Sisteme de Gestiune a Bazelor de Date

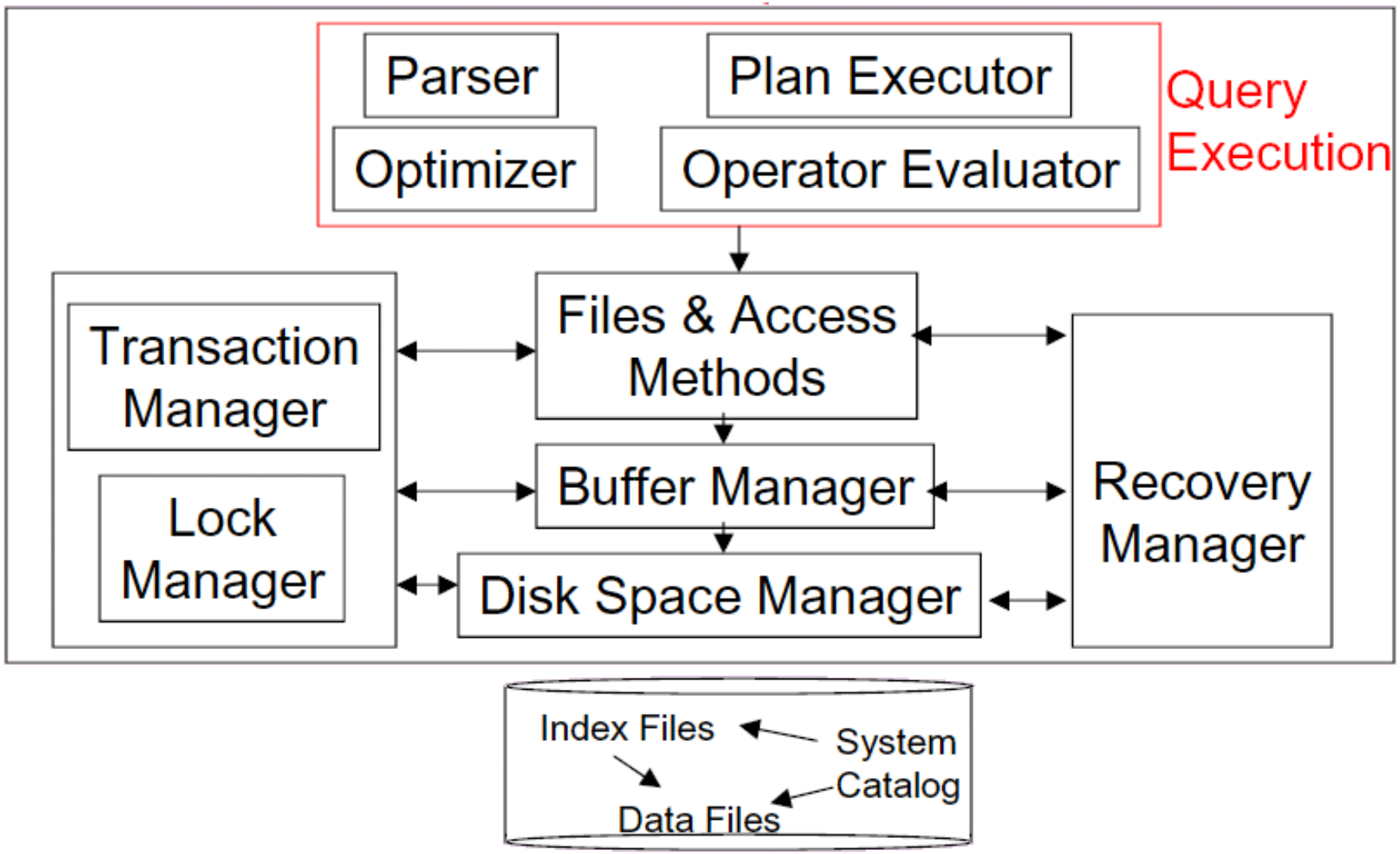
Detalii organizatorice

- Team code: **7xykd68**

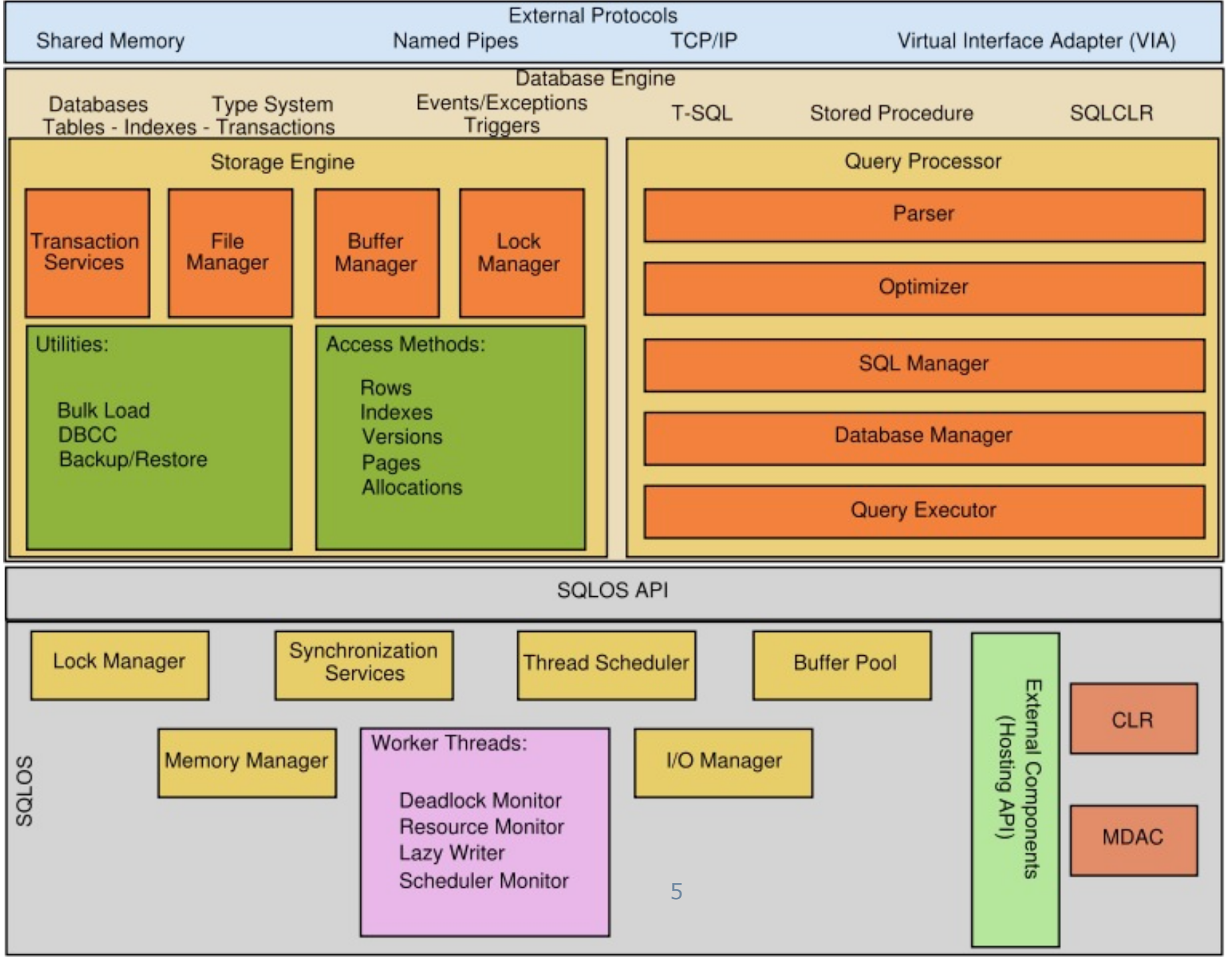
Conținut curs

- Gestiunea tranzacţiilor
- Controlul concurenţei
- Recuperarea datelor
- Sortare externă
- Evaluarea operatorilor relaţionali
- Optimizarea interogărilor
- Baze de date distribuite / paralele
- Securitate

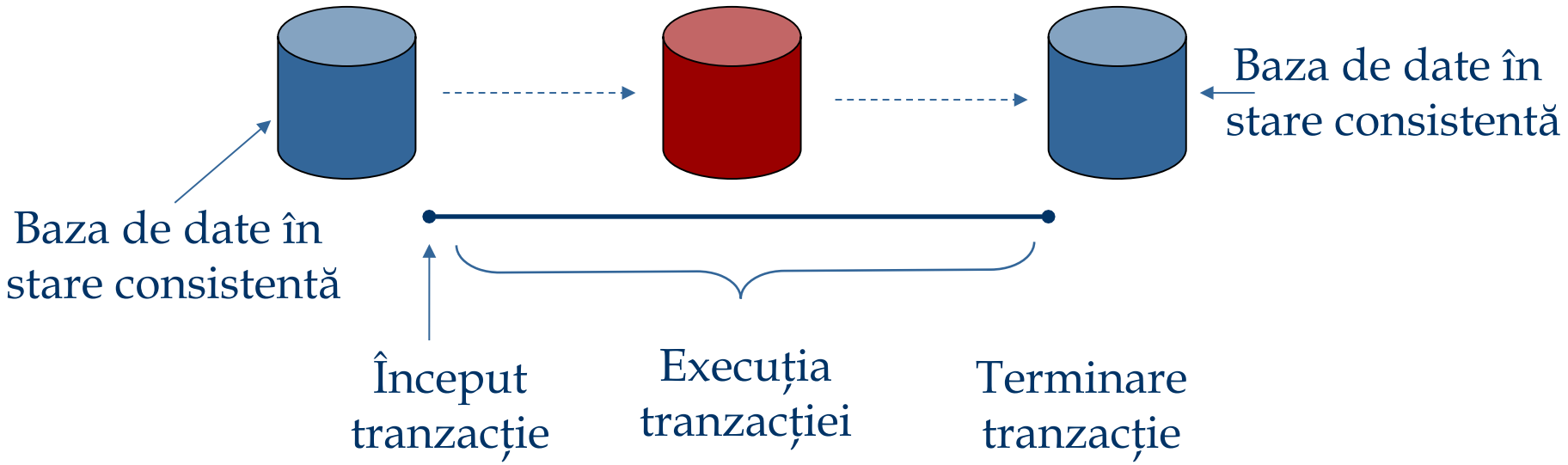
Structura unui SGBD



Structura
MS SQL Server



Tranzacții


Baza de date în stare consistentă
Baza de date în stare consistentă
Început tranzacție
Execuția tranzacției
Terminare tranzacție

Tranzacții (cont.)

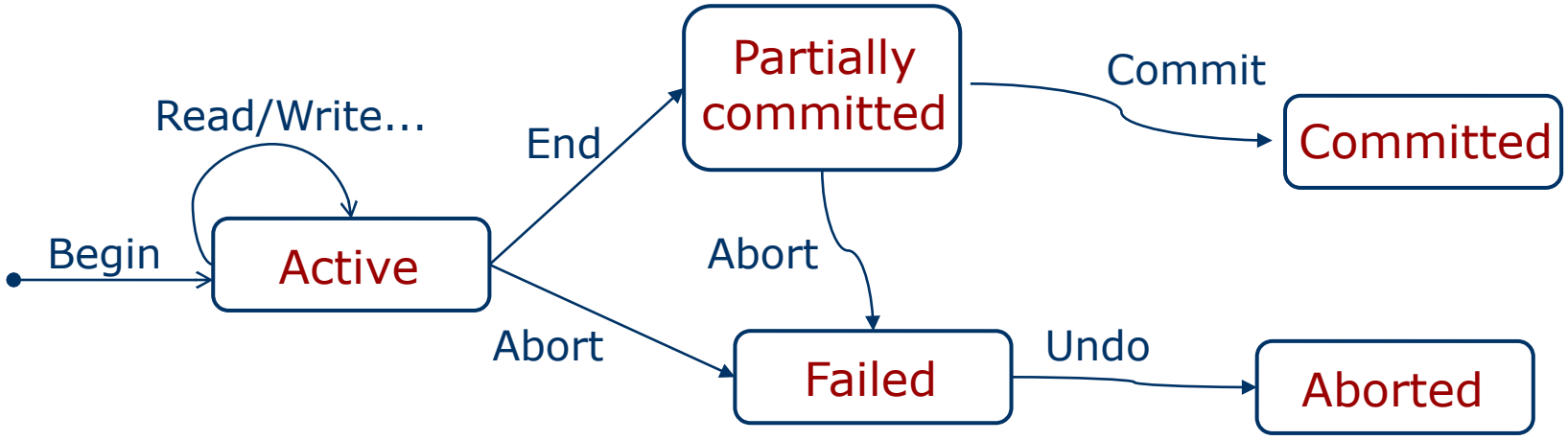
- Execuția concurentă este esențială pentru performanța unui SGBD
  - Deoarece harddisk-ul este accesat frecvent, iar accesul este relativ lent, este preferabil ca CPU-ul să fie "ocupat" cu alte task-uri executate concurent.

- SGBD-ul "vede" un program ce interacționează cu baza de date ca o secvență de operații de **citire** şi **scriere**.

Begin – transaction
Read
Write
End – transaction
Commit – transaction
Abort – transaction
Undo
Redo

Stările tranzacţiilor

Active: tranzacţia este în execuţie
Partially Committed: tranzacţia urmează să se finalizeze
Committed: terminare cu succes
Failed: execuţia normală a tranzacţiei nu mai poate continua
Aborted: terminare cu *roll back*



Concurenţa într-un SGBD

Un utilizator transmite unui SGBD mai multe tranzacţii spre execuţie:

- Concurenţa este implementată de SGBD prin intercalarea operaţiilor mai multor tranzacţii (citiri/modificări ale obiectelor bazei de date)
- Fiecare tranzacţie trebuie să lase baza de date într-o stare consistentă
  - Constrângeri de integritate (intră in responsabilitatea SGBD).
  - SGBD nu "inţelege" semantica datelor (responsabilitatea programatorului).

Probleme: Efectul *intercalării* tranzacţiilor şi *blocări*.

Proprietăţile tranzacţiilor - **ACID**

Atomicitate (*totul sau nimic*)

Consistenţă (*garantare constrângeri de integritate*)

Izolare (*concurenţa este invizibilă* $\rightarrow$ *serializabilitate*)

Durabilitate (*acţiunile tranzacţiilor executate persistă*)

Atomicitate

O tranzacţie se poate termina cu succes, după execuţia tuturor acţiunilor sale, sau poate eşua (uneori forţat de SGBD) după execuţia anumitor acţiuni.

Utilizatorii (programatorii) pot privi o tranzacţie ca o operaţie indivizibilă.

SGBD salvează in *loguri* toate acţiunile unei tranzacţii pentru a le putea anula la nevoie.

Acţiunea prin care se asigură atomicitatea tranzacţiilor la apariţia unor erori poartă numele de recuperarea datelor (*crash recovery*)

Consistenţă

O tranzacţie executată *singură* pe o bază de date consistentă, lasă baza de date într-o stare consistentă.

Tranzacţiile păstrează constrângerile de integritate ale bazelor de date.

Tranzacţiile sunt programe *corecte*

Izolare

Dacă mai multe tranzacţii sunt executate concurent, rezultatul trebuie sa fie identic cu una dintre execuţiile seriale ale acestora (indiferent de ordine) - *serializabilitate*.

Tranzacțiile sunt executate “ferrite” de efectele altor tranzactii, ca și cum ar fi executate izolat, o tranzactie este “vazuta” ca si cand baza de data ar lucre single-user mode, fara paralelism.

O tranzacţie nu poate partaja modificările operate până nu este finalizată

Condiţie necesară pentru evitarea eşecurilor in cascadă.

Durabilitate

Odată o tranzacţie finalizată, sistemul trebuie să garanteze că rezultatul operaţiilor acesteia nu se vor pierde, chiar si la apariţia unor erori sau blocări ulterioare.

Recuperarea datelor

Exemplu

```
T1:    BEGIN  A=A+100,  B=B-100  END

T2:    BEGIN  A=1.06*A,  B=1.06*B  END
```

- Prima tranzacţie transferă 100€ din contul B în contul A.
- Cea de-a doua tranzacţie adaugă o dobandă de 6% sumelor din ambele conturi.

Exemplu

O posibilă intercalare a operaţiilor (*plan*):

```
T1:   A=A+100,                    B=B-100
T2:               A=1.06*A,                 B=1.06*B
```

O a doua variantă:

```
T1:   A=A+100,                            B=B-100
T2:               A=1.06*A, B=1.06*B
```

Cum “vede” SGBD al doilea plan:

```
T1:   R(A), W(A),                         R(B), W(B)
T2:               R(A), W(A), R(B), W(B)
```

Anomalii de executie in tranzactiile intercalate

RR:

Două tranzacții doar citesc un obiect de date => fără conflict, ordinea execuției nu este importantă.

R/W (set date diferit)

Două tranzacții citesc și/sau scriu obiecte de date complet separate => fără conflict, ordinea execuției nu este importantă.

R/W (acelasi set date diferit)

Două tranzacții operează asupra aceluiași obiect de date, iar cel puțin una dintre ele efectuează o operație de scriere => ordinea execuției este important – **Conflicte!**

Conflicte:

Conflicte:
WR Conflict – T2 citește un obiect de date scris anterior de T1.

RW Conflict – T2 scrie un obiect de date citit anterior de T1.

WW Conflict – T2 scrie un obiect de date scris anterior de T1.

Anomalii ale execuţiei concurente

Reading Uncommitted Data (conflict WR, "*dirty reads*"):

T1: R(A), W(A), R(B), W(B), **A**
T2: R(A), W(A), **C**

Unrepeatable Reads (conflict RW):

T1: R(A), R(A), W(A), **C**
T2: R(A), W(A), **C**

Overwriting Uncommitted Data (Conflict WW, "*blind writes*"):

T1: W(A), W(B), **C**
T2: W(A), W(B), **C**