Administrare BD

Indecşi

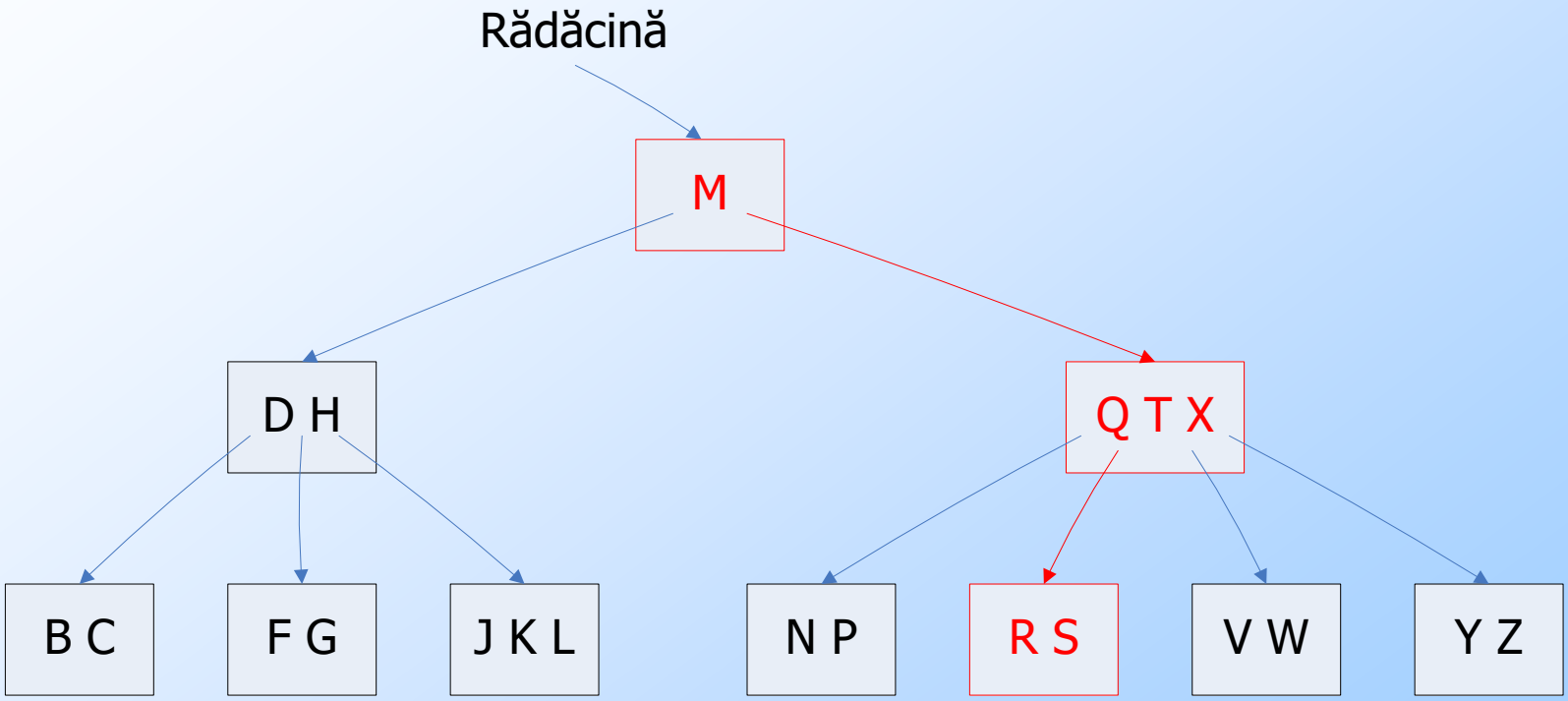
Autorizare

Tranzacţii

Indecşi

- *Index* = structură de date folosită (în scopul performanţei) pentru a mări viteza de acces la tuplele unei relaţii, definită pe valorile unuia sau mai multor atribute.
- Poate fi o tabelă de dispersie (hash), dar de obicei SGBD-urile folosesc arbore de căutare echilibrat cu noduri gigant (o pagină disc completă) numit *B-tree*.

B-Tree



Un arbore “B-Tree”, cu culoare roşie este marcat traseul de căutare a literei R.

Declarare Indecşi

- Nu există standard!
- Sintaxa:

```
CREATE INDEX BeerInd ON
  Beers(manf);

CREATE INDEX SellInd ON
  Sells(bar, beer);
```

Folosire Indecşi

- Fiind dată valoarea *v*, indexul conduce doar la acele tuple ce au *v* în atributul(-ele) indexului.
- Exemplu: se foloseşte BeerInd şi SellInd pentru a găsi preţurile berilor fabricate de “Pete’s” şi vândute de “Joe”.

Folosire Indecşi

```
SELECT price FROM Beers, Sells
WHERE manf = 'Pete''s' AND
    Beers.name = Sells.beer AND
    bar = 'Joe''s Bar';
```

1. Se foloseşte BeerInd pentru a obţine toate berile fabricate de "Pete's".
2. Apoi se foloseşte SellInd pentru a obţine preţurile acelor beri, ce au bar = "Joe's Bar"

“Database Tuning”

- Decizia ce indecşi să fie creaţi reprezintă o problemă foarte importantă.
- Pro: Un index măreşte viteza de execuţie a interogărilor (ce fac uz de indexul respectiv).
- Con: Un index încetineşte toate actualizările relaţiei de definiţie a indexului deoarece trebuie modificat şi indexul.

Exemplu: Tuning

- Presupunem că singurele lucruri făcute cu BD “beers” au fost:
  1. S-au adăugat fapte noi într-o relaţie (10%).
  2. S-a căutat preţul unei beri la un bar dat (90%).
- În acest caz SellInd pentru Sells(bar, beer) este necesar, dar BeerInd pentru Beers(manf) nu este necesar.

Sfaturi pentruTuning

- Să se efectueze o cercetare laborioasă.
  - Reglarea manuală pentru performanţă este dificilă.
- Sfatul este să se urmărească *încărcarea interogărilor*, de exemplu:
  1. Se aleg aleator interogări din istoricul interogărilor pe BD, sau
  2. Proiectantul oferă un eşantion al încărcării.

Sfaturi pentruTuning

- Sfatul este ca în continuare să se genereze indecşi candidaţi şi să se evalueze fiecare index pentru încărcare .
  - Fiecare eşantion de interogare se transmite optimizatorului de interogări, presupunându-se că doar indexul curent este disponibil.
  - Se măsoară îmbunătăţirea/degradarea duratei medii de rulare a interogărilor.

Autorizare

Autorizare

- Un sistem fişier identifică anumite privilegii asupra obiectelor (fişierelor) pe care le gestionează.
  - De obicei read, write, execute.
- Un sistem fişier identifică anumiţi participanţi cărora l-i se acordă (grant) privilegiile.
  - De obicei pentru owner, un grup, toţi utilizatorii.

Privilegii

- SQL identifică un set mult mai detaliat de privilegii asupra obiectelor (relaţiilor).
- Există nouă privilegii în total, unele din ele pot fi restricţionate la nivel de coloană, altele la nivel de tabelă.

Privilegii

- Cele mai importante privilegii pentru o relaţie:
  1. SELECT = dreptul de a interoga relaţia.
  2. INSERT = dreptul de a adăuga tuple.
     - Se poate aplica unui singur atribut.
  3. DELETE = dreptul de a şterge tuple.
  4. UPDATE = dreptul de a modifica tuple.
     - Se poate aplica unui singur atribut.

Exemplu: Privilegii

- Pentru instrucţiunea următoare:

```
INSERT INTO Beers(name)
   SELECT beer FROM Sells
   WHERE NOT EXISTS
        (SELECT * FROM Beers
         WHERE name = beer);
```

berile ce nu apar în Beers, se adaugă în Beers cu NULL pentru manufacturer.

- Este nevoie de privilegii SELECT asupra Sells şi Beers, şi INSERT asupra Beers sau Beers.name.

Obiecte BD

- Obiectele pentru care există privilegii includ tabele de bază şi vederi.
- Alte privilegii sunt dreptul de a crea obiecte de un anumit tip, de exemplu, trigger-e.
- Vederile formează o unealtă importantă pentru controlul accesului.

Exemplu: Vederile şi Controlul Accesului

- Se poate să nu se dorească să se acorde privilegiul SELECT asupra Emps(name, addr, salary).
- Mai sigur este să se acorde SELECT asupra:

```
CREATE VIEW SafeEmps AS
  SELECT name, addr FROM Emps;
```

- Interogările pe SafeEmps nu necesită SELECT asupra Emps, ci doar asupra SafeEmps.

ID-uri de Autorizare

- Un utilizator este referit printr-un *ID de autorizare*, de obicei “login name”.
- Există un ID de autorizare PUBLIC.
  - Acordarea unui privilegiu PUBLIC îl face disponibil oricărui ID de autorizare.

Acordarea Privilegiilor

- Pentru obiectele (de exemplu relaţiile) create de un utilizator, acesta are toate privilegiile posibile.
- Se pot acorda privilegii altor utilizatori (ID-uri de autorizare), inclusiv PUBLIC.
- Se pot acorda privilegii cu clauza WITH GRANT OPTION, ce permite celui autorizat să acorde mai departe privilegii.

Instrucţiunea GRANT

- Pentru a acorda privilegii:

  GRANT <listă de privilegii>

  ON <relaţie sau alt obiect>

  TO <listă ID-uri de autorizare>;

- Dacă se doreşte ca primitorul să fie capabil să transmită privilegiul(-ile) altora, se adaugă:

  WITH GRANT OPTION

Exemplu: GRANT

- Presupunem că utilizatorul curent este proprietar (owner) al Sells.

```
GRANT SELECT, UPDATE(price)
ON Sells
TO sally;
```

- După aceasta, Sally are dreptul să emită orice interogare asupra relaţiei Sells şi poate modifica preţul.

Exemplu: Opţiunea GRANT

- Se presupune următoarea instrucţiune:

```
GRANT UPDATE ON Sells TO sally
WITH GRANT OPTION;
```

- În urma acestei instrucţiuni, Sally are dreptul să modifice valoarea oricărui atribut al relaţiei "Sells", şi poate acorda altora privilegiul "UPDATE ON Sells".
  - De asemenea, Sally poate acorda privilegii mai specifice: `UPDATE(price)ON Sells`.

Revocarea Privilegiilor

REVOKE <listă de privilegii>
ON <relaţie sau alt obiect>
FROM <listă de ID-uri de autorizare>;

- Privilegiile specificate, acordate de utilizatorul curent nu mai pot fi folosite de utilizatorii specificaţi pentru a justifica utilizarea privilegiului.
  - Dar utilizatorii specificaţi pot avea privilegiul deoarece l-au primit de la altcineva.

Opţiuni REVOKE

- La instrucţiunea REVOKE se ataşează una din următoarele:
  1. CASCADE. Orice acordare mai departe a privilegiului se revocă pe toată lungimea căii de acordare.
  2. RESTRICT. Dacă privilegiul a fost transmis altora, REVOKE eşuează, este o atenţionare că trebuie făcute alte acţiuni pentru a îngrădi privilegiul.

Diagrame Grant

- Noduri = utilizator/privilegiu/opţiune "grant"?/este "owner"?
  - UPDATE ON R, UPDATE(a) on R şi UPDATE(b) ON R aparţin la noduri diferite.
  - SELECT ON R şi SELECT ON R WITH GRANT OPTION aparţin la noduri diferite.
- Arcul *X* -> *Y* semnifică nodul *X* a fost folosit pentru "grant" *Y*.

Notaţie pentru Noduri

- Se foloseşte *AP* pentru nodul ce reprezintă ID-ul de autorizare *A* cu privilegiul *P*.
  - *P** = privilegiul *P* cu opţiunea “grant”.
  - *P*** = sursa privilegiului *P*.
    - Adică, *A* este “owner” pentru obiectul pentru care *P* este privilegiu.
    - Notă ** implică opţiunea “grant”.

Trasarea Arcelor

- Atunci când *A* acordă privilegiul *P* lui *B*, se trasează un arc de la *AP* * la *AP* ** pentru *BP*.
  - Sau pentru *BP* * dacă acordarea privilegiului este cu opţiunea "grant".
- Dacă *A* acordă un subprivilegiu *Q* al lui *P* (de exemplu UPDATE(a) pentru R dacă *P* este UPDATE ON R) atunci arcul merge la *BQ* sau *BQ* *.

Trasarea Arcelor

- Regula fundamentală: Utilizatorul *C* are privilegiul *Q* atât timp cât există o cale de la *XP* ** la *CQ*, *CQ* *, sau *CQ* ** şi *P* este un superprivilegiu al lui *Q*.
  - *P* poate fi *Q* şi *X* poate fi *C*.

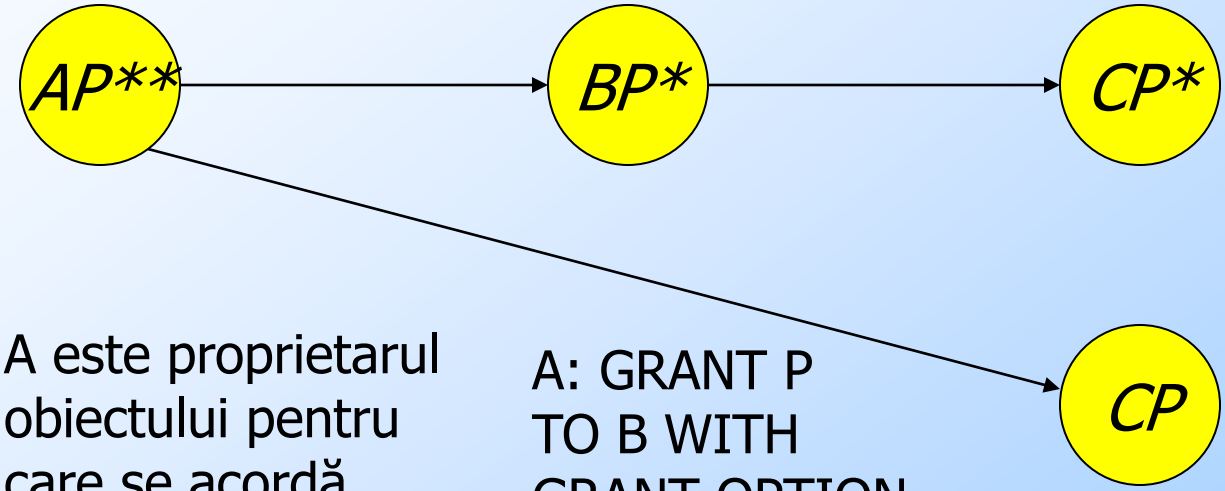
Trasarea Arcelor

- Dacă *A* revocă *P* pentru *B* cu opţiunea CASCADE, se şterge arcul de la *AP* la *BP*.
- Dacă *A* foloseşte în schimb, RESTRICT şi există un arc de la *BP* în altă parte, atunci se respinge revocarea şi nu se produce nici o modificare a grafului.

Trasarea Arcelor

- La verificarea arcelor, se testează ca fiecare nod să aibă o cale de la un anume nod **, ce reprezintă proprietarul.
- Orice nod ce nu are o astfel de cale reprezintă un privilegiu revocat şi se şterge din diagramă.

Exemplu: Diagramă "Grant"



B: GRANT P
TO C WITH
GRANT OPTION

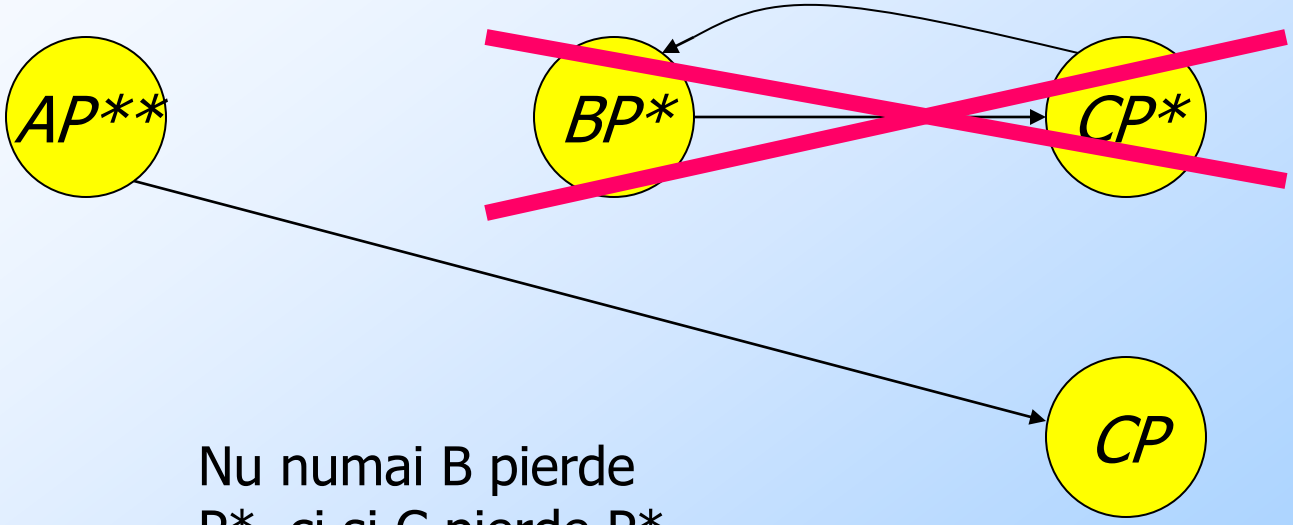
A este proprietarul obiectului pentru care se acordă privilegiul P.

A: GRANT P
TO B WITH
GRANT OPTION

A: GRANT P
TO C

Exemplu: Diagramă "Grant"

A execută
REVOKE P FROM B CASCADE;



Chiar dacă C a obţinut P de la B, ambele noduri sunt tăiate.

Nu numai B pierde P*, ci şi C pierde P*. Se şterg BP* şi CP*.

Totuşi, C mai are P fără opţiune "grant" deoarece i s-a acordat direct privilegiul.

Tranzacţii

Tranzacţii - Motivaţie

- Sistemele de BD sunt accesate în mod normal de mai mulţi utilizatori sau procese la acelaşi moment de timp.
  - Atât interogări cât şi actualizări.
- Spre deosebire de sistemele de operare, ce *susţin* interacţiunea proceselor, un SGBD are nevoie de supervizare a proceselor contra interacţiunilor ce cauzează probleme.

Exemplu: Interacţiune cu probleme

- Tatăl şi fiul posedă carduri bancare pentru acelaşi cont din bancă.
- Fiecare scoate de la ATM-uri diferite 100 LEI, în acelaşi timp.
  - SGBD-ul trebuie să asigure să nu se piardă nici una din operaţiile asupra contului.
- Comparaţie: Un SO permite ca două persoane să editeze un document în acelaşi timp. Dacă ambele scriu, modificările efectuate de una din persoane se pierde.

Tranzacţii

- *Tranzacţie* = un proces ce implică interogări şi/sau actualizări ale BD.
- În mod normal există proprietăţi puternice cu privire la concurenţă.
- În SQL este formată din instrucţiuni singulare sau control explicit de programare.

Tranzacţii ACID

- *Tranzacţiile ACID* au proprietăţile:
  - *Atomicitate* : "Totul sau nimic".
  - *Consistenţă* : Constrângerile BD să fie respectate.
  - *Izolare* : Utilizatorul vede ca şi cum la un moment dat de timp se execută un singur proces.
  - *Durabilitate* : Efectele unui proces "supravieţuiesc" unei căderi a sistemului.
- Opţional: deseori sunt susţinute forme mai slabe de tranzacţii.

COMMIT

- Instrucţiunea SQL COMMIT cauzează încheierea cu succes a unei tranzacţii.
  - Modificările asupra BD devin permanente.

ROLLBACK

- Instrucţiunea SQL ROLLBACK cauzează terminarea tranzacţiei, dar cu *abandonare*.
  - Nu există efecte în BD.
- Eşuările ca de exemplu împărţirea la 0 sau violarea unei constrângeri pot cauza rollback, chiar dacă programatorul nu a apelat-o.

Exemplu: Procese ce Interacţionează

- Presupunem relaţia Sells(bar,beer,price) şi faptul că "Joe's Bar" vinde numai "Bud" la 2.50 ($) şi "Miller" la 3.00 ($).
- Sally interoghează Sells pentru a afla preţul cel mai mare şi cel mai mic cu care vinde Joe.
- Joe decide să nu mai vândă Bud şi Miller, numai Heineken la 3.50 ($).

Program lui Sally

- Sally execută următoarele două instrucţiuni SQL numite (min) şi (max):

(max)
```
SELECT MAX(price) FROM Sells
WHERE bar = ’Joe”s Bar’;
```

(min)
```
SELECT MIN(price) FROM Sells
WHERE bar = ’Joe”s Bar’;
```

Programul lui Joe

- Aproximativ în acelaşi timp, Joe execută următorii paşi: (del) şi (ins).

(del) DELETE FROM Sells
WHERE bar = 'Joe"s Bar';

(ins) INSERT INTO Sells
VALUES('Joe"s Bar', 'Heineken', 3.50);

Intercalarea Instrucţiunilor

- Deşi (max) trebuie să apară înaintea lui (min), şi (del) trebuie să apară înaintea lui (ins), nu există alte constrângeri asupra ordinii acestor instrucţiuni, doar dacă se grupează instrucţiunile lui Sally şi/sau instrucţiunile lui Joe în tranzacţii.

Exemplu: Intercalare

- Presupunem paşii următori de execuţie: (max) (del) (ins) (min).

Preţurile lui Joe: {2.50,3.00}{2.50,3.00} {3.50}

Instrucţiune:	(max)	(del)	(ins)	(min)
Rezultat:	3.00			3.50

- Sally observă MAX < MIN!

Rezolvarea Problemei prin Utilizare de Tranzacţii

- Dacă se grupează instrucţiunile lui Sally (max)(min) într-o tranzacţie, ea nu poate vedea această inconsistenţă.
- Ea vede preţurile lui Joe la un moment fix de timp.
  - Fie înainte, fie după modificarea preţurilor, sau la mijloc, dar MAX şi MIN sunt calculate pe aceleaşi preţuri.

O altă Problemă: Rollback

- Presupunem că Joe execută (del)(ins), nu ca o tranzacţie şi după execuţia acestor instrucţiuni, se gândeşte să execute instrucţiunea ROLLBACK.
- Dacă Sally execută instrucţiunile proprii după (ins) dar înainte de rollback, ea vede o valoare, 3.50, ce nu a existat niciodată în BD.

Soluţia

- Dacă Joe execută (del)(ins) ca o tranzacţie, efectele acesteia nu pot fi văzute de alţii până ce tranzacţia execută COMMIT.
  - Dacă tranzacţia execută ROLLBACK, atunci efectele nu pot fi *niciodată* văzute.

Nivele de Izolare

- SQL defineşte patru *nivele de izolare* = opţiuni în legătură cu ce interacţiuni le sunt permise tranzacţiilor ce se execută în acelaşi timp.
- Doar un singur nivel (“serializable”) = tranzacţii ACID.
- Fiecare SGBD implementează tranzacţiile în mod propriu.

Alegerea Nivelului de Izolare

- În cadrul unei tranzacţii, se poate preciza:

SET TRANSACTION ISOLATION LEVEL *X*

unde *X* =

1. SERIALIZABLE
2. REPEATABLE READ
3. READ COMMITTED
4. READ UNCOMMITTED

Tranzacţii Serializabile

- Dacă Sally = (max)(min) şi Joe = (del)(ins) sunt fiecare tranzacţii, şi Sally execută cu nivelul de izolare SERIALIZABLE, atunci ea va vedea BD fie înainte, fie după ce Joe execută, dar nu la mijloc.

Nivelul de Izolare este o Alegere Personală

- Opţiunea aleasă de un utilizator, de exemplu "run serializable", afectează doar felul în care *acel utilizator* vede BD, nu felul în care o văd alţii.
- Exemplu: Dacă Joe execută "serializable", dar Sally nu, atunci Sally se poate să nu vadă preţuri pentru "Joe's Bar".
  - adică, Sally observă ca şi cum ar fi executat la mijlocul tranzacţiei lui Joe.

Tranzacţii Read-Commited

- Dacă Sally execută cu nivelul de izolare READ COMMITTED, atunci ea vede doar datele "committed", dar nu neapărat aceleşi date de fiecare dată.
- Exemplu: Cu READ COMMITTED, intercalarea (max)(del)(ins)(min) este permisă atât timp cât Joe face "commit".
  - Sally vede MAX < MIN.

Tranzacţii Repeatable-Read

- Cerinţa este ca şi la read-committed, plus: dacă datele sunt citite din nou, atunci tot ceea ce a fost văzut prima dată este văzut a doua oară.
  - Dar a doua oară şi citiri ulterioare pot vedea *mai multe* tuple.

Exemplu: Repeatable Read

- Presupunem că Sally execută cu REPEATABLE READ, şi ordinea de execuţie este (max)(del)(ins)(min).
  - (max) vede preţurile 2.50 şi 3.00.
  - (min) poate vedea 3.50, dar de asemenea vede 2.50 şi 3.00, deoarece aceste preţuri au fost văzute la prima citire de (max).

Read Uncommitted

- O tranzacţie ce se execută cu READ UNCOMMITTED poate vedea datele din BD, chiar dacă nu au fost scrise de o tranzacţie ce nu a efectuat committed (şi poate niciodată).
- Exemplu: Dacă Sally execută cu READ UNCOMMITTED, ea poate vedea preţul 3.50 chiar dacă Joe va abandona mai târziu.