=> Handling failures and maintaining durability is done by 2 keywords => COMMIT and ROLLBACK

1) **COMMIT** :- If the entire transaction goes on properly then to save the changes, we do commit.

2) **ROLLBACK** :- If anything goes wrong or we are not able to move forward with the transaction, then to go back to the initial state and undo all changes, we do **ROLLBACK**

How transactions handle concurrency :-

→ Read uncommitted

→ Read committed

→ Repeatable Read

→ Serializable

becomes more strict

default in MySQL

C → Create → Write

R → Read → Read

U → Update → Write

D → Delete → Write

=> READ UNCOMMITTED :-

got cashback (500) T1	X = 2000	purchase (1000) T2
READ(X) → 2000		
		X = 2000
		READ(X) → 2000
		X = X − 1000
		WRITE(X) ← 1000
READ(X) → 1000 –	– – –	
X = X+500		ROLLBACK
WRITE(X) → 1500		
COMMIT		

X = 1500

whereas actual should have been 2500

In Read Uncommitted, as the name suggests, another transaction can read uncommitted changes of another transaction.

RUC allows dirty reads.

→ reading uncommitted data that might / might not get committed.

RUC , since it doesnt take any lock, it's highly optimised. We can use it for places where dirty reads will not be an issue.

=> READ COMMITTED : As the name suggests we only read committed data.

got cashback (500) | p rchase (1000)

X = 2000

T_1	T_2
READ(X) → 2000	
	X = 2000
	READ(X) → 2000
	X = X-1000
	WRITE(X) ← 1000
READ(X) → ~~1000~~ 2000	
	ROLLBACKS
X = X+500	
WRITE(X) → ~~1500~~ 2500	
COMMIT	

X = ~~1500~~ 2500 .

=> REPEATABLE READ :-

T1	T2
eg) Students => 100	
(I) i) read and collect all email ids for students → 100	
(II) ii) send them updates on mail ✓	
	(III) i) added a new Student COMMIT
(IV) iii) mark all rows as update sent →	

update → read all rows → 101

↓

We want to read the same data that we read in step (I), also in step (III)

To read the same data within a transaction, we will use REPEATABLE READ → default isolation level in mysql

X = 1000

T1

READ(X) = 1000

X = X+500

X = 1500

WRITE(X)

↓

X = 1500

READ(X) = 1000

X = X+200

WRITE(X, 1200)

COMMIT

X = 1200

X = 1700 ✓

4) SERIALIZABLE:

- uses lock behind the scenes
- if one transaction has a lock over the data, no other transactions will be allowed to even read that row.
- each transaction will have to wait until another trans. gets over.
- As locks are involved, system will be slower.

* How serializable works :

- locks the rows and wont allow any reads if I am updating the row.
- it will read (just read) the data before the transaction.
- it will never read for update