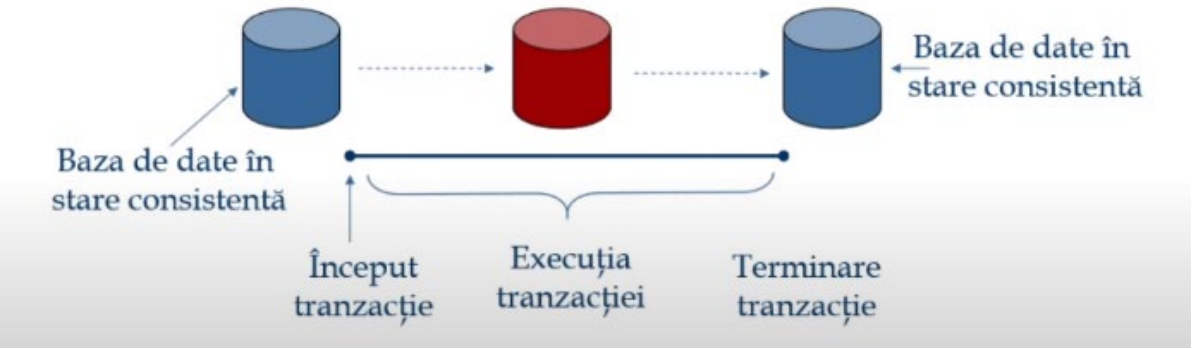
Tranzacții

O tranzacție reprezintă o secvență de mai multe instrucțiuni/operații care, împreună, sunt privite ca o singură operație în sine (pentru utilizatorii obișnuiți) → nu putem efectua tranzacții pe jumătate.

O altă proprietate a unei tranzacții este că aceasta lasă baza de date într-o stare consistentă.



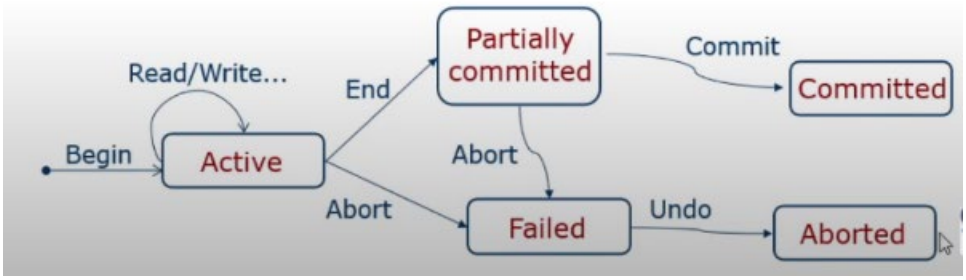
Un SGBD vede o tranzacție ca o serie de citiri și scrieri.

!!! End transaction nu specifică dacă tranzacția s-a încheiat cu succes sau nu.

!!! Undo & Redo sunt operații care sunt efectuate doar atunci când trebuie recuperate date.

Stările tranzacțiilor:

- **Active**: tranzacția este *în execuție*
- **Partially Committed**: tranzacția *urmează să se finalizeze*
- **Committed**: terminare cu *succes*
- **Failed**: execuția normală a tranzacției *nu mai poate continua*
- **Aborted**: terminare cu *roll back*



Proprietățile tranzacțiilor – **ACID**

- **A**tomicitate – fie se execută toate operațiile unei tranzacții, fie niciuna
- **C**onsistență – constrângerile de integritate trebuie să fie păstrate în continuare
- **I**zolare – atunci când un programator scrie o tranzacție, nu ar trebui să se gândească cu cine ar putea să intre în conflict; tranzacția este scrisă ca un program care rulează de sine-stătător
- **D**urabilitate – în momentul în care o tranzacție s-a executat cu succes, în baza de date trebuie să se regăsească efectele execuției acelei tranzacții

Atomicitate:

- **toate** operațiile sunt atomice: delete, insert, update
- în momentul în care o tranzacție își oprește tranzacția la mijloc, trebuie să existe un mecanism prin care să refacem contextul → SGBD salvează în *loguri* toate acțiuniile unei tranzacții, pentru a le putea anula la nevoie
- **recuperarea datelor** (crash recovery) = acțiunea prin care se asigură atomicitatea tranzacțiilor la apariția unei erori

Consistență:

- SGBD este responsabil cu păstrarea consistenței unei baze de date, însă logica după care omul face modificări la baza de date nu intră în responsabilitatea acestuia

Izolare:

- în momentul în care o tranzacție se execută în paralel cu o altă tranzacție, pe aceleași date, efectul asupra bazei de date trebuie să fie identic cu execuția serială a lor
- dacă T_1 și T_2 se execută în același timp, după ce își termină execuția, când ne uităm în baza de date, trebuie să fie ori efectul execuției $T_1 \rightarrow T_2$, ori $T_2 \rightarrow T_1$

Durabilitate:

- odată ce o tranzacție s-a executat cu succes, sistemul garantează că rezultatul operațiilor nu se vor pierde
- SGBD este responsabil cu recuperarea datelor în cazul în care o tranzacție nu s-a executat cu succes

Anomalii ale execuției concurente:

1. *Dirty Reads (Reading Uncommitted Data) – conflict WR*

   T_1: R(A), W(A), R(B), W(B), **Abort**

   T_2: R(A), W(A), **Commit**

- valoarea lui A revine la valoarea pe care a avut-o înainte de tranzacții
- dirty read-ul apare la T_2: T_2 a citit o modificare a lui T_1 care nu a persistat în baza de date, deci valoarea pe care a salvat-o este una greșită
- un utilizator care efectuează T_2, observă că tranzacția s-a efectuat cu succes, însă când merge în baza de date să vadă modificarea, vede că nu există, deoarece s-a făcut undo pentru T_1 și s-a suprascris baza de date cu valoarea de dinaintea execuției lui T_1
- nu este o operație durabilă

2. *Unrepeatable Reads – conflict RW:*

   T_1: R(A), R(A), W(A), **Commit**

   T_2: R(A), W(A), **Commit**

- fără să facă vreo modificare, T_1 va citi valori diferite pentru aceeași resursă
- nu este o operație consistentă

3. *Blind writes (Overwritting Uncommitted Data) – conflict WW:*

   T_1: W(A), W(B), **Commit**

   T_2: W(A), W(B), **Commit**

- după ce se execută tranzacțiile, A rămâne cu ce a zis T_2, iar B rămâne cu ce a zis T_1

4. *Phantom Reads:*

- se fac două select-uri, însă la al doilea, apare o înregistrare "*fantomă*", care nu a apărut în primul select
- se întâmplă când cineva face o modificare între cele două selecturi