Australian National University

Database Transactions – Part 2

ACID Properties

ACID Properties

- DBMSs ensure the following properties of transactions.
  - **Atomicity**:
    - The execution of each transaction is atomic, i.e., **either all operations are completed or not done at all**.
  - **Consistency**:
    - The states of a database are consistent (w.r.t. defined business rules) **before and after each transaction**.
  - **Isolation**:
    - Execution results of each transaction should be **unaffected by other concurrent executing transactions**.
  - **Durability**:
    - Once a transaction has been successfully completed, **its effects should persist in the database**.

Note: These properties are not independent from one another, but **atomicity is the central property**.

Australian National University

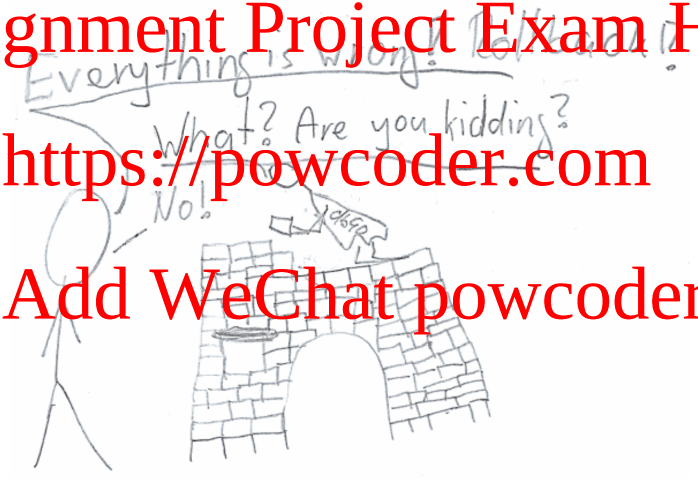
Atomicity

- **Atomicity** requires that we execute a transaction to completion with only **two possibilities**:
  - **ALL**: all the operations are executed;
  - **NONE**: none of the operations are executed.
- If a transaction fails to complete for some reason, it may leave database in an inconsistent state. Thus a DBMS **must remove effects of partial transactions** to ensure atomicity.

  Example: The money can only be taken from Steve's account if the money has been transferred into Bob's account.

Operations	Steve	Bob	
before 1	$1000	$200	**None are executed.**
after 1	$1000	$200	
after 2	$500	$200	
after 3	$500	$200	
after 4	$500	$700	**All are executed.**

Atomicity



Australian National University

Consistency

- **Consistency** requires that each transaction should **preserve the consistency of the database**.

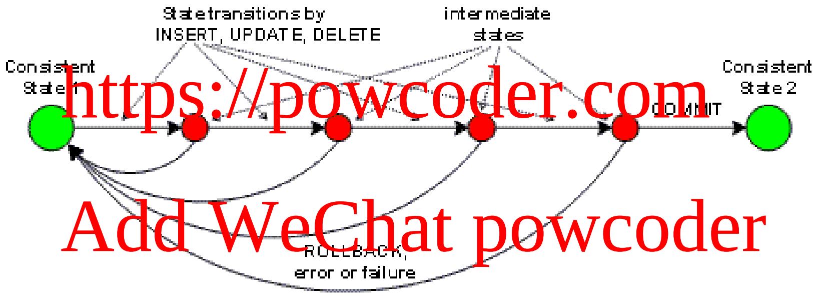
- **Note:** Intermediate states may be inconsistent.

Example: Suppose that we have

Steve's account balance + Bob's account balance = $1200,

Operations	Steve	Bob	
before 1	$1000	$200	$1000+$200=$1200
after 1	$1000	$200	
after 2	$500	$200	**Not required to be consistent.**
after 3	$500	$200	
after 4	$500	$700	$500+$700=$1200

Consistency [1]


State transitions by INSERT, UPDATE, DELETE
intermediate states
Consistent State 1
Consistent State 2
COMMIT
ROLLBACK, error or failure

- The database is in a consistent state before and after executing the transaction, but is not necessarily consistent in intermediate states.

1 The figure is taken from http://maxdb.sap.com

Isolation

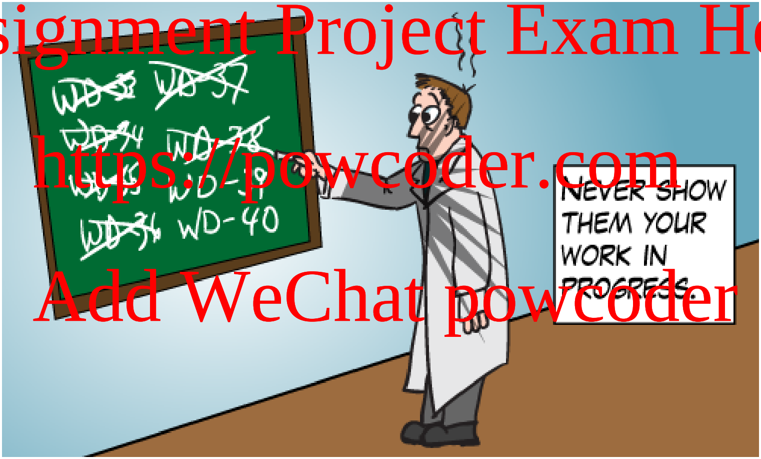
- **Isolation** requires that transactions are **isolated from one another**.

Example: Other transactions can't see the changes on objects *A* (Steve's account balance) and *B* (Bob's account balance) until the transaction for the money transfer is completed.

T_1
read(A)
write(A) (A:=A−500)
read(B)
write(B) (B:=B+500)
commit

T_2
read(A)
write(A) (A:=A+400)
commit

Isolation [2]



[2] The figure is taken from http://michaeljswart.com/

Durability

- **Durability** requires that once the transaction is successfully completed, its changes to the database **must be persistent despite failures**.

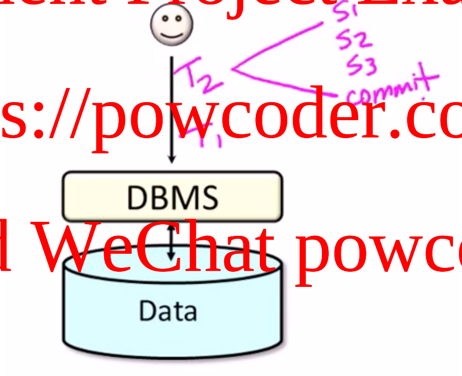
- The decision is irrevocable: once committed, the transaction cannot revert to abort. **Changes are durable**.

- **Example:** Once Steve received the notification:

  "**$500 has been successfully transferred to Bob's account**",

  the money can't go back to Steve's account and must appear in Bob's account.

Durability [3]



[3] The figure is taken from http://toyhouse.cc/profiles/blogs/the-acid-properties-of-transactions