svaki se podatak mora moći rekonstruirati iz nekih drugih informacija redundantno pohranjenih negdje drugdje u sustavu (na traci, na drugom disku, na zrcalnom disku, ...)
- Redundancija se postiže:
  - dnevnicima izmjena (*logical log*) koji služe za:
    - poništavanje transakcija
    - ponovno obavljanje transakcija
  - zrcaljenjem podataka (*mirroring*)
  - sigurnosnim kopijama (*backup*)

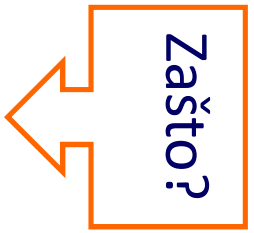
Općeniti opis postupka koji omogućuje obnovu

❶ Periodičko kopiranje sadržaja baze podataka na arhivski medij (drugi disk, diskovni automat (*jukebox*) specijaliziranih sustava za sigurnosne kopije; nekad su to bile mag. trake)

(1 × dnevno, 1 × tjedno - ovisno o učestalosti promjena)

❷ Svaka izmjena u bazi podataka evidentira se u logičkom dnevniku izmjena (*logical log, journal, transaction log*)

- stara vrijednost zapisa, nova vrijednost zapisa
- korisnik, vrijeme, …
- ***write-ahead log rule:*** izmjena se prvo zapisuje u dnevnik, a tek se onda provodi!



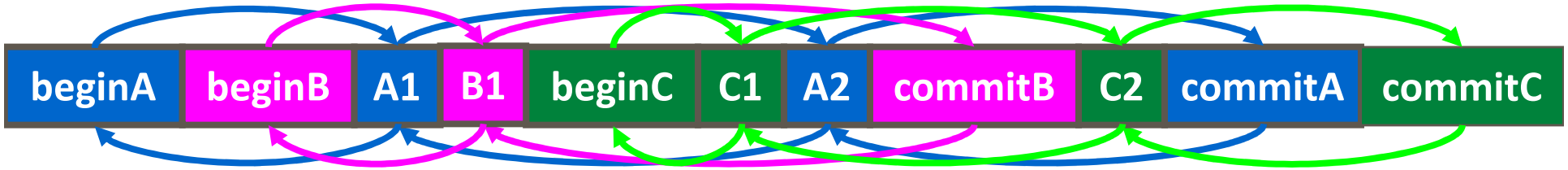
- Dnevnici izmjena omogućuju
  - **poništavanje** transakcija (važno radi svojstva **nedjeljivosti**)
  - **ponovno obavljanje** transakcija (važno radi svojstva **izdržljivosti**)

Kad nastane kvar …

- Ako je baza je potpuno uništena:
  - Učitava se najsvježija arhivska kopija (naredbom ROLLFORWARD DATABASE) – time se baza podataka dovodi u stanje kakvo je bilo u trenutku kad je napravljena posljednja arhivska kopija
  - Koristeći dnevnik izmjena ponovno se obavljaju izmjene koje su dogodile u međuvremenu - nakon izrade arhive
- Baza nije uništena - sadržaj je nepouzdan - program je prekinut tijekom obavljanja niza logički povezanih izmjena – potrebno je vratiti bazu podataka u ispravno stanje
  - pomoću podataka sadržanih u dnevniku izmjena poništavaju se sve izmjene koje su načinile nezavršene transakcije

Dnevnik izmjena

Transakcija A	beginA A1 A2 commitA
Transakcija B	beginB B1 commitB
Transakcija C	beginC C1 C2 commitC



Tipovi pogrešaka

❶ Pogreške transakcija (*transaction failure)* - pogreške koje su posljedica neplaniranog prekida transakcije

❷ Pogreška računalskog sustava (*system failure*) - baza podataka nije fizički uništena

❸ Kvar medija za pohranu (*media failure*) - baza podataka je fizički uništena

---

Slučaj ❶ - pomoću dnevnika izmjena poništavaju se efekti transakcije, kao da transakcija nikada nije započela s radom

Slučaj ❷ - transakcije koje su se obavljale u trenutku prekida se nakon ponovnog pokretanja poništavaju – koriste se **kontrolne točke** (eng. ***checkpoint***)

Slučaj ❸ - baza podataka se obnavlja pomoću arhivske kopije i pripadnog dnevnika izmjena

Ostali mehanizmi obnove

- Zrcaljenje podataka (*mirroring*)
- Arhiviranje tijekom obavljanja transakcija (*on-line backup*)
- Inkrementalno arhiviranje (*incremental backup*)

22

Zrcaljenje

- Postoje dva jednaka područja – primarno područje i zrcalno područje
- Promjene se provode istovremeno u primarnom i zrcalnom području
- U slučaju pogreške u jednom od područja
  - ➔ nastavak rada na ispravnom području
  - ➔ nalog za ponovo kreiranje (“popravljanje”) zrcalnog područja
  - ➔ nakon „popravka” područja se sinkroniziraju
- Zrcaljenje se može provoditi pod kontrolom sklopovlja
  - RAID - Redundant Arrays of Independent (Inexpensive) Disks
- Zrcaljenje pod kontrolom SUBP-a – na razini baze podataka određuje se koji dio baze podataka će se zrcaliti

Zrcaljenje na razini baze podataka

- Omogućuje finiju granulaciju zrcaljenog područja
  - mogu se zrcaliti samo dijelovi baze podataka - tablice koje uvijek moraju biti dostupne
- Visoka dostupnost – u slučaju kvara jednog područja dostupni su podaci u drugom
- Skalabilnost - podjela opterećenja

- Zrcaljenje ne može pomoći pri poništavanju transakcija niti kod kvara čitavog sustava
- Zrcaljenje je samo dodatni mehanizam za postizanje visoke dostupnosti
- Ne može zamijeniti arhiviranje i vođenje dnevnika

Arhiviranje tijekom rada korisnika

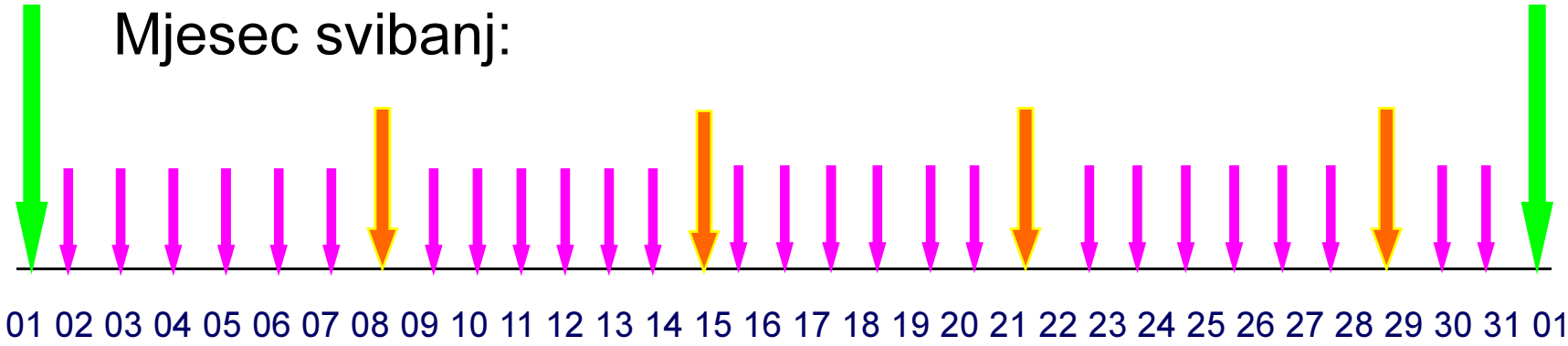
- *On-line backup*
- U klasičnim sustavima arhiviranje se obavljalo na sustavu na kojem nije bilo korisnika (npr. u petak na kraju radnog vremena)
- Današnji sustavi omogućuju da se arhiviranje obavlja tijekom rada korisnika, odnosno izvođenja transakcija
- Stanje baze podataka pohranjeno u arhivskoj kopiji je konzistentno i odgovara stanju kakvo je bilo u bazi podataka u času pokretanja arhiviranja.

Inkrementalno arhiviranje

- Arhiviranje baze podataka dodatno opterećuje sustav i usporava rad korisnika
  - ➔ želi se skratiti trajanje i obim arhiviranja
- Inkrementalno arhiviranje omogućuje stvaranje arhiva različitih razina
- Na primjer (u različitim SUBP-ovima se koristi različita terminologija):
- razina 0 – kopija čitave baze podataka – npr. jednom mjesečno
- razina 1 – tjedna arhiva – sadrži promjene nastale nakon arhive razine 0
- razina 2 – dnevna arhiva - sadrži promjene nastale nakon arhive razine 1

26

... Inkrementalno arhiviranje

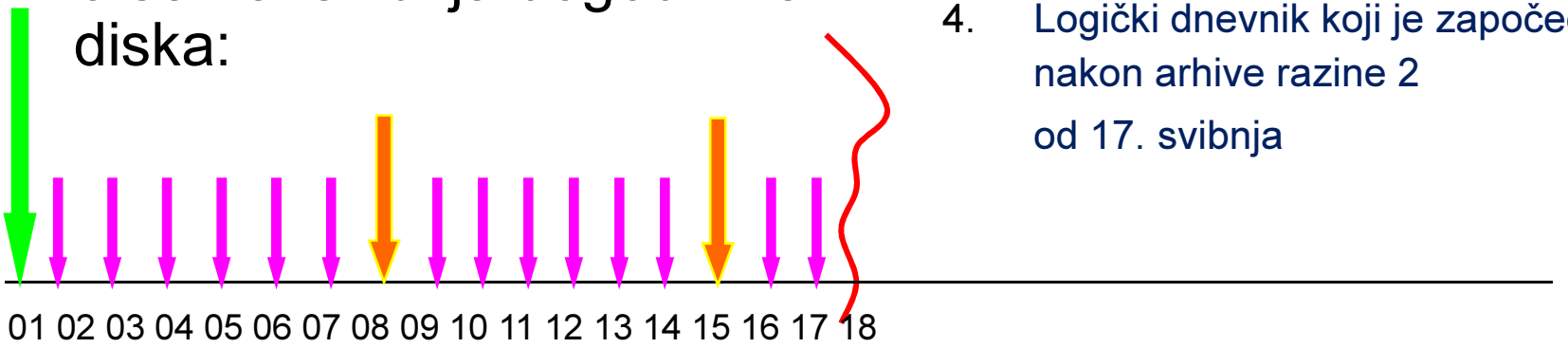


arhiva razine 0

arhiva razine 1

arhiva razine 2

Ako se 18. svibnja dogodi kvar diska:



Pri obnovi se koriste se:

1. Arhiva razine 0 od 1. svibnja
2. Arhiva razine 1 od 15. svibnja
3. Arhiva razine 2 od 17. svibnja
4. Logički dnevnik koji je započeo nakon arhive razine 2 od 17. svibnja

NoSQL - Uvod

Uvod - povijest

1980
- RBP

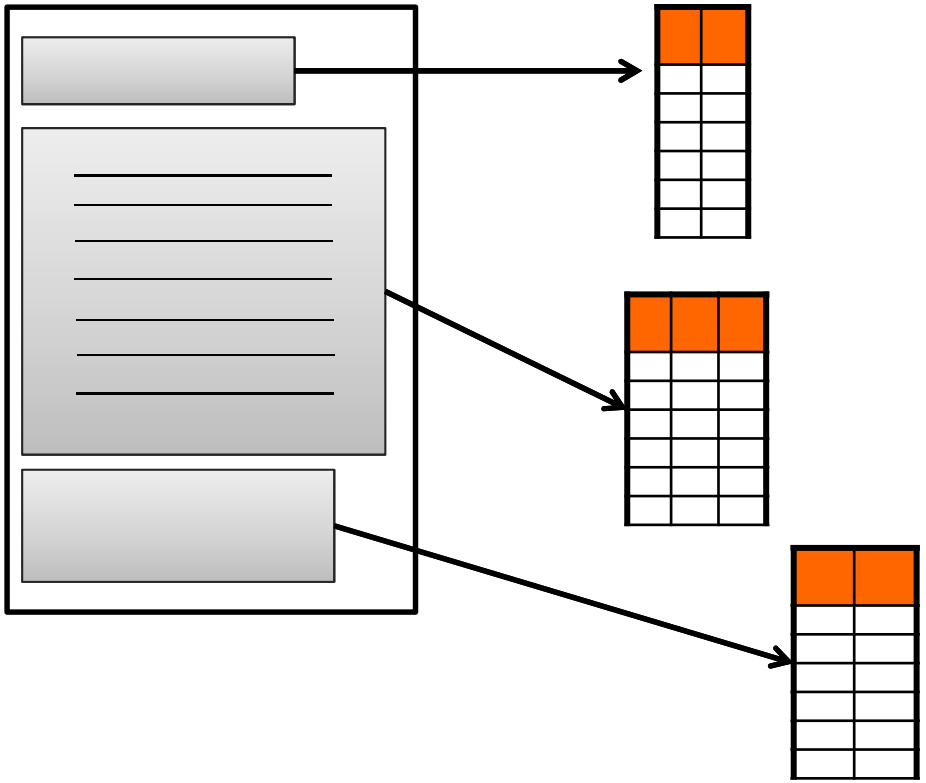
1990
- OOBP

2000
- i dalje RBP
- Web -> NoSQL

2010
- RDB, NoSQL

"HIC SVNT DRACONES" ☺

✕ *Impedance mismatch*



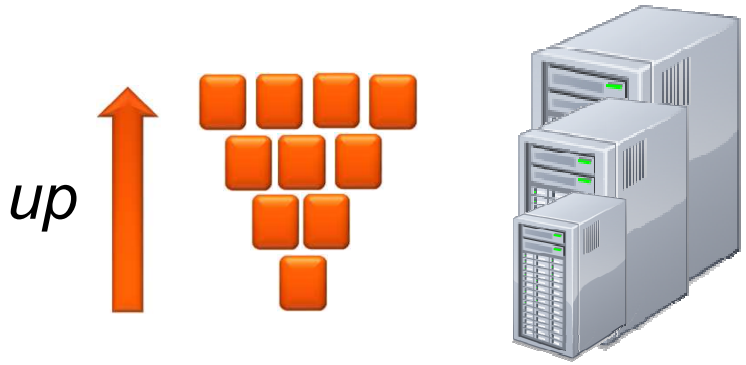
Uvod - relacijske baze podataka

- Relacijske baze podataka
  - ✓ Perzistencija
  - ✓ Istodobni pristup, ACID
  - ✓ Integracija
  - ✓ Standardni model podataka, upitni jezik
  - ✗ *Impedance mismatch*
  - ✓✗ Application vs Integration databases
  - ✗ Velike količine podataka - skalabilnost
  - ✗ Dostupnost

IEC prefix		Representations				Customary prefix	
Name	Symbol	Base 2	Base 1024	Value	Base 10	Name	Symbol
kibi	Ki	2^{10}	1024^1	1 024	$\approx 1.02 \times 10^3$	kilo	k, K
mebi	Mi	2^{20}	1024^2	1 048 576	$\approx 1.05 \times 10^6$	mega	M
gibi	Gi	2^{30}	1024^3	1 073 741 824	$\approx 1.07 \times 10^9$	giga	G
tebi	Ti	2^{40}	1024^4	1 099 511 627 776	$\approx 1.10 \times 10^{12}$	tera	T
pebi	Pi	2^{50}	1024^5	1 125 899 906 842 624	$\approx 1.13 \times 10^{15}$	peta	P
exbi	Ei	2^{60}	1024^6	1 152 921 504 606 846 976	$\approx 1.15 \times 10^{18}$	exa	E
zebi	Zi	2^{70}	1024^7	1 180 591 620 717 411 303 424	$\approx 1.18 \times 10^{21}$	zetta	Z
yobi	Yi	2^{80}	1024^8	1 208 925 819 614 629 174 706 176	$\approx 1.21 \times 10^{24}$	yotta	Y

Uvod - skalabilnost

- Mogućnost sustava da se nosi s rastućom količinom podataka
- Zadržavanje prihvatljivih performansi
- **Vertikalna** skalabilnost (engl. ***scale up***):
  - Dodavanje resursa jednom čvoru (memorija, procesor, procesi, …)



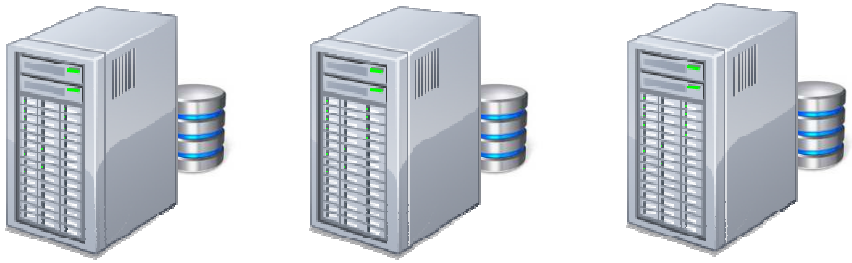
- **Horizontalna** skalabilnost (engl. ***scale out***)
  - Dodavanje čvorova u sustav

Relacijski SUBP-ovi imaju problema s horizontalnom skalabilnošću

Uvod - dostupnost (eng. *availability*), klasteri

- RBP klasteri

- Mi želimo:

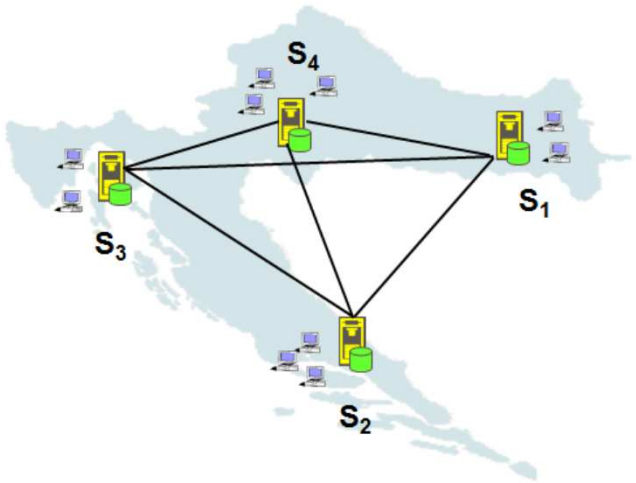


Distribuirane i replicirane relacijske baze podataka

Ova tema se puno detaljnije radi na predmetu
Sustavi baza podataka
http://www.fer.unizg.hr/predmet/sbp
sljedeći slajdovi su preuzeti (i pojednostavljeni)
iz predavanja Sustava baza podataka

Distribuirana baza podataka i distribuirani SUBP

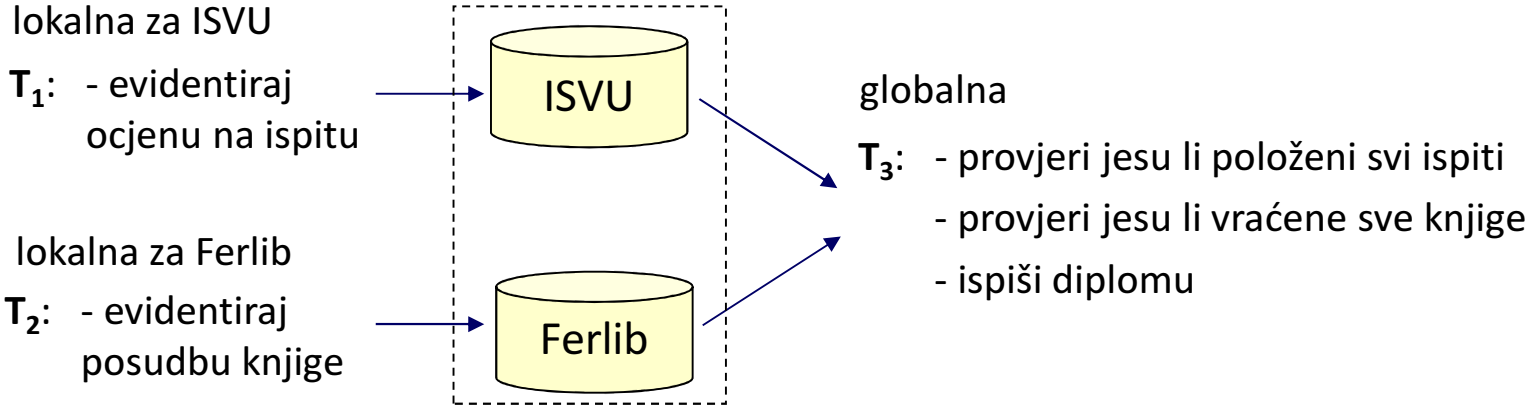
- **Distribuirana baza podataka** (DBP) je skup logički povezanih baza podataka razmještenih **u različitim čvorovima** računalne mreže (LAN, MAN, WAN)
- **Distribuirani sustav za upravljanje bazama podataka** (DSUBP) je programski sustav koji upravlja distribuiranom bazom podataka na takav način da je distribuiranost sustava transparentna prema korisnicima



- DSUBP obuhvaća n lokalnih SUBP-ova.
- Svaki lokalni SUBP, označen sa S_i, (i = 1, ..., n) predstavlja jedan čvor (*site, node*) distribuiranog sustava
- Svaki čvor S_i može direktno ili indirektno komunicirati sa svakim čvorom S_j, tj. postoji dvosmjerna veza između svaka dva čvora
- Čvorovi distribuiranog sustava za upravljanje bazama podataka ne dijele zajedničke fizičke komponente (disk, memorija, procesor)

Distribuirana baza podataka i distribuirani SUBP

- Čvorovi su sposobni obavljati transakcije koje zahtijevaju isključivo lokalni pristup podacima (lokalne transakcije), ali također i transakcije koje zahtijevaju pristup podacima iz različitih čvorova (globalne transakcije)
  - čvorovi posjeduju određeni stupanj lokalne autonomije
- lokalne aplikacije (transakcije)
- globalne aplikacije (transakcije)
- baza podataka je distribuirana ako podržava barem jednu globalnu aplikaciju



Oblikovanje distribuirane baze podataka

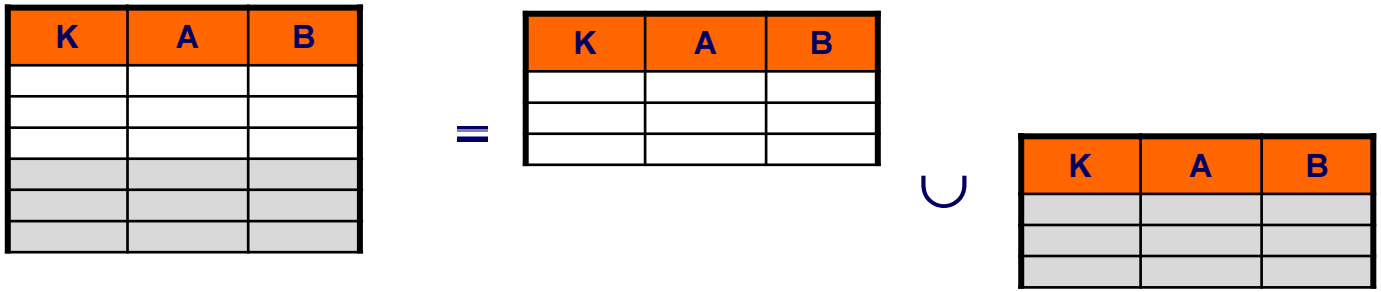
- Važan dio postupka oblikovanja distribuirane baze podataka jest određivanje načina distribucije podataka.
- podaci se smještaju u čvorove u kojima se najčešće koriste
  - minimalizira se mrežni promet

oblikovanje distribucije = fragmentacija + alokacija

- Shema **fragmentacije**
  - podjela baze podataka u **disjunktni** skup fragmenata koji obuhvaćaju sve podatke u bazi podataka uz zadovoljenje pravila da se baza podataka može rekonstruirati iz tih fragmenata bez gubitka informacije
  - relacije mogu biti razdijeljene u fragmente horizontalno ili vertikalno (ili horizontalno i vertikalno)
- Shema **alokacije**
  - shema kojom se opisuje koji je fragment pridružen kojem čvoru distribuiranog sustava

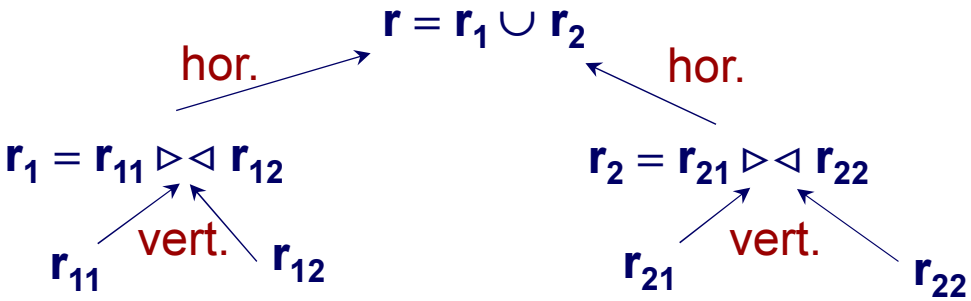
Fragmentacija

- Horizontalna, npr. dva fragmenta



- Vertikalna, npr. dva fragmenta

- Hibridna



Alokacija

Stupanj replikacije fragmenta

- broj čvorova u kojima je fragment alociran

- Svaki fragment mora biti alociran u barem jednom čvoru!

- **Particionirana (ili nereplicirana) baza podataka**
  - svaki od fragmenata alociran je u točno jednom čvoru, tj. stupanj replikacije svakog fragmenta = 1
- **Potpuno replicirana baza podataka**
  - svaki od fragmenata alociran je u svim čvorovima - svaki čvor sadrži repliku cijele baze podataka, tj. stupanj replikacije svakog fragmenta = n (broj čvorova u DSUBP)
- **Parcijalno replicirana baza podataka**
  - baza podataka nije niti particionirana niti potpuno replicirana (svaki od fragmenata može biti alociran u jednom, više ili svim čvorovima)

Primjer alokacije

- Čvorovi S_1, S_2, S_3
- Fragmenti:
  - $student_1, student_2, student_3$
  - $fakultet_1$

$$student_1 = \sigma_{sifFakultet = 36}(student)$$
$$student_2 = \sigma_{sifFakultet = 102}(student)$$
$$student_3 = \sigma_{sifFakultet = 81}(student)$$

parcijalno replicirana baza podataka

S_2

- $student_2$
- $fakultet_1$

S_1

- $student_1$
- $fakultet_1$

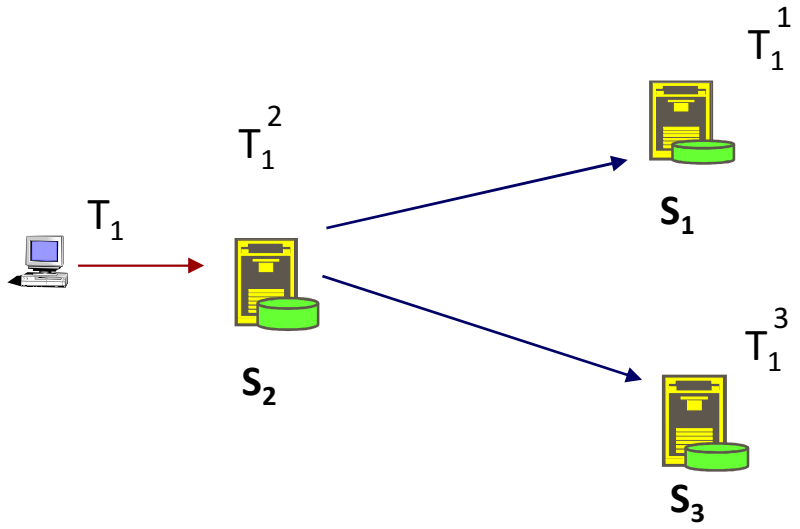
S_3

- $student_3$
- $fakultet_1$

Globalne transakcije, subtransakcije i lokalne transakcije

Primjer: čvorovi DSUBP-a: S_1, S_2, S_3

- Korisnik inicira globalnu transakciju T_1 u čvoru S_2
- Transakcija T_1 se preslikava u skup subtransakcija: T_1^1, T_1^2, T_1^3
- Svaka od subtransakcija sadrži operacije koje se obavljaju u pripadnom čvoru
- oznake: T_i^j subtransakcija globalne transakcije T_i koja se izvršava u čvoru S_j



Transakcija u DSUBP-u

- U svakom čvoru se nalazi zasebni, potpuno funkcionalan SUBP
- Transakcija se više ne može promatrati kao (samo) niz logički povezanih operacija koje se izvršavaju u jednom SUBP-u
- Globalna transakcija je **skup subtransakcija** koje koordinirano izvršavaju SUBP-ovi u više čvorova i pri tome prevode *distribuiranu* bazu podataka iz jednog u drugo konzistentno stanje
- **ACID?**
  - Svojstvo ***C**onsistency* se relativno jednostavno ostvaruje uobičajenim mehanizmima
  - Svojstvo ***D**urability* osiguravaju SUBP-ovi u čvorovima
    - jer svaki čvor garantira izdržljivost svoje subtransakcije
  - Znatno teži problem: svojstva ***A**tomicity* i ***I**solation*

Atomarnost

- Atomarnost subtransakcija osiguravaju lokalni čvorovi
- Kako osigurati atomarnost globalne transakcije?
  - za vrijeme obavljanja globalne transakcije može se dogoditi prekid veze između jednog ili više čvorova ili se u jednom ili više čvorova može dogoditi kvar
- Atomarnost globalne transakcije znači da SUBP u svim čvorovima u kojima se izvršavaju pripadne subtransakcije moraju donijeti i provesti jednaku odluku o ishodu obavljanja transakcije: **ili su sve** subtransakcije globalne transakcije obavljene, **ili nije niti jedna**
- DSUBP provodi protokol potvrđivanja globalne transakcije
  - 2PC - *two-phase commit* (protokol dvofaznog potvrđivanja)

2PC - neformalni opis (1)

- **u svakom čvoru nalazi se menadžer transakcija (TM)**
  - zadaće jednake onima u centraliziranom sustavu: obnova, izolacija, ...
  - razlika: osim lokalnih transakcija, obavljaju se i subtransakcije koje se izvršavaju na njegovoj lokaciji

- **u svakom čvoru nalazi se koordinator transakcija (TC)**
  - pokreće globalnu transakciju koja ima izvor na njegovoj lokaciji
  - distribuira subtransakcije u odgovarajuće čvorove - daje naloge pojedinim TM da obave subtransakciju
  - upravlja završetkom globalne transakcije koja ima izvor na njegovoj lokaciji na način da pripadne subtransakcije budu potvrđene u svim čvorovima ili poništene u svim čvorovima

2PC - neformalni opis (2)

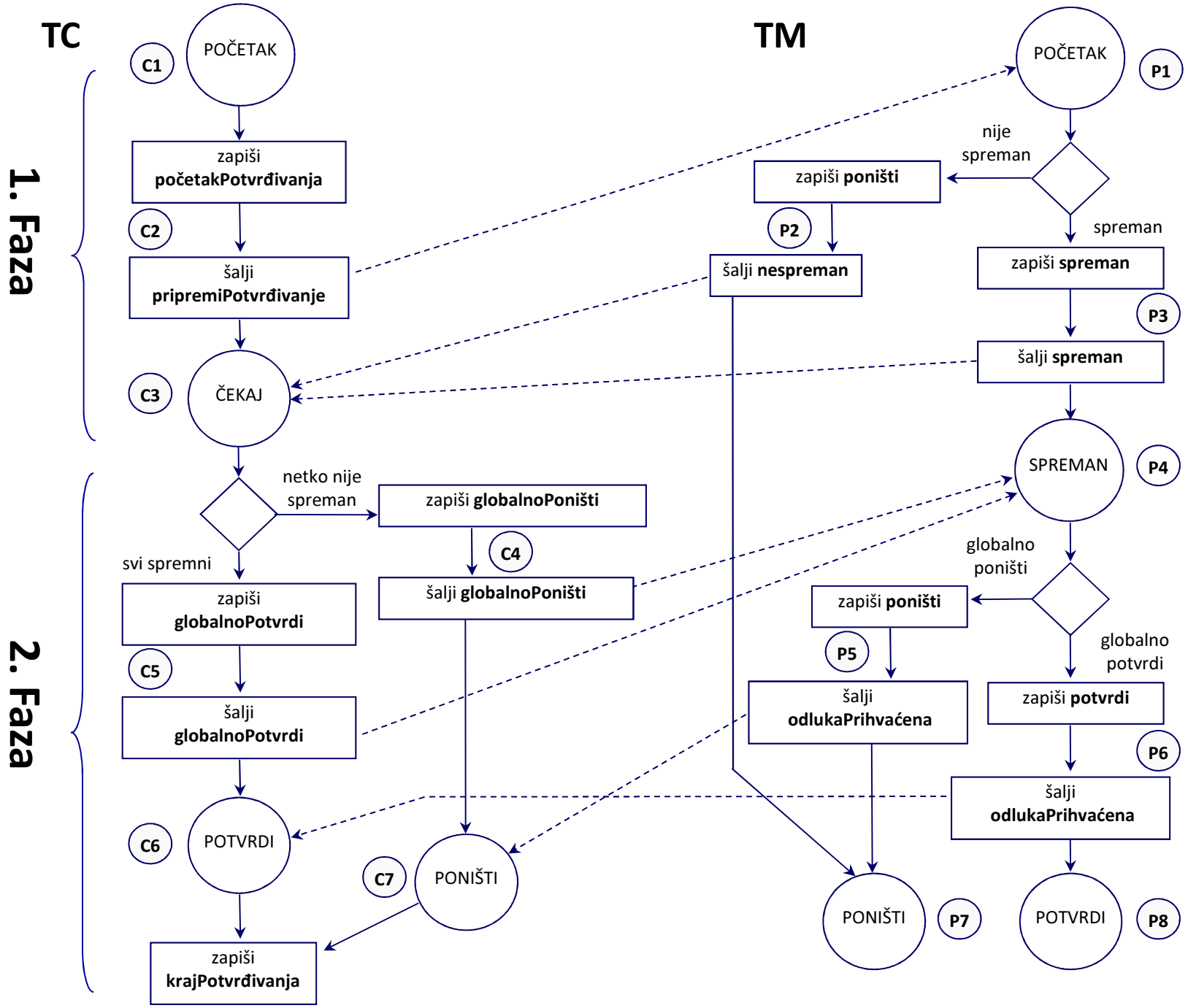
- TC koji se nalazi u izvornom čvoru transakcije distribuira subtransakcije prema odgovarajućim TM-ovima
- nakon obavljanja subtransakcija, svi TM-ovi izvješćuju TC o uspješnom obavljanju operacija subtransakcija. **Tek tada započinje 2PC!**

1. FAZA

- TC šalje svim TM poruku *pripremiPotvrđivanje*. Svaki pojedini TM odgovara *spreman* ili *nespreman* ili ne odgovara

2. FAZA

- ako je sa svih lokacija stigla poruka *spreman* → TC odluku upisuje u svoj dnevnik i svim TM šalje poruku *globalnoPotvrdi*
- u slučaju da je neki od TM odgovorio *nespreman* ili se tijekom zadanog vremena neki od TM nije odazvao → TC odluku upisuje u dnevnik i svim TM šalje poruku *globalnoPoništi*
- TM zapisuju odluku TC-a u svoj dnevnik, prema TC-u šalju potvrdu o prihvaćanju odluke i potvrđuju ili poništavaju subtransakcije
- kada TC dobije potvrde svih TM, u logički dnevnik upisuje oznaku *krajPotvrđivanja*


TC
TM
1. Faza
2. Faza
C1
POČETAK
zapiši
početakPotvrđivanja
C2
šalji
pripremiPotvrđivanje
C3
ČEKAJ
netko nije
spreman
zapiši globalnoPoništi
C4
šalji globalnoPoništi
svi spremni
zapiši
globalnoPotvrdi
C5
šalji
globalnoPotvrdi
C6
POTVRDI
C7
PONIŠTI
zapiši
krajPotvrđivanja
POČETAK
P1
nije
spreman
zapiši poništi
P2
šalji nespreman
spreman
zapiši spreman
P3
šalji spreman
SPREMAN
P4
globalno
poništi
zapiši poništi
P5
šalji
odlukaPrihvaćena
globalno
potvrdi
zapiši potvrdi
P6
šalji
odlukaPrihvaćena
PONIŠTI
P7
POTVRDI
P8

2PC - opis protokola

- ako zbog kvara u mreži ili kvara udaljenog čvora poruka ne stigne u unaprijed zadanom vremenu (*timeout*), TC ili TM nastoje nastaviti s obavljanjem operacija u cilju izbjegavanja blokiranja transakcije

C3:	TC čeka odluku jednog ili više TM. TC može donijeti odluku o poništavanju globalne transakcije
C6, C7:	TC ne može zaključiti jesu li svi TM proveli odluku o potvrđivanju ili poništenju subtransakcije. TC uzastopno proziva TM čije poruke o prihvaćanju odluke nije zaprimio
P1:	TM očekuje poruku od TC o početku pripreme za potvrđivanje. TM može nakon isteka *timeout* unilateralno poništiti subtransakciju, a ako TC nakon toga pošalje poruku *pripremiPotvrđivanje*, TM šalje poruku *nespreman*
P4:	TM je poslao poruku *spreman*, ali nije mu poznata konačna odluka TC-a. TM mora čekati ponovnu uspostavu komunikacije s TC

2PC - opis protokola - kvar u TC-u

- TC koji tijekom obnove pomoću zapisa iz dnevnika utvrdi da je u trenutku kvara sudjelovao u 2PC protokolu, ovisno o trenutku u kojem se dogodio kvar obavlja sljedeće akcije:

C1:	nakon obnove, TC može ponovo započeti 2PC protokol na uobičajeni način
C2, C3:	TC je prekinuo rad nakon što je u dnevnik zapisao *početakPotvrđivanja*. Nakon obnove nastavit će obavljanje protokola ponovnim slanjem poruke *pripremiPotvrđivanje*
C4, C5, C6, C7:	TC je prekinuo rad nakon što je u dnevnik zapisao *globalnoPotvrdi* ili *globalnoPoništi*. Nakon obnove, ponovo će poslati odgovarajuće poruke prema TM

2PC - opis protokola - kvar u TM-u

- TM koji tijekom obnove pomoću zapisa iz dnevnika utvrdi da je u trenutku kvara sudjelovao u 2PC protokolu, ovisno o trenutku u kojem se dogodio kvar obavlja sljedeće akcije:

P1:	TM je prekinuo rad prije nego je u dnevnik zapisao *poništi* ili *spreman*. Tijekom obnove TM unilateralno poništava transakciju
P2:	TM je prekinuo rad nakon što je u dnevnik zapisao *poništi*. TM poništava subtransakciju i TC-u prepušta da nakon isteka zadanog vremena čekanja globalno poništi transakciju
P3, P4:	TM je prekinuo rad nakon što je u dnevnik zapisao *spreman*. TM šalje prema TC poruku *spreman* i čeka odgovor jer bez TC ne može donijeti konačnu odluku
P5, P6:	TM poznaje sudbinu globalne transakcije i postupa u skladu s time
P7, P8:	TM ne treba poduzeti ništa jer se nalazi u stanju u kojem je završio transakciju

Mogućnost blokiranja protokola

- protokol **je blokirajući ako** postoji mogućnost da ispravan čvor (TC ili TM) neće moći završiti transakciju zbog prekida veze ili kvara nekog drugog čvora

Primjer:

- točka P4 na prethodnoj slici
  - TM je prema TC poslao poruku *spreman* i nalazi se u stanju čekanja na odluku TC o sudbini globalne transakcije. U tom trenutku dogodi se kvar na vezi prema TC (ili kvar sustava u kojem se nalazi TC)
  - TM ne smije unilateralno poništiti lokalnu transakciju jer ne zna kakvu je odluku donio TC (možda je TC svim ostalim čvorovima uspio poslati poruku *globalnoPotvrdi*)
  - TM mora čekati uspostavu veze s TC (ili obnovu sustava u kojem se nalazi TC)

⇒ 2PC protokol je blokirajući protokol

Nezavisnost protokola s obzirom na mogućnost obnove

- protokol **je nezavisan s obzirom na mogućnost obnove ako** svaki čvor (TC ili TM), nakon što se u njemu dogodio kvar sustava (ili medija) može samostalno, bez komunikacije s ostalim čvorovima, donijeti odluku o sudbini svih (sub)transakcija koje su se u tom čvoru odvijale u trenutku kvara

Primjer:

- točka P4 na prethodnoj slici
  - TM je u dnevnik zapisao *spreman* i prema TC poslao poruku *spreman*. U tom trenutku se dogodi kvar sustava na TM
  - kada TM započne obnovu utvrdit će da je u trenutku prekida rada sudjelovao u 2PC protokolu. Bez TC ne može donijeti odluku treba li subtransakciju potvrditi ili poništiti, stoga prema TC šalje poruku *spreman* i čeka odgovor

$\Rightarrow$ 2PC protokol nije nezavisan s obzirom na mogućnost obnove

Kvarovi u DSUBP-u

- Upravljanje kvarovima u DSUBP-u je složenije nego u centraliziranim sustavima
  - centralizirani sustav funkcionira u cijelosti ili ne funkcionira
  - dijelovi DSUBP-a mogu biti u kvaru, a dijelovi nastaviti s radom

- Osim kvarova koji su karakteristični za centralizirane sustave (npr. pogreške programske podrške, sklopovlja, uništenja diska), u DSUBP-u su moguće dodatne vrste kvarova:
  - prestanak rada jednog ili više čvorova
  - gubitak veze među čvorovima
  - gubitak poruka
  - podjela mreže: mreža je podijeljena (particionirana) kad je podijeljena u nekoliko podsustava koji međusobno ne mogu komunicirati. Naročiti problem: čvor S_i ne može utvrditi je li se dogodila podjela mreže ili je neki čvor S_j prestao raditi

Nedostaci DSUBP u odnosu na centralizirani SUBP

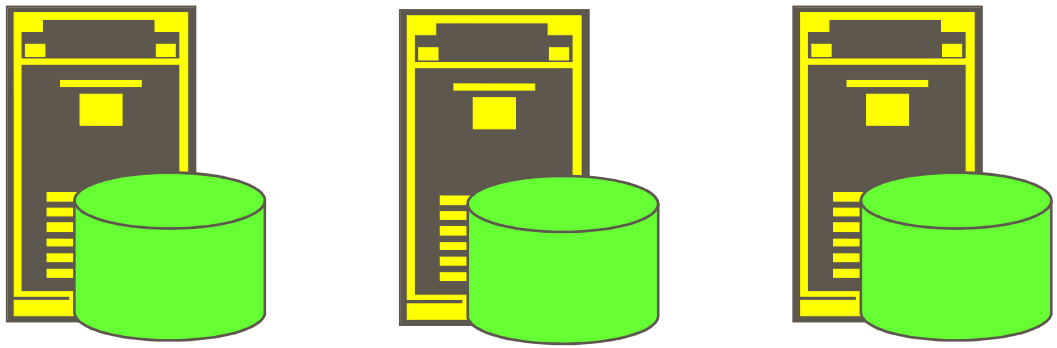
- Bitno veća složenost sustava
- Povećanje troškova, npr.
  - skuplja programska podrška
  - potreba angažiranja većeg broja administratora sustava
- Veći problemi sigurnosti
- Veći troškovi u osiguravanju integriteta podataka
- Nedostatak standarda
- Nedostatak iskustva
- Povećanje složenosti postupka projektiranja baze podataka
- Loša implementacija distribuirane baze podataka može uzrokovati
  - povećanje komunikacijskih troškova
  - smanjenje dostupnosti podataka
  - smanjenje performanci

Funkcionalnost i tehnike DSUBP, koje su rezultat mnogobrojnih provedenih istraživanja, nisu u cijelosti implementirane niti u jednom danas raspoloživom komercijalnom sustavu.

Replicirane baze podataka

Replicirane baze podataka

- fragment je repliciran ako je alociran u dva ili više čvorova
- za jedan logički element x (n-torku, fragment, relaciju) postoji više fizičkih elemenata (kopija, replika), x^1, x^2, ..., u čvorovima S_1, S_2, ...



Prednosti repliciranih BP

- povećanje **dostupnosti**
  - ako je čvor u kojem je pohranjena kopija fragmenta nedostupan, sustav može pristupiti kopiji fragmenta u nekom drugom čvoru
- smanjenje volumena prijenosa podataka
  - podacima koji se često koriste u više čvorova aplikacije mogu pristupati lokalno
- paralelno obavljanje dijelova istog upita
  - upit koji obrađuje fragment može se dekomponirati, te se svaki dio upita može obavljati nad jednom od kopija fragmenta

Nedostaci repliciranih BP

- problem konzistentnosti kopija istog elementa
  - sustav mora osigurati **konzistentnost svih kopija**. Operacija pisanja (unos, brisanje, izmjena) jedne kopije fragmenta mora se propagirati prema svim čvorovima u kojima je taj fragment alociran
  - veći broj operacija koje treba obaviti u većem broju čvorova može uzrokovati smanjenje dostupnosti i povećanje broja potpunih zastoja (pri sinkronoj replikaciji) ili narušavanje konzistentnosti (pri asinkronoj replikaciji)

Sinkroni (eager) protokoli

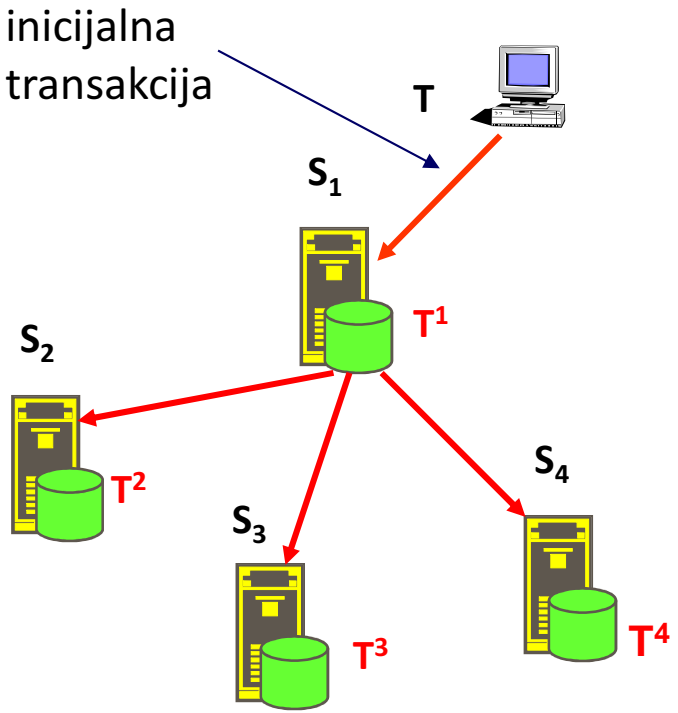
- sve fizičke operacije koje proizlaze iz logičkih operacija inicijalne transakcije obavljaju se unutar granica inicijalne transakcije, tj. sve kopije se moraju izmijeniti u okviru inicijalne transakcije

✓ osigurana **potpuna konzistentnost**

✓ dobre performanse pri čitanju

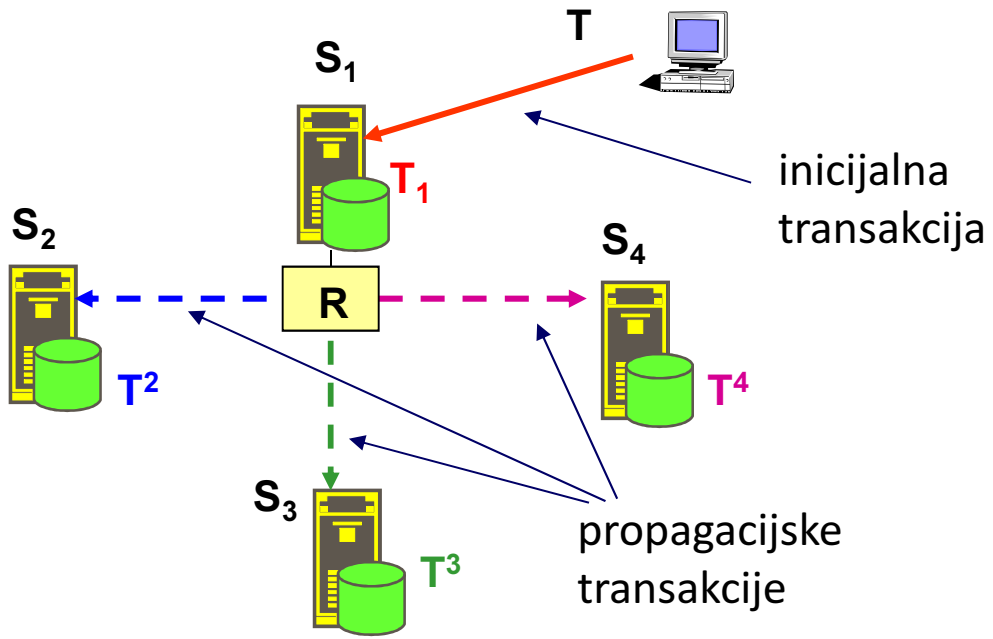
✗ slabije pri izmjeni

✗ produljenje trajanja transakcije, povećanje broja potpunih zastoja, niska dostupnost
(ispadom samo jednog čvora operacije izmjene postaju neizvedive)



Asinkroni (lazy) protokoli

- operacije inicijalne transakcije obavljaju se isključivo u inicijalnom čvoru i niti na koji način ne ovise o komunikaciji s ostalim čvorovima
- inicijalna transakcija može završiti prije nego su obavljene izmjene nad svim kopijama. Izmjene ostalih kopija obavljaju se asinkrono
- visoka dostupnost, dobre performanse
- velika mogućnost pojave nekonzistentnih podataka

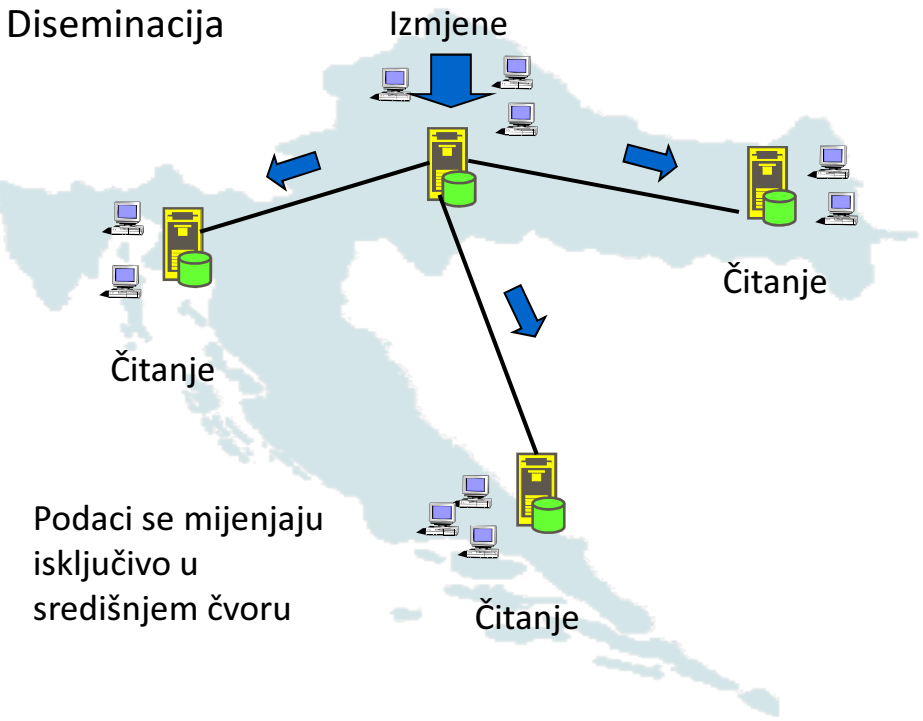


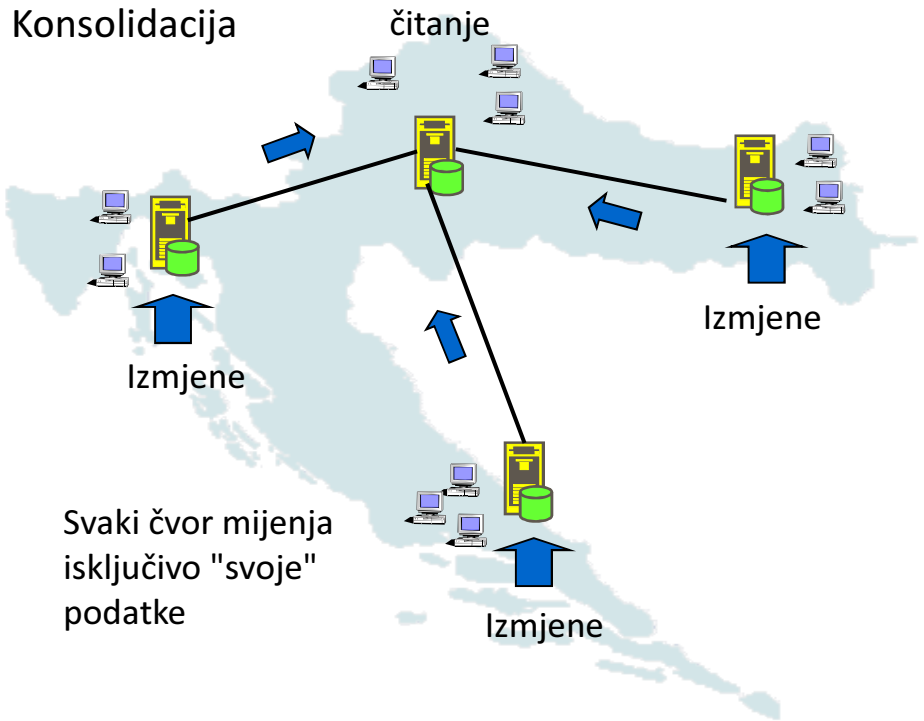
Jednosmjerni protokoli

- Eng. *one-way*, *master-ownership*, *primary-copy*
- za svaki logički element x određuje se samo jedan fizički element x^p koji se proglašava **primarnom kopijom**, te se operacija izmjene koju nad elementom x obavlja inicijalna transakcija mora prvo obaviti nad kopijom x^p
- svaki čvor u kojem je alocirana primarna kopija barem jednog elementa naziva se ravnatelj (*master*)
  - sustav s jednim ravnateljem (*single-master*), u kojem se nalaze sve primarne kopije
  - sustav s više ravnatelja (*multi-master*), u kojem su primarne kopije različitih podataka smještene u različitim čvorovima

Često korišteni oblici jednosmjernih protokola

- 1Master-nSlaves (diseminacija podataka)
  - izmjene se obavljaju u točno jednom nadređenom (master) čvoru i propagiraju prema podređenim čvorovima (slave). Podređeni čvorovi ne smiju obavljati transakcije koje sadrže operacije pisanja
- nMasters-1Slave (konsolidacija podataka)
  - izmjene obavljene u više čvorova propagiraju se prema točno jednom (nadređenom) čvoru. Nadređeni čvor ne smije obavljati transakcije koje sadrže operacije pisanja





Dvosmjerni protokoli

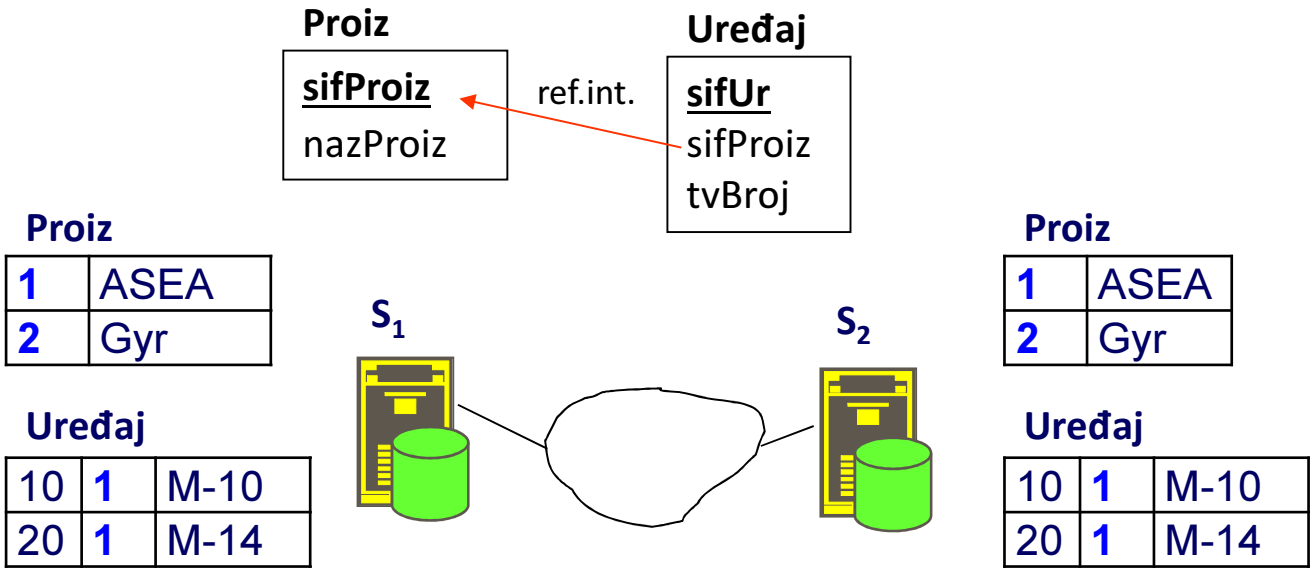
- *two-way, n-way, peer-to-peer, group-ownership, update-anywhere*
- inicijalna transakcija može operaciju izmjene obaviti nad bilo kojom fizičkom kopijom
- dostupnost sustava se bitno povećava u odnosu na jednosmjerne sustave
- koristi li se u kombinaciji s asinkronim protokolom serijalizabilnost izvršavanja transakcija se ne može garantirati

Važni nedostaci dvosmjernih asinkronih protokola

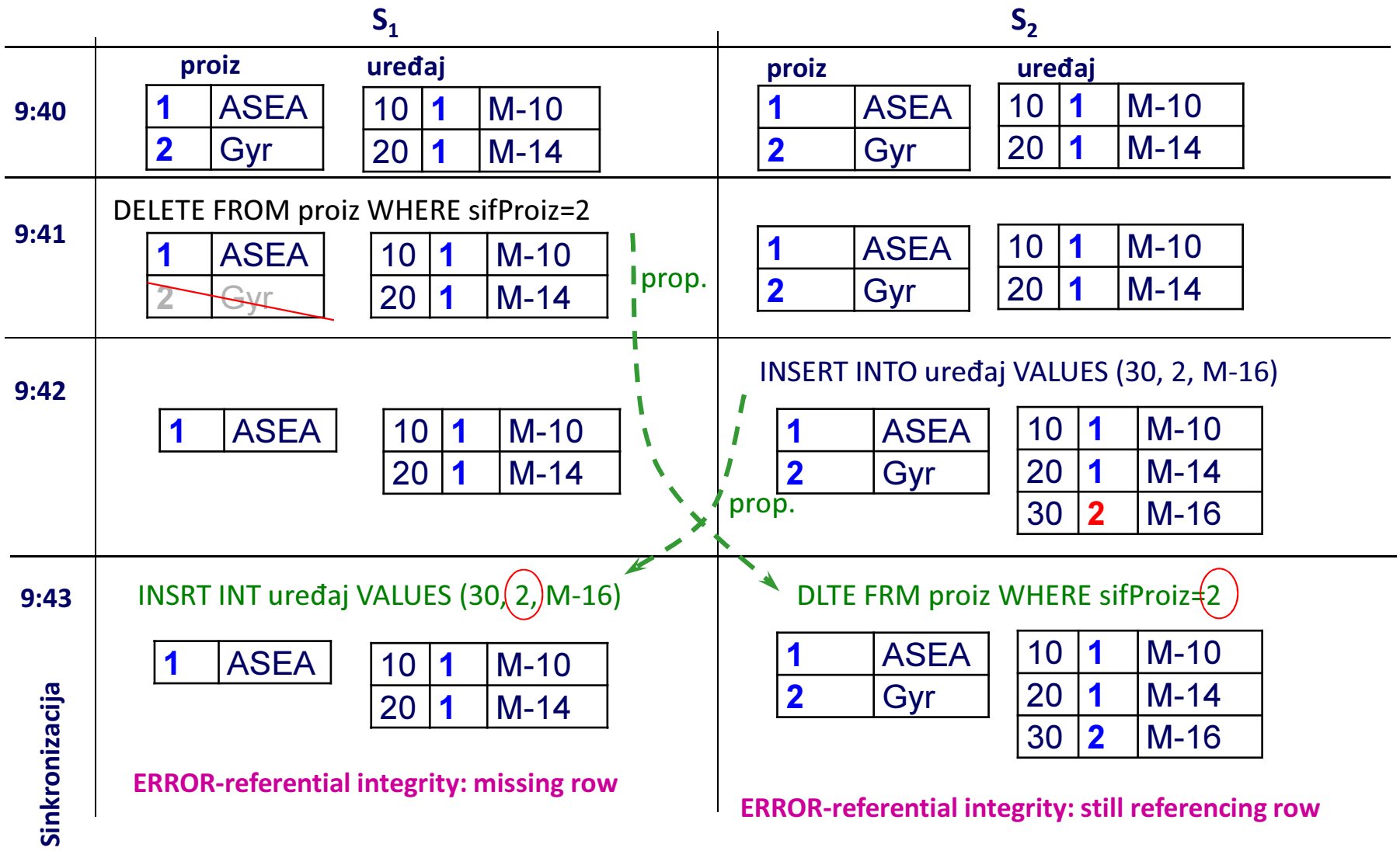
- neserijalizabilno obavljanje transakcija može dovesti do teško ispravljivog narušavanja konzistentnosti podataka
- problem detekcija konflikata: neki konflikti mogu biti otkriveni tek nakon propagacije izmjena (kad je inicijalna transakcija već potvrđena)
- problem razrješavanja konflikta: može zahtijevati poništavanje potvrđene transakcije → narušavanje svojstva izdržljivosti transakcije
- automatsko rješavanje konflikata često nije moguće – potrebna je intervencija čovjeka

Primjer:

potpuno replicirana baza podataka



Važni nedostaci dvosmjernih asinkronih protokola



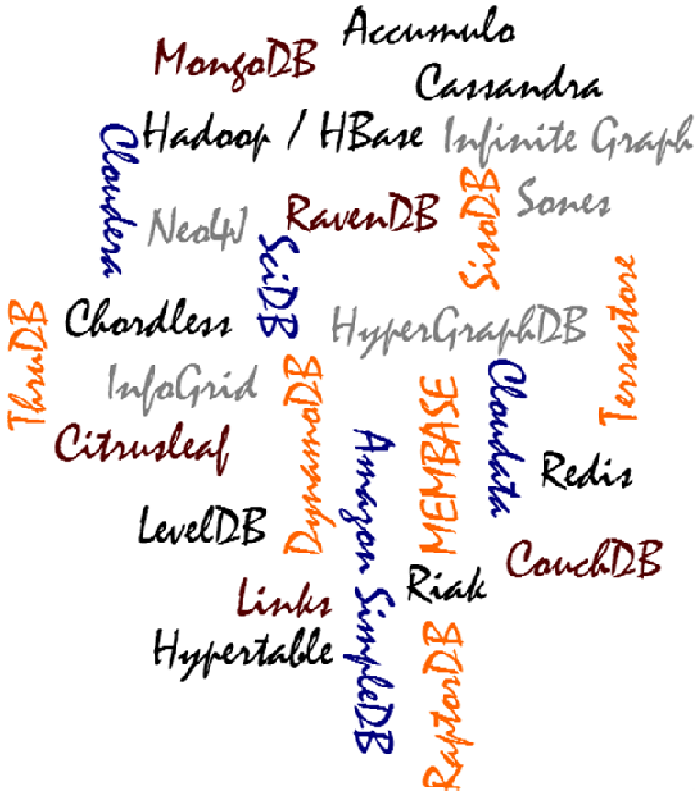
rezultat: *system delusion*

Važni nedostaci dvosmjernih asinkronih protokola

- moderni sustavi podržavaju dvosmjernu asinkronu replikaciju, ali sveobuhvatno rješenje za prethodno opisani problem ne postoji
- u različitim sustavima se nude različite ugrađene opcionalne funkcionalnosti koje mogu pomoći u specifičnim slučajevima. Npr. u nekim sustavima je moguće:
  - umjesto direktnog propagiranja SQL naredbe, propagirati poziv pohranjene procedure (koju definira korisnik) i koja razriješava moguće konflikte
  - ako se uz pomoć vremenskih oznaka (*timestamp*) utvrdi mogući konflikt, poništava se inicijalna transakcija (kako to utječe na izdržljivost?)
  - *last wins, first wins, greatest value wins, ...*

NoSQL

NoSQL baze podataka



- Naziv potječe od (relacijskog!) SUBP-a koji je 1998. razvio *Carlo Strozzi*
- Naziv (*twitter hashtag*) ponovo upotrijebljen 2009. godine na skupu „distribuiranih, ne-relacijskih baza podataka, otvorenog koda” kojeg je organizirao *Johan Oskarsson*
- No + SQL:
  - „SQL” se odnosi na „tradicionalne” relacijske SUBP-ove
  - Inicijalno tumačeno kao „ne koristi SQL”, odnosno ne koristi relacijske SUBP-ove
  - *Not Only SQL* – „ne samo SQL”, rješenja koja nisu zasnovana isključivo na relacijskim SUBP-ovima

NoSQL neformalna definicija

- Neformalna definicija (preuzeto s http://nosql-database.org/):

  Baze podataka novije generacije koje uglavnom imaju sljedeće značajke:

  ne-relacijske, distribuirane, otvorenog koda i horizontalno skalabilne…

  …često posjeduju i dodatna svojstva: nema podatkovnog modela, lagana replikacija, jednostavan API, BASE (nisu ACID), rad s velikom količinom podataka i sl.

otvorenog koda

ne-relacijske

distribuirane

web 21. stoljeća

nemaju shemu

NoSQL vs RBP

RDB su najbolje
one-size-fits-all
rješenje

NoSQL su **specijalizirana**
rješenja za određene
(grupe) problema

Model podataka

Model podataka - uvod

- Model kojim predstavljamo i radimo s podacima
- <> fizički model (kojeg većinom ne moramo znati)
- Npr. relacijski model
- Ne postoji "pravi" model svijeta odnosno domene

- U NoSQL svijetu, četiri glavna modela:
  1. Ključ-vrijednost (*Key Value*)
  2. Dokument (*Document*)
  3. *Column family (<> column, columnar)*
  4. Graf (*Graph*)