Transaction is single move in database

- Different than a select or update statement
- Ex. Moving money from one account to another (that would be 2 update statements)
- When those updates are done together, collectively that’s called a transaction

Benefits of relational model

- Works well w/ highly structured data

Ways that relational database increases efficiency

- Indexing
- Directly controlling storage
- Column oriented storage vs row oriented storage
    - Ex. Amazon - every transaction is a row in a table
    - Column oriented storage - all data from 1 column stored, then next, then next
        - Benefit is that for really large datasets, you’re typically only thinking abt a few attributes. So you don’t need to call the entire table.
- Caching / prefetching
    - Storing frequently accessed data in temporary storage (cache)
- Materialized views
    - Select statement that looks like a view

Transaction

- Sequence of one orem ore operations performed as a single, logical unit of work
- Either entire sequence succeeds (commits) or fails (rollback, abort)

ACID Properties

- Atomicity
    - Transaction is treated as an atomic unit: fully executed or not executed at all
- Consistency

    - Transaction takes database from one consistent state to another consistent state (all data meets integrity constraints)
- Isolation
    - One transaction’s execution should not negatively effect execution of another
    - If T1 is reading the same data that T2 is writing, could result in:
        - Dirty read
        - non repeatable read
        - phantom read
- Durability
    - Once transaction is completed and committed, its changes are permanent
    -

What happens if you’re halfway through processing data and there’s system failure?

- Any committed transactions are preserved

Dirty read:

- Transaction T1 is able to read a row that has been modified by T2 but hasn’t executed a commit
    - If T1 is writing/updating data, no other T should be able to read it

Non repeatable read

- 2 queries in single transaction execute a select but get different values b/c T2 has changed data and committed
- 2 transactions are reading, thats fine, its only a problem if one of them is updating

Phantom read

- When T1 is running and T adds or deletes rows from set T1 is using
- You end up reading a new row in the middle of my transaction

Isolation is ensured through locking

- Databases can lock data at different levels of granularity
- So if this transaction is updating transactions table, it can lock it

- Can lock a table for writes, lock a table for reads, for both
- Different levels of granularity: row level locking, etc.