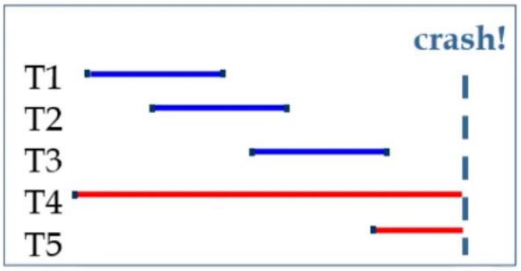
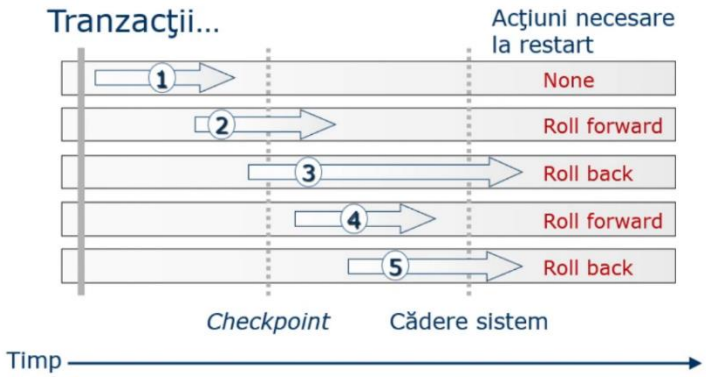
Recuperarea datelor

Recuperarea datelor și ACID

- **Atomicitatea** – garantată prin refacerea efectului acțiunilor corespunzătoare tranzacțiilor necomise
- **Durabilitatea** – garantată prin asigurarea faptului că toate acțiunile tranzacțiilor comise „rezistă" erorilor și întreruperilor neașteptate ale funcționării sistemului



Ex: modificările făcute de T1, T2 și T3 trebuie să fie văzute (trebuie să fie durabile) , iar cele de T4 și T5 nu (efectele nu vor persista)



Roll forward – să vedem care a fost momentul până unde știm că toate modificările sunt în baza de date și de acolo să reluăm execuția tranzacției

Categorii generale de întreruperi

(1) Eşuarea tranzacţiilor – eșec simplu
- unilateral sau din cauza unui deadlock
- in medie 3% din tranzacţii eşuează (date de intrare eronate, cicluri infinite, depăşirea limitei de resurse)

(2) Eşuarea sistemului – eșec simplu
- Eşuarea procesorului, memoriei interne, etc...
- Conţinutul memoriei interne se pierde însă memoria secundară nu este afectată

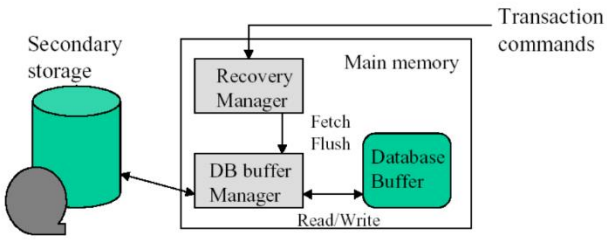
(3) Eşecuri media – eșec catastrofal
- Pierdere date de pe hard disk

Recuperarea datelor

- Eșecuri simple
  - se folosește logul de tranzacții
  - Anularea modificărilor prin inversare operații
  - Re-executarea unor operații

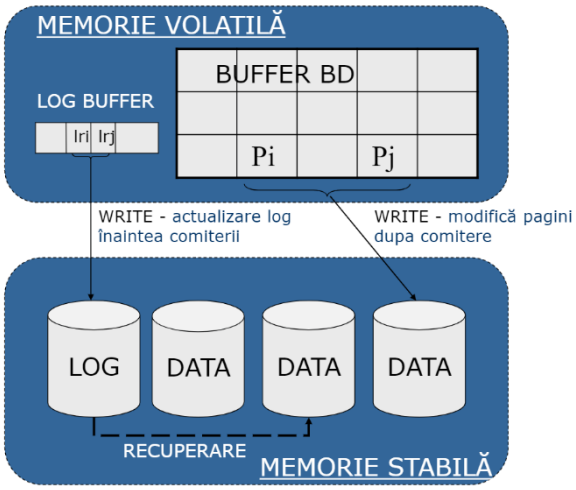
- Eșecuri catastrofale
  - Utilizarea arhivelor pentru restaurare
  - Reconstruirea celei mai recente stări consistente prin re-executarea acțiunilor tranzacțiilor comise

Recovery Manager



- Memorie volatilă: memoria principală (conține buffer)
- Memorie stabilă: disc magnetic(sau variante). Rezistent la erori, iar datele se pierd numai atunci când are loc o eroare fizică sau un atac intenționat

Logarea acțiunilor



- De obicei, logul se stochează pe un disc diferit de cel pe care se află baza de date.
- Logul conţine înregistrări (sau intrări) adăugate mereu la final.
- Pentru recuperare logul este citit în ordine inversă
- O intrare in log conţine:
  - Identificatorul tranzacţiei
  - Tipul operaţiei(inserare, ștergere, modificare)
  - Obiectul accesat de către operaţie
  - Vechea valoare a obiectului
  - Noua valoare a obiectului
  - ...
- Log-ul mai poate conţine
  - begin-transaction – pentru oprirea căutării inverse
  - commit-transaction,
  - abort-transaction.
  - End
- Dacă o tranzacţie T e întreruptă, atunci se realizează un rollback → scanare inversă a log-ului, iar când se întâlnesc acţiuni ale tranzacţiei T, valoarea iniţială a obiectului modificat este salvată în BD.
- La refacerea contextului după o întrerupere:
  - commit→tranzacţiile complete
  - tranzacţiile active→abort.

Modificările bazei de date

O tranzacţie T modifică obiectul x aflat în buffer. Dacă apare o întrerupere înainte de finalizarea execuției tranzacției:

Scenariul 1: Modificarea nu a reuşit să se salveze pe disc →T este anulată. BD consistent

Scenariul 2: Modificarea lui x se salvează pe disc, dar întreruperea a survenit înaintea modificării logului → Nu se poate face rollback deoarece nu există informaţia despre valoarea anterioară a lui x → BD inconsistent.

Scenariul 3: Modificarea lui x fost logată și s-a actualizat şi baza de date →T este anulată şi valoarea originală este utilizată pentru a înlocui valoarea din baza de date → BD consistent.

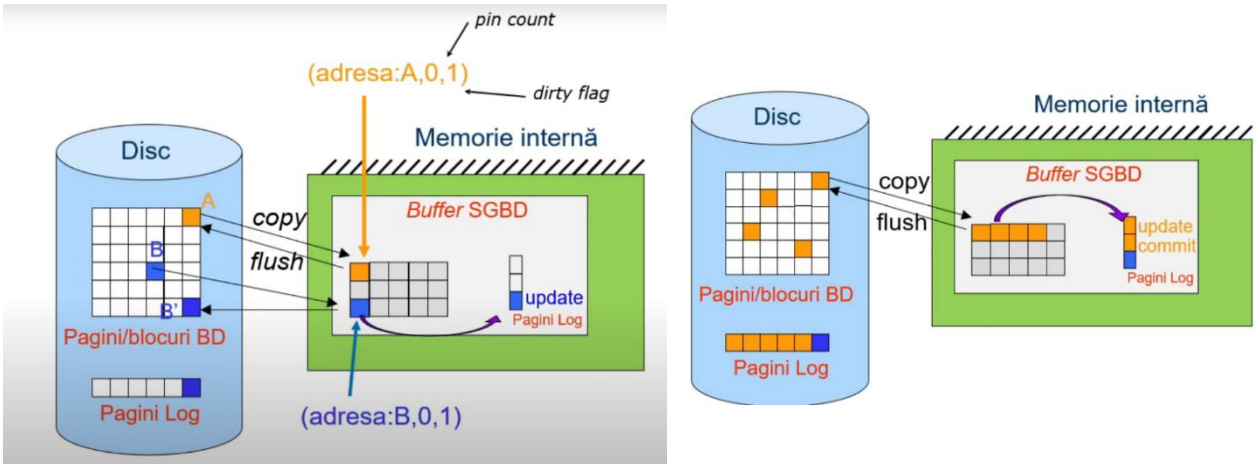
Checkpoint

– Acțiuni:
  - Suspendă acțiunea tuturor tranzacțiilor
  - Forțează salvarea pe disc a tuturor paginilor din buffer care au fost modificate (dirty flag = true)
  - Adaugă în fișierul de log o intrare checkpoint și o salvează pe disc imediat după salvarea paginilor
  - Reia execuția tranzacțiilor
– Un checkpoint se execută la fiecare m minute sau t tranzacții

Recuperarea datelor într-un context nedistribuit

Actualizarea datelor

– **Actualizare imediată:** de indată ce s-a realizat o modificare în buffer, este actualizat si corespondenta paginii de date de pe disc
– **Actualizare amânată:** toate datele modificate în buffer sunt actualizate pe disc după ce execuția unei tranzacții sau a unui nr. Fix de tranzacții este finalizată
– **Actualizare „in-place”:** versiunea originală a paginii ce conține datele pe disc este suprascrisă de corespondenta sa din buffer
– **Actualizare „shadow”:** pagina de date din buffer nu se copiază peste corespondenta sa originală de pe disc, ci peste o copie a acesteia memorată la o adresă diferită



Protocol Write-Ahead-Logging(WAL):

Modificările unei înregistrări trebuie inserate în log înainte a actualizării bazei de date!

1. Trebuie asigurată adăugarea unei intrări coresp. unei modificări în loc *înainte* ca pagina ce conține înregistrarea să fie salvată pe disc
2. Trebuie adăugate toate intrările corespunzătoare unei tranzacții înainte de commit.

Intrările se adaugă în coada unui logging

Conflict de interes: Buffer Manager vs Recovery Manager

Buffer Manager	Recovery Manager
Performanță – minimizează nr. de transferuri	Atomicitate & Durabilitate

Poate decide Buffer Manager-ul salvarea anumitor pagini (modificate de o tranzacție din buffer pe disc fără a aștepta instrucțiuni specifice de la Recovery Manager? (uncommitted transaction)

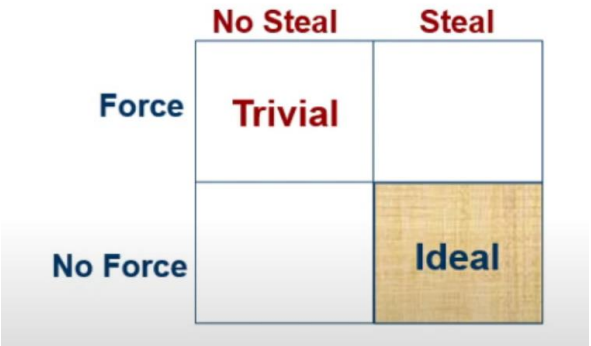
- ⇨ Decizie steal/no-steal
- ⇨ No-steal înseamnă că RM păstrează referința către paginile modificate din buffer
- ⇨ Steal – BM îi fură una din acele pagini și o salvează pe hard disk, eliberează zona de memorie și înlocuiește pagina cu altă pagină care probabil a fost reclamată de o altă tranzacție care se execută în paralel
- ⇨ BM nu poate „fura" pagini care au pin-count>=0

Poate RM „forța" BM să salveze anumite pagini din buffer pe disc la finalul executării unei tranzacții? (committed transactions)

- ⇨ Decizie force/no-force
- ⇨ Force – in momentul în care o tranzacție s-a terminat cu succes, pentru a garanta durabilitatea, RM vrea să ducă toate modificările făcute de tranzacție pe hard-disk. Astfel, nr de transferuri poate crește
- ⇨ No-force – BM nu-l lasă să facă acest lucru

Se forțează salvarea pe disc a fiecărei modificări?

- Timpi mari de răspuns
- Garantează durabilitatea
- Garantează atomicitatea



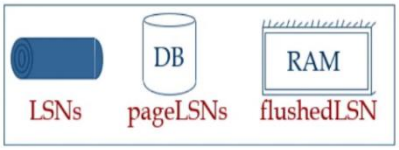
Se permite salvarea unor pagini de memorie modificate de tranzacții ce nu s-au comis?

- Dacă nu, concurență redusă, anumite tranzacții fiind blocate
- Dacă da, cum se poate garanta atomicitatea?

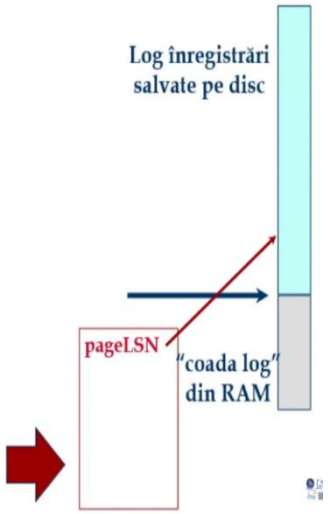
- **Steal / No-force**
  - BM poate salva modificări intermediare ale tranzacţiilor.
  - RM salvează doar un commit
- **Steal / force**
  - BM poate salva modificări intermediare ale tranzacţiilor.
  - RM salvează toate modificările (flush) înainte de commit
- **No-steal / no-force**
  - Niciuna din paginile modificate nu se salvează decât la commit.
  - RM salvează un commit şi elimină referinţele către paginile modificate.
- **No-steal / force**
  - Niciuna din paginile modificate nu se salvează decât la commit.
  - RM salvează toate modificările (flush) la commit

- **STEAL** (de ce garantarea Atomicității e dificilă)
  - To steal frame F: Pagina curentă memorată în F (să spunem P) este copiată pe disc; este posibil ca anumite tranzacții să blocheze anumite obiecte memorate în P.
    - ⇨ Ce se întâmplă dacă tranzacția k, ce bloca anumite obiecte din P, eșuează?
    - ⇨ Trebuie memorată vechea valoare a lui P (pentru a aplica UNDO modificărilor apărute în pagina P)
- **NO FORCE** (de ce garantarea Durabilității e dificilă)
  - Ce se întâmplă dacă sistemul se blochează înainte ca o pagină modificată să fie copiată pe disc?
  - În momentul comiterii unei tranzacții este necesar să se scrie pe disc informația minimă pentru ca modificările tranzacției să poată fi reproduse.

Contextul WAL



- Fiecare intrare din log are un Log Sequence Number (LSN), care crește incremental
- Fiecare pagină de date conține un pageLSN = LSN al celei mai recente intrări din log a unei modificări din pagină
- Sistemul mai reține un flushedLSN = LSN maxim până la care totul e salvat pe disc
- pageLSN <= flushedLSN



Câmpurile intrărilor:

- LSN
- prevLSN – operația anterioară executată de tranzacție
- TransID – identificatorul tranzacției care a generat această intrare
- type (Update, Commit, Abort, Checkpoint, End (terminarea unui commit sau abort), Compensation Log Records (CLRs) – pentru UNDO)
- pageID – care pagină de pe hard disk e modificată
- length
- offset – distanța de la începutul acelei pagini
- before-image
- after-image

} Doar pentru modificări

Compensation Log Record (CLR)

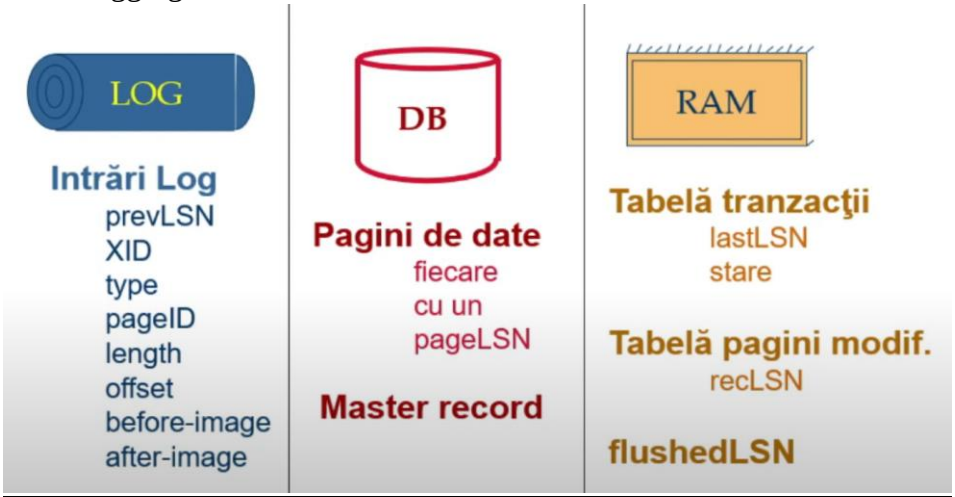
- Utilizat în faza de recuperare a datelor
- Este adăugat chiar înainte de anularea unei modificări marcate printr-o intrare în log
- Conține un câmp numit undoNextLSN = LSN-ul următoarei intrări de tip update ce trebuie anulată pentru o anumită tranzacție; se inițializează cu prevLSN al intrării curente
- Indică ce acțiuni au fost deja anulate
- Previne anularea de mai multe ori a aceleiași acțiuni

Alte construcții utilizate de RM

- Tabela de tranzacții
  - O înregistrare pentru fiecare tranzacție activă
  - Conține XID(id tranzacție), stare (running/committed/aborted) și lastLSN
- Tabela paginilor cu modificări (Dirty Page Table)
  - O înregistrare pentru fiecare pagină cu modificări din buffer
  - Conține recLSN = LSN al primei intrări din log care a adus o modificare paginii

Execuția normală a unei tranzacții

- Secvență de citiri & modificări, urmate de commit sau abort (Vom presupune că scrierea unei pagini pe disc e atomică)
- Strict 2PL
- Abordare gestiune buffer: STEAL, NO-FORCE
- Write-Ahead Logging



Exemplu: Întreruperea simplă a unei tranzacții

- Se parcurge log-ul în ordine inversă, anulând modificările
  - Se pornește de la lastLSN al tranzacției din tabela de tranzacții
  - Se parcurge lista de intrări ale log-ului urmând câmpul prevLSN
  - Înainte de anulare se adaugă o înregistrare Abort în log – utilă la recuperarea în cazul unei întreruperi în timpul operației de anulare a modificărilor
- Obiectul a căreia modificare se anulează va fi blocat
- Înainte de salvarea noii valori se adaugă un CLR:
  - Log-ul se actualizează și pe parcursul anulării

- Câmpul undonextLSN al CLR referă următoarea intrare din log pentru anulat (adică prevLSN al înregistrării anulate)
- Intrările de tip CLR nu se anulează niciodată

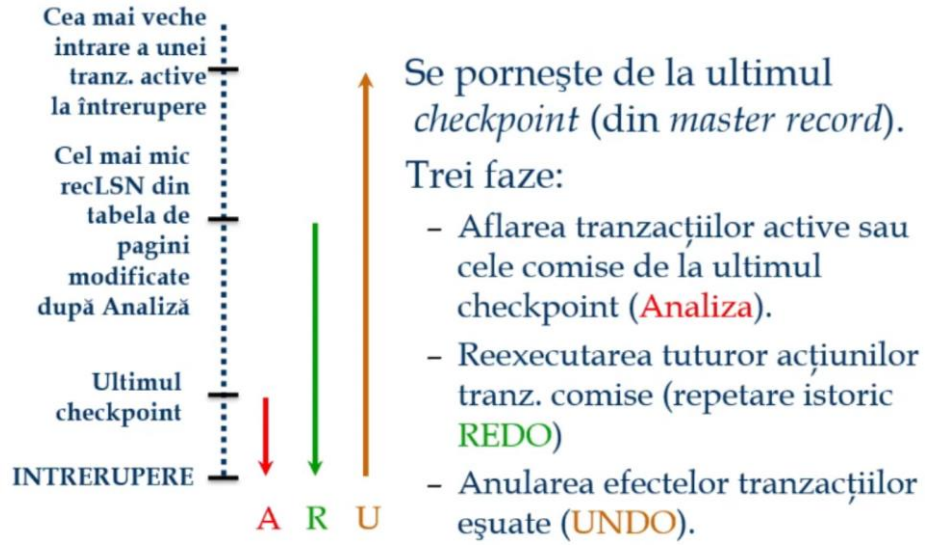
- La finalul anulării tuturor modificărilor tranzacției se inserează o intrare end în log

Comiterea unei tranzacţii

- Se inserează o intrare commit în log.
- Toate intrările de log corespunzătoare tranzacţiei se salvează pe disc (până la lastLSN).
  - Garanteză că flushedLSN>=lastLSN.
  - Inserările în log se fac secvenţial, sincron pe disc
  - Există mai multe intrări de log per pagină.
- Se inserează o intrare end în log.

Faze ale ARIES (Algorithm for Recovery and Isolation Exploiting Semantics)

- Analiză: Se parcurge log-ul de la cel mai recent checkpoint spre final pentru identificarea tuturor tranzacţiilor active şi a tuturor paginilor modificate existente în buffer la momentul întreruperii
- Redo: Reface toate modificările paginilor din buffer, corespunzătoare tranzacţiilor comise înainte de întrerupere, pentru a asigura că toate modificările s-au salvat pe disc.
- Undo: Modificările tuturor tranzacţiilor active în momentul întreruperii se anulează (folosind valoarea anterioară prezentă în intrare), mergând din spate în faţă.



Recuperarea datelor distribuite

- Tipuri noi de eșec: întrerupere rețea și oprire site-uri
- Dacă „sub-tranzacțiile” unei tranzacții sunt executate pe site-uri diferite, trebuie să ne asigurăm că se vor comite toate sau niciuna
- E nevoie de un „protocol de comitere” a „sub-tranzacțiilor” unei tranzacții – fiecare site are propriul log unde se vor memora acțiunile protocolului de comitere

Comitere în două faze (2PC)

- Site-ul de unde se generează tranzacţia se numeşte coordonator; celelalte site-uri pe care se execută se numesc subordonate.
- Atunci când tranzacţia comite:
  1. Coordonatorul transmite mesajul **prepare** tuturor subordonaţilor.
  2. Subordonaţii inserează **abort** sau **prepare** în log şi apoi transmit mesajul **no** sau **yes** către coordonator.
  3. Dacă coordonatorul primeşte yes de la toţi subordonaţii, inserează **commit** în log record şi transmite **commit** tuturor. Altfel, inserează **abort** în log rec şi transmite **abort** tuturor.
  4. Subordonaţii inserează **abort/commit** în log pe baza mesajului primit, apoi transmit **done** coordonatorului.
  5. Coordonatorul scrie **end** în log după ce primeşte toate done-urile.
- Comentarii:
  - ⇨ Două runde de comunicare: votare urmat de terminare. Ambele sunt iniţiate de coordonator.
  - ⇨ Orice site poate decide eşuarea tranzacţiei.
  - ⇨ Fiecare mesaj reflectă o decizie; pentru a garanta că această decizie rezistă unor erori, ea este inserată mai intâi într-un log
  - ⇨ Toate intrările în log conţin TransactionID şi CoordinatorID. Comenzile abort/commit logate de către coordonator includ id-urile tuturor subordonaţilor.
- Recuperarea datelor
  - ⇨ Dacă avem un commit sau abort logat pentru tranzacția T, dar nu este un end, se apelează redo/undo pentru T.
    - Dacă site-ul este coordonator pentru T, se vor transmite mesaje commit/abort către subordonați până se recepționează done.
  - ⇨ Dacă avem un prepare logat pentru tranzacția T, dar nu este commit/abort, iar site-ul este subordonat lui T
    - se contactează coordonatorul în mod repetat pentru verificarea stării lui T, apoi se inserează commit/abort în log rec + redo/undo aplicat asupra lui T; se inserează end in log.
  - ⇨ Dacă nu apare nici măcar un prepare în log pentru T, T se va termina unilaterar
    - Acest site poate fi chiar coordonator!
- Blocări:
  - ⇨ Când coordonatorul pentru tranzacția T eșuează, subordonații care au votat yes nu se vor putea decide dacă să termine cu commit sau abord până când coordonatorul își revine
    - T este **blocat**
    - Chiar dacă toți subordonații ar putea comunica între ei (prin extra info transmisă cu mesajul prepare) ei rămân blocați până când unul din ei transmite no
- Eșuarea rețelei / a unui site
  - ⇨ Dacă un site nu răspunde în timpul derulării protocolului de comitere pentru tranzacția T
    - dacă site-ul curent este coordonator pentru T, T va trebui întrerupt

        - dacă site-ul curent este un subordonat și nu a transmis încă yes, T va trebui întrerupt
        - dacă site-ul curent este un subordonat și a transmis yes, este blocat până când coordonatorul răspunde
- Mesajul done e folosit pentru a informa coordonatorul că poate „ignora” o tranzacție; tranzacția T rămâne în tabela de tranzacții până aceasta recepționează toate mesajele done
- Dacă coordonatorul eșuează după trimiterea mesajului prepare și înainte de scrierea în log a instrucțiunilor commit/abort, la revenire tranzacția se va termina fără succes
- Dacă o sub-tranzacție nu modifică BD, faptul că ea se comite sau nu este **irelevant**

2PC cu eșuare dedusă

- Bazată pe 2PC
- Atunci când coordonatorul întrerupe tranzacția T, reface contextul de dinaintea execuției lui T și o elimină imediat din tabela de tranzacții
    - ⇨ Mesajele done nu se mai așteaptă; avem „eșec dedus” dacă tranzacția nu se află în tabela de tranzacții. Intrarea abort din log nu conține în acest caz numele subordonaților
- Subordonații nu transmit done la eșec
- Dacă sub-tranzacțiile nu modifică BD, acestea răspund la prepare cu reader în loc de yes/no
- Coordonatorul va ignora tranzacțiile „reader”
- Dacă toate sub-tranzacțiile sunt „reader” a doua fază nu este necesară

Protocol de comitere în trei faze (3PC)