



# Assignment Project Exam Help

## Database Transactions – Part 1

<https://tutorcs.com>

WeChat: cstutorcs



## Transaction – A Classical Example

# Assignment Project Exam Help

- Scenario: Suppose that Steve's account balance is **\$1000** and Bob's balance is **\$200**. Now Steve wants to transfer **\$500** into Bob's account.

- There are several steps involved in transferring the money:

- 1 **Check** Steve's balance;

<https://tutorcs.com>

- 2 **Update** Steve's balance;

- 3 **Check** Bob's balance;

- 4 **Update** Bob's balance.

[WeChat: cstutorcs](#)

- Steve later checked his balance (it was **\$500**), which looked good to Steve. However, Bob told Steve that he hadn't received his money yet (still **\$200** in Bob's account instead of **\$700**).

**Question:** What did happen?



## Transaction – A Classical Example

- **Reason:** Due to power outage, the system ~~stopped working just after~~ updating Steve's balance
- **Task:** Transfer **\$500** from Steve's account to Bob's account

1 SELECT balance FROM ACCOUNT

WHERE name = 'Steve';

2 UPDATE ACCOUNT

SET balance = balance-500

WHERE name='Steve';

3 SELECT balance FROM ACCOUNT

WHERE name = 'Bob';

4 UPDATE ACCOUNT

SET balance = balance+500

WHERE name = 'Bob';

Operations	Steve	Bob
before 1	\$1000	\$200
after 1	\$1000	\$200
after 2	\$500	\$200
after 3	\$500	\$200
after 4	\$500	\$700

<https://tutorcs.com>

WeChat: cstutorcs



## Transaction – A Classical Example

- We need an approach to ensure that
  - either the balances of Steve and Bob remain unchanged if the money transfer fails
  - or Steve's balance is \$500 and Bob's is \$700 if the money transfer succeeds.

1 SELECT balance FROM ACCOUNT  
WHERE name = 'Steve';

2 UPDATE ACCOUNT  
SET balance = balance-500  
WHERE name='Steve';

3 SELECT balance FROM ACCOUNT  
WHERE name = 'Bob';

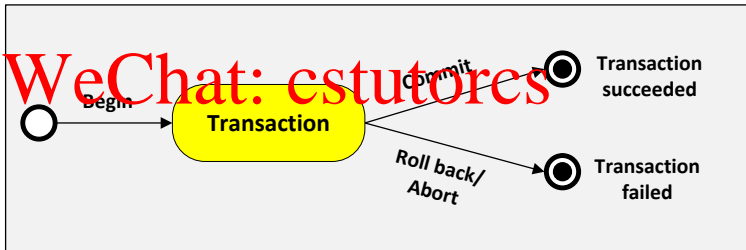
4 UPDATE ACCOUNT  
SET balance = balance+500

Operations	Steve	Bob
before 1	\$1000	\$200
after 1	\$1000	\$200
after 2	\$500	\$200
after 3	\$500	\$200
after 4	\$500	\$700



## What is a Transaction?

- DBMSs provide **transaction support** for solving this kind of problem
- A **transaction** is a sequence of database operations grouped together for execution as a **logic unit** in a DBMS.
- Different from an execution of a program outside the DBMS (e.g., a C program) in many ways!





## What is a Transaction?

# Assignment Project Exam Help

- Database applications often access a database by **transactions** rather than individual operations.

- e.g., large databases and hundreds of concurrent users: banking, supermarket checkout, airline reservation, online purchasing, etc.

- Why transactions?** They can enforce data integrity in the following situations:

- multiple users may modify and share data at the same time;
- transaction, system, and media failures may happen from time to time.

- What does a transaction look like?**

- `INSERT`, `SELECT`, `UPDATE`, `DELETE`, `BEGIN`, `COMMIT`, `ABORT` (`ROLLBACK`), etc. from a high-level language perspective;
- `read`, `write`, `begin`, `commit`, `abort` at the internal process level.



## Transaction – Language Level

Database operations of a transaction (at the SQL language level) may include: **SELECT**, **INSERT**, **UPDATE**, **DELETE**.

- Other operations: **BEGIN**, **COMMIT**, **ABORT** (**ROLLBACK**)

**BEGIN TRANSACTION**

1 **SELECT** balance **FROM** ACCOUNT **WHERE** name = 'Steve';

2 **UPDATE** ACCOUNT

**SET** balance = balance-500 **WHERE** name='Steve';

3 **SELECT** balance **FROM** ACCOUNT **WHERE** name = 'Bob';

4 **UPDATE** ACCOUNT

**SET** balance = balance+500 **WHERE** name = 'Bob';

**COMMIT**



## Transactions - Internal Process Level

- **Basic operations** of a transaction (at the internal process level) are
  - **read(X)**: loads object X into main memory;
  - **write(X)**: modifies in-memory copy of object X (and writes it to disk later on);

- **Granularity of objects**: tables, rows, cells, or memory pages,

- **Other operations**:

- **begin**: marks the beginning of a transaction;
- **commit**: signals a successful end of the transaction - all changes can safely be applied to the database permanently;
- **abort**: signals the transaction has ended unsuccessfully - undo all operations of the transaction.





## Transactions - Internal Process Level

# Assignment Project Exam Help

```
T: BEGIN TRANSACTION
T: SELECT balance FROM ACCOUNT WHERE name = 'Steve';
T: UPDATE ACCOUNT SET balance = balance-500 WHERE name='Steve';
T: SELECT balance FROM ACCOUNT WHERE name = 'Bob';
T: UPDATE ACCOUNT SET balance = balance+500 WHERE name = 'Bob';
T: COMMIT;
```

Objects

- A - Steve's account balance;
- B - Bob's account balance.

Steps	T
1	read(A)
2	write(A) (A:=A-500)
3	read(B)
4	write(B) (B:=B+500)
5	commit