Adatbázisok elmélete
Elosztott adatbázisok

Gajdos Sándor

2023. május.

Elosztott DBMS bevezetés I.

- Distributed DBMS (DDBMS), definíció, site/node...
- Vö: párhuzamos architektúrák
- A DDBMS csp-jai:
  - általában földrajzilag is szeparáltak,
  - külön adminisztráltak,
  - a kapcsolatok lassúbbak (WAN), valamint
  - lokális és globális tranzakciók...

Elosztott DBMS bevezetés II.

Fő előnyök

- Rendelkezésreállás
- Adatmegosztás
- Autonómia

Hátrányok

- Nagyobb komplexitás (ld. Adatb algoritmusai...)
- Nagyobb feldolgozási overhead

Elosztott DBMS bevezetés III.

- Heterogén (multiadatbázis, federated DB, szövetséges ~)
  - Különböző DBMS-ek
  - Közös séma különböző csp-ok között (multiadatbázis réteg)
  - Globális tranzakciók nehézkesek
  - Létező rendszerek összekapcsolására
  - Kevésbé elterjedt
  - Pl. IBM DataJoiner (relációs, 2001-), DataBroker (OODB)
- **Homogén**
  - Minimum: azonos adatmodellt implementáló DBMS-ek
  - Közös, globális séma
  - (tip: mind támogassa a 2PL-t, csak azonos gyártó rendszerei, sőt azonos „típusú”)
  - Pl. Oracle, MySQL, SQL Server,…

Elosztott homogén adatbázisok

- Jellegzetességek:
  - Adatok multiplikált tárolása
  - Többszörös elérési utak
  - Nagy megbízhatóságú komponensek
- Főbb kihívások/tárgyalandó témakörök:
  - Elosztott zárkezelés
  - Elosztott tranzakciókezelés
  - Elosztott időbélyegek

Lokális-globális adatok

- Globális (logikai): egyetlen adategység a teljes rendszer szempontjából, amin a műveleteket megfogalmazzuk. Valójában részekből állhat. A részek lehetnek
  - Azonos másolatok (tudjuk, hol vannak)
  - Egy adatelem részei (tudjuk, mi-hol van)
  - Kombinációjuk.
- Lokális (fizikai): amin a műveleteket ténylegesen végrehajtjuk.
  - LOCK A → → → LOCK {Ai} ???

    (Logikai (globális) lokális (fizikai))
  - Logikai (ACID) tranzakció → → →fizikai tranzakciók ???

Elosztott zárak

Megvalósítás: lokális zárakkal úgy, hogy

- T_i (globális tr.) WLOCK A_k esetén egyetlen lok. tr. sem helyezhet el semmilyen zárat A_k egyetlen lokális példányára sem
- Ha A-nak egyetlen lokális példánya van:

  globális zárkezelés korrekt ←→ a lokális zárkezelés korrekt

  Ehhez: Ha pl. az N_i-beli T zárolni akarja A-t, amelynek egyetlen A_1 példánya az N_k-ban van, akkor N_i-ből egy üzenet megy N_k-ba, és az eredményről egy visszaüzenet N_i-be.
- Ha A-nak több lokális példánya van:

  Számos megoldás (lokális zárak + protokollok) (WALL, többségi, k az n-ből, elsődleges példányok, centrális csúcs, tokenes)

  Zárak költsége ~ üzenetek költsége
  - Kontroll üzenet: rövid, olcsó
  - Adat üzenet: hosszú, drágább

WALL protokoll I.

Write locks ALL

- Lokális zárszemantika: mint a R/W modellnél
- Globális zárszemantika:
  - RLOCK A: RLOCK A_i érvényes A-nak legalább egy példányán
  - WLOCK A: WLOCK A_i érvényes A-nak az összes példányán

Tétel: Globális zárkompatibilitási mátrix

	RLOCK	WLOCK
RLOCK	I	N
WLOCK	N	N

WALL protokoll II.

Elemzése, ha n csomópontban van A_i példány

- Globális WLOCK (kb.):
  - n db. kérés (kontroll)
  - n db. válasz (kontroll)
  - Ha zárolhat, akkor A_i új értéke n db. üzenet (adat)
  - (n db. zárfelszabadítás (kontroll), de ez a commit-tal megspórolható)
  - Összesen 2n kontroll + n adat
- Globális RLOCK:
  - Összesen 1 kontroll + 1 adat

Többségi zárolás

- Lokális zárszemantika: mint a R/W modellnél
- Globális zárszemantika:
  - RLOCK A: RLOCK A_i érvényes A példányainak többségén
  - WLOCK A: WLOCK A_i érvényes A példányainak többségén

Tétel: Globális zárkompatibilitási mátrix: mint WALL-nál

- Elemzése:
  - WLOCK *A*-hoz az *n* példány többségének ($\lceil (n+1)/2 \rceil$) kontroll üzenet, ugyanennyi válasz vissza, majd az íráshoz *n* adat üzenet.

    Összesen n kontroll + n adat
  - RLOCK *A*-hoz az *n* példány többségének ($\lceil (n+1)/2 \rceil$) kontroll üzenet, a válaszokból (legalább) az egyik adat üzenet, ez tartalmazza a kért adategység értékét.

    Összesen n kontroll + 1 adat

WALL és többségi összehasonlítása

- Üzenetek száma szerint:
  - sok olvasás: WALL hatékonyabb,
  - sok írás: többségi hatékonyabb.
- Patt valószínűsége szerint:

  Ha két tranzakció közel egyidőben akarja A-t zárolni:
  - WALL: valószínűleg patt lesz (aminek a feloldása költséges),
  - többségi zárolás: az egyik sikeres lesz, a másik pedig nem.

k az *n*-ből protokoll

Az előző kettő általánosítása, ahol $\lceil (n+1)/2 \rceil \leq k \leq n$

- Lokális zárszemantika: mint a R/W modellnél
- Globális zárszemantika:
  - WLOCK A: WLOCK A_i érvényes A-nak k db. példányán
  - RLOCK A: RLOCK A_i érvényes A-nak n+1-k példányán
- k=n megfelel a WALL-nak, k= $\lceil (n+1)/2 \rceil$ a többségi zárolásnak.

Elsődleges példányok módszere

- Az A adategység zárkéréseit az X_A csúcs ítéli meg (célszerűen, ahol van A-nak példánya, vö: „elsődleges példány”)
- Ha ez minden adategységre ugyanaz a node: *centrális csúcs módszere*
- Elemzése
  - WLOCK *A*-hoz egy kérés X_A-ba, erre egy válasz, majd (jó esetben) az íráshoz n adatüzenet.
  - RLOCK *A*-hoz egy kérés X_A-ba, erre egy válasz, majd az olvasáshoz 1 adatüzenet.
  - Összegezve: sokkal hatékonyabb, de sebezhetőbb, ha X_A kiesik

Tokenes protokoll I.

Elsődleges példányok módszerének adaptív továbbfejlesztése

- Elemei:
  - írási token: WT(*A*)
  - olvasási token: RT(*A*).
- WT(*A*) szemantika:
  - Ha az *X* csomópontban van, akkor az *X* csomópont jogosult az RLOCK *A*-t vagy WLOCK *A*-t megítélni az *X* csomópontban futó (globális) tranzakciók számára
- RT(*A*) szemantika:
  - Ha az *X* csomópontban van, akkor az *X* csomópont jogosult az RLOCK *A*-t megítélni az *X* csomópontban futó (globális) tranzakciók számára

Ha létezik WT(A), akkor nem létezhet RT(A), ha nincs WT(A), akkor viszont akárhány RT(A) létezhet.

Tokenes protokoll II.

A tokenek mozgatása

- **Írás:** Y csomópontban B adategység írásához WT(B)-t az Y csomópontba kell juttatni (ha nincs ott).
  - *Y*-ból WT(*B*)-t kérő üzenetek *mindegyik* csomópontba (*m* db kontroll), erre a válasz
    - *a)*, ha nincs náluk sem RT(*B*) sem WT(*B*), vagy náluk van, de lemondanak róla,
    - *b)*, ha nála van RT(*B*) vagy WT(*B*), és kell is neki. Ekkor a csúcs megjegyzi a kérést.
  - Az Y csomópont
    - a tokent megszerezheti, ha mindenki *a)*-t üzen (ehhez üzenetek minden csomópontba, hogy semmisítsék meg a B-re vonatkozó tokenjüket)
    - b) esetén visszavonja a kérését azoktól, akik *a)*-t válaszoltak

Tokenes protokoll III.

- **Olvasás:** Y csomópontban B adategység olvasásához RT(B)-t az Y csomópontba kell juttatni (ha nincs ott).
  - *Y*-ból RT(*B*)-t kérő üzenetek *mindegyik* csomópontba (*m* db kontroll), erre a válasz
    - Semmi, ha nála van RT(B)
    - *a)*, ha nincs nála WT(*B*), vagy nála van, de lemond róla,
    - *b)*, ha nála van WT(*B*), és kell is neki. Ekkor a csúcs megjegyzi a kérést.
  - Az Y csomópont
    - a tokent megszerezheti, ha mindenki *a)*-t üzen (ehhez üzeneteket küld, hogy semmisítsék meg a WT(B)-t
    - b) esetén nem tudja megszerezni RT(A)-t.
- A token mozgatása költséges, de utána már kb. csak az elsődleges példányok módszerének költsége jelentkezik.

Tranzakciókezelés DDBMS-ekben

Ismétlés

- A: pl. commit pont
- C: pl. szigorú protokollok
- I: pl. sorosítás
- D: pl. naplózás/többszörös diszkre írás

Elosztott tranzakciók problémái

Egy csomóponton elindított (logikai) tr. hatására számos további csomóponton indulhatnak el fizikai tr.-k a lokális adatpéldányok helye alapján.

- Cél (itt és most): ACI biztosítása
- Eszközök: elosztott zárak és protokollok
- **Def.:** tranzakciók egy ütemezése egy elosztott adatbázison **sorosítható**, ha hatása a logikai adategységeken ugyanaz, mintha a tranzakciók valamely soros ütemezésben futottak volna le.

Elosztott sorosíthatóság I.

- 2PL elégséges nem elosztott esetben, de...
- Ellenpélda: $T_1 = T_{11} + T_{12}$, $T_2 = T_{21} + T_{22}$

csomópont 1

T_{11}	T_{21}
WLOCK *A*	
UNLOCK *A*	
	WLOCK *A*
	UNLOCK *A*

csomópont 2

T_{12}	T_{22}
	WLOCK *B*
	UNLOCK *B*
WLOCK *B*	
UNLOCK *B*	

A lokális sorosítási gráfok:

$T_1 \rightarrow T_2$ $\qquad$ $T_1 \leftarrow T_2$

A globális sorosítási gráf:

$T_1 \rightleftarrows T_2$

Elosztott sorosíthatóság II.

- A lokális kétfázisúság nem elégséges a (globális) sorosíthatósághoz
- Globális sorosíthatóság elégséges feltétele: *globális kétfázisúság*
- **Def.:** egy T_i (globális/logikai) tranzakció egyetlen T_{ij} lokális tranzakciója sem engedhet el egyetlen (lokális) zárat sem, ameddig bármelyik T_{ij} még kérhet új zárat
- Elosztott megegyezési feladat (distributed agreement)
- Megvalósítás: **közös zárponttal**

Elosztott sorosíthatóság III.

- Lavinák problémája megmaradt
- Megoldás: elosztott szigorúság (nincs írás az adatbázisba, ameddig mindegyik lokális tr. el nem érte a commit/kész pontját).
- Elosztott megegyezés ☺
- Megvalósítás: **közös commit (kész) pont**
- Tr. begin,... közös zárpont,... közös készpont,...Tr. end
- Működik, de költséges
- Közös commit pont → közös zárpont
- Közös commit pont = közös zárpont képzése

Elosztott készpont képzése I.

Alapeset: nincsenek hibák

- *T* logikai/globális tranzakció, T_i lokális tranzakciók az N_i csomópontokban.
- Ahol a tranzakciót kezdeményezték: *Koordinátor (főnök)*
- többi csomópont: *résztvevő*
- Cél a közös döntés:
  - vagy mindenki abortáljon
  - vagy mindenki commitáljon.
- Megoldás: üzenetváltások a főnök és a résztvevők között
  1. A csomópontok a lokális commitokról vagy abortokról üzennek a főnöknek
  2. Főnök döntést hoz (commit v. abort)
  3. Főnök ezt megüzeni a lokális tranzakcióknak
  4. A lokális tr.-k egységesen abortálnak vagy commitálnak.

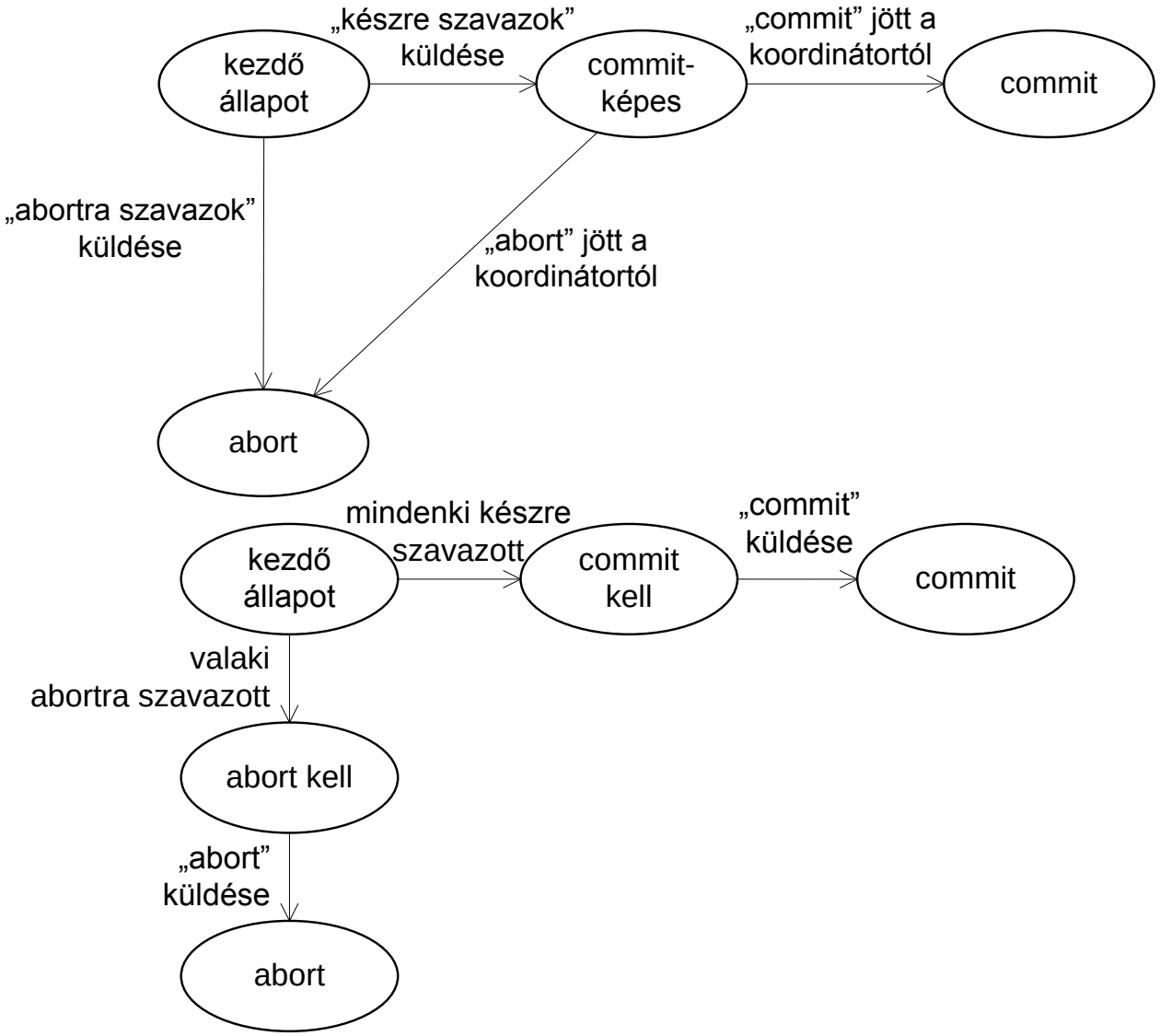
Elosztott készpont II. - az üzenetek

- Üzenet a csomópontoktól:
  - a csomópont commitra szavaz, ha a lokális tranzakció elérte a készpontját
  - a csomópont abortra szavaz, ha a lokális tranzakció abortálni kényszerült.
- Üzenet a főnöktől:
  - Ha minden résztvevőtől „készre szavazok” üzenetet kapott, akkor mindenkinek „commit” üzenetet küld. Ez alapján a résztvevők egységesen tudják commitálni a lokális tranzakciókat.
  - Ha bárhonnan is „abortra szavazok” üzenetet kap, akkor „abort” üzenetet küld. Ez alapján a résztvevők egységesen abortálják a lokális tranzakciókat.

Elosztott készpont - Hibamentes eset I.

résztvevő

főnök



Elosztott készpont - Hibamentes eset II.

- A főnök is résztvevő
- Főnök nélkül az üzenetszám n^2-tel arányos
- Commit akkor kell, ha **mindenki** commitra szavazott, és akkor, miután ezt a résztvevő megtudja
- Hibák esetén hibásan/egyáltalán nem működik
  - Pl. commit képes állapotban nem kap üzenetet
    - commit nem lehet, mert jöhet „abort”,
    - abort sem lehet, mert jöhet „commit”,
    - zárait sem engedheti el.
  - „blokkolás”, a fő cél ennek megakadályozása ->2PC

Kétfázisú commit protokoll (2PC) I.

- Főnök felszólítja a résztvevőket szavazásra
- Időmérés
- Hiba esetén
  - Segítségkérés a többi résztvevőtől
  - Stabil tár alapján emlékszik a korábbi döntésére (naplózás)

2PC II.

résztvevő

főnök

2PC III.

- Commit akkor van, ha/amikor a résztvevő megtudja, hogy mindenki commitra szavazott
- Helyreállítás
  - Segítségkérés küldése, más résztvevő
    - ha commit állapotban van, akkor commitot küld,
    - ha abort állapotban van, akkor abortot küld,
    - ha commit képes állapotban van, akkor nem küld semmit.
  - nem kaphat commit és abort választ is a segítségkérésre,
  - nem kaphat abortot, ha a koordinátor már globális commitot küldött,
  - nem kaphat commitot, ha a koordinátor már globális abortot küldött.

2PC IV.

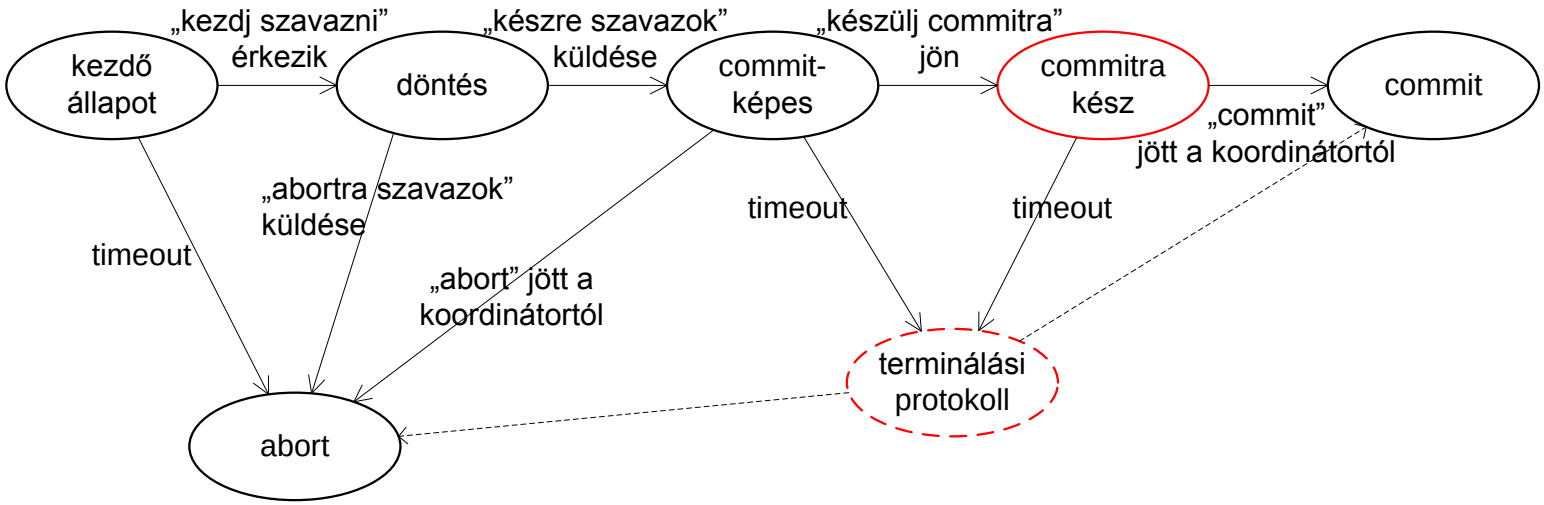
- Kezelni tudja:
  - Hálózati hiba miatt elvesző üzenetek
  - Kieső csomópont, amely később újra is indulhat
- Nem tudja kezelni pl.:
  - Üzenetsorrend felcserélődés
  - valótlan/hibás üzenetek
- Blokkolás lehet, ha a résztvevő semmilyen segítséget nem kap
  - Kieső főnök, miután begyűjtötte a szavazatokat
  - Kieső résztvevők, akik már értesültek a döntésről

A háromfázisú kész protokoll (3PC) I.

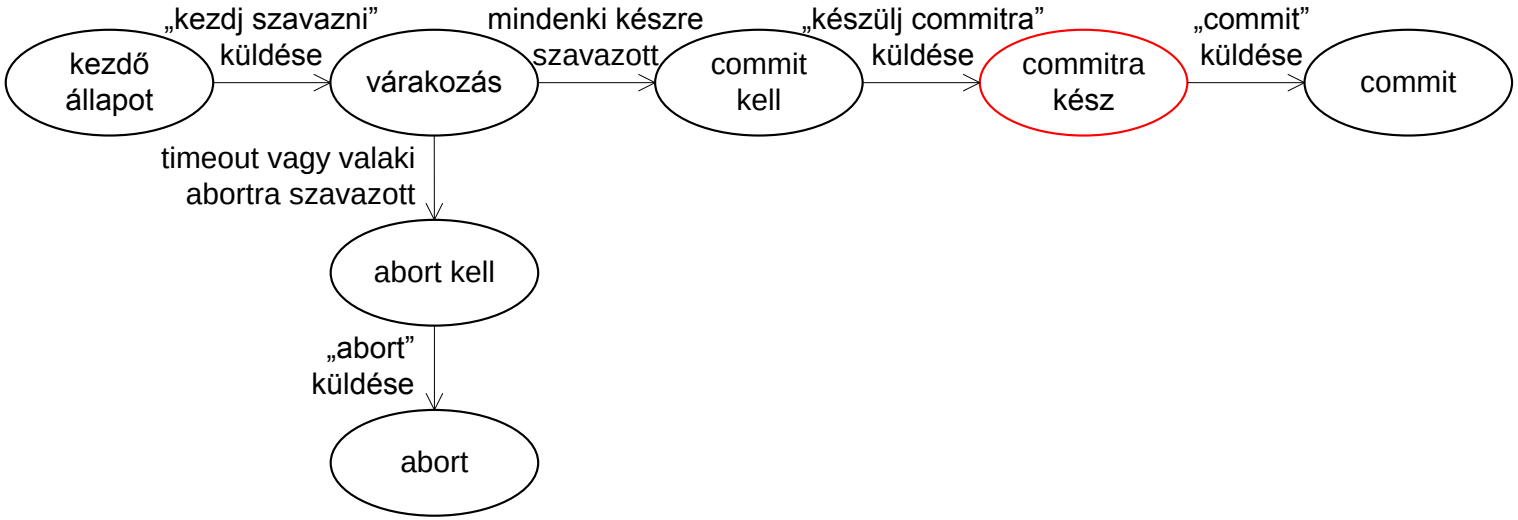
- 2PC-nél commit akkor, ha a résztvevő megtudja, hogy mindenki commitra szavazott.
- Ha a résztvevő nem tudja meg, hogy mindenki commitra szavazott, akkor blokkolódik (pl. mert a főnök kiesik)
- Megoldás: biztosítsuk, hogy **mindenki meg is tudja azt**, hogy mindenki commitra szavazott.
- Erről informálni is kell a résztvevőket: 3PC

3PC II.

résztvevő



főnök



3PC III.

1. 2PC-ből ismert, szavazásra felszólítás majd szavazás
2. Egységes commit esetén a főnök „commitra készülj” üzenetet küld, majd nyugtára vár. Egy nyugta = „a résztvevő már tudja, hogy mindenki commitra szavazott”. Összes nyugta = „*a főnök már tudja*, hogy minden résztvevő tudja, hogy mindenki commitra szavazott”.
3. a főnök commit üzenetet küld a résztvevőknek, amiből *minden résztvevő is megtudja*, hogy már mindenki megtudta, hogy mindenki commitra szavazott. ☺

A háromfázisú kész protokoll (3PC) IV.

- blokkolás megoldására terminálási protokoll:
- Ha „commit képes” vagy „commitra kész” állapotban timeout, akkor
  - a működő csomópontok új főnököt választanak
  - új főnök bekéri a résztvevők állapotait.
  - döntést hoz az új főnök és ezt közli a résztvevőkkel:
    - Ha van „abortált” vagy még nem szavazott, akkor globális abort.
    - Ha van „commitált” résztvevő, akkor globális commit.
    - ...

Elosztott időbélyeges tranzakciókezelés

Ismétlés (nem elosztott környezetben):

- t(*Tranzakció*) szigorúan egyedi, rendszeridővel arányos
- Egyértelmű sorrendet határoz meg
- T_r-k a kezdőidejükben zérus idő alatt lefutnak
- A soros ekvivalens a $t(T_{i1}) < t(T_{i2}) <\ldots< t(T_{in})$-nek megfelelő $T_{i1}, T_{i2},\ldots,T_{in}$
- Minden írás-olvasás előtt adategységek időbélyegeinek vizsgálata, döntés, időbélyegek frissítése (atomian)

Elosztott időbélyeges tranzakciókezelés II.

- Korábbi elvek alkalmazhatók
- Ha egy globális/logikai tranzakció egy *N* csúcsban írja/olvassa valamely A_i példányt, akkor rajta hagyja az időbélyegét
- Írni minden példányt kell, olvasni elég egyet
- Tr. időbélyegét az a csp. adja, ahol a tr. elindult.
  - Egyértelműséghez: csp azonosító hozzáadása az időbélyeg LSB-hez
  - A csp-ok órái különbözőképpen járhatnak.

Elosztott időbélyeges tranzakciókezelés III.

- Művelet végrehajthatósága: lokális időbélyegek vizsgálata alapján
- Hasonló, mint elosztott zárkezelésnél, pl.
  - READ *A* esetén egyetlen R(A_i), W(A_i) vizsgálata
  - WRITE *A* esetén az összes R(A_i), W(A_i) vizsgálata
  - …
- Óraeltérések problémája
  - Egzakt szinkronizmus nem létezik
  - Következményei

Egyéb tranzakciókezelési kérdések

- Csúcsok helyreállítása rendszerhibák után
- Elosztott pattok keletkezése és kezelése
- …

Irodalom:

Gajdos: Adatbázisok, 11. fejezet

Jó tanulást!