14. (Relational Database ACID Transactions)

(Transactions).

* A collection of queries
* One unit of work

Account_ID	BALANCE
1	$900
2	$600

* Send $100 from Account1 to Account2

BEGIN TX1

```
SELECT BALANCE FROM ACCOUNT WHERE ID =1

BALANCE > 100

UPDATE ACCOUNT SET BALANCE = BALANCE -100
                                   WHERE ID = 1

UPDATE ACCOUNT SET BALANCE = BALANCE +100
                                   WHERE ID =2
```

COMMIT TX1

[Atomicity] : All queries must succeed. If one fails all should rollback.

[Durability] : Committed transactions must be persisted in a durable non-volatile storage

(Isolation)

(Isolation Read Phenomena)

[Dirty Reads] :

Reading a value that hasn't been committed by another trasaction.

SALES

PID	QNT	PRICE
Product1	15	$5
Product2	20	$4

Transaction1 | Transaction2

BEGIN TX1

SELECT PID, QNT*PRICE FROM SALES

Output: Product1 , 75
Product2 , 80

While that is happening Transaction2 updates quantity of Product1

BEGIN TX2

UPDATE SALES
SET QNT=QNT+5
where PID=1

Not Consistent

SELECT * From SALES
where PID=1

Product1	20	5

COMMIT TX2

COMMIT TX1

[Non- Repeatable Read]: Same as dirty Read.

Some other transaction, writes between an ongoing transaction, and now the 2nd read value, does not match the ~~20~~ 1st one

Only diff is that ~~Tx~~ TX2 committed in this Case than Dirty Read

[Phantom Read]:

SALES

PID	QNT	PRICE
Product1	10	$5
Product2	20	$4

Transaction1 | Transaction2

BEGIN TX1

SELECT PID, QNT*PRICE FROM SALES

Product1, 50
Product2, 80

SELECT Sum (QNT*PRICE) FROM SALES.

COMMIT TX1.

BEGIN TX2

INSERT INTO SALES VALUES (New Row for Product 3)

COMMIT TX2

This would account for the new Row as well which it should not have.

[Isolation levels → For In flight transactions]

These are levels that are implemented by databases to fix those Read Phenomena, that we saw in previous slide.

(fastest)

- [Read Un Committed] → Lowest level, Literally provides no isolation, any change from the outside is visible to the transaction. [No Isolation Basically].

- [Read committed] → Little bit of Isolation, Each query in a transaction only sees committed stuff.

- [Repeatable Read] → Nice Isolation. Each query in a transaction only sees committed updates at the ~~beg~~ beginning of the transaction.

[Apply Lock Read Lock to the rows or table]

But versioning is better, It is optimistic lock [stamped Lock]

- Serializable : Transactions cannot be executed in parallel, they have to be executed one by one

⇓

(Slowest)

(fastest to slowest)

Isolation Level	Dirty Read	Lost updates	Non-Repeatable Reads	Phantom Reads
Read Uncommitted	may occur.	may occur	may occur	may occur.
Read Committed	Safe	may occur.	may occur.	may occur
Repeatable Read	Safe	Safe.	Safe	may occur, Since adding new row has no lock
Serializeable	Safe	Safe	safe	safe

[Consistency]

[Consistency in Read] (Eventual Consistency)

*(Consistency in Data) (All replicas have same data)

this stores which user liked which picture.

Table1: Pictures ID to likes Map.

ID	Likes
1	2
2	1

⋮

Picture_Likes

USER	PICTURE_ID

Inconsistency Among these two tables is okay. As nobody is going to ~~[illegible]~~ match the follower count to likes count