Databases & Persistence (Tester View)

This document explains databases **in a child-simple way**, but focused on **real bugs testers face in production**. No DBA theory. Only what QA must understand.

---

1. What Does “Data Persisted” Mean?

Persisted means:

> Data is safely saved and will not disappear.

Simple Example

- Writing on **paper** → stays forever
- Writing on **sand** → wave comes, gone

When an API says **success**, data should be written on **paper**, not sand.

---

2. Transactions (Commit vs Rollback)

A transaction is like a **deal**.

Commit = Save

- Everything went well
- Data is permanently stored
- Cannot be undone easily

Example:

- Money paid
- Chocolate received
- Deal completed

Rollback = Undo

- Something went wrong
- Everything is cancelled
- No data is saved

Example:

- Power cut during payment
- Money returned
- No chocolate

Tester Rule

- **Commit = data exists**
- **Rollback = no data exists**

---

3. Read-After-Write Problem (Very Common Bug)

Child Story

1. You write your name in one notebook
2. You immediately open another notebook
3. Your name is not there

Why?

- You checked the wrong notebook
- Or writing is still happening

System Reality

- Data written to **Main DB**
- Data read from **Replica DB**
- Replica is slow

So API says success, but DB check shows nothing **yet**.

---

4. What DB Checks Should a Tester Do After API Success?

If API response says:

> "Order created successfully"

Tester should verify:

- Order record exists
- Correct **order_id**
- Correct **user_id**
- Correct **amount**
- Correct **status** (CREATED / PROCESSING)
- Recent **timestamp**

Tester mindset:

> Don’t just check row exists. Check **correct row exists**.

---

5. API Returned Success but DB Has No Record – Possible Reasons

This is a **real production scenario**.

Reason 1: Transaction Rolled Back

- Database error
- Validation failed
- Exception happened

API responded success **before failure**.

Reason 2: Async Processing

- API returned **202 Accepted**
- DB write happens later

- Tester checked too early

This is not a bug. This is bad testing timing.

Reason 3: Wrong Database Checked

- Wrong environment
- Checking replica instead of main DB
- Old database connection

Very common human mistake.

Reason 4: Response Sent Before DB Insert

Bad API design:

1. API sends 200 OK
2. Tries DB insert later
3. Insert fails

Result:

- Success response
- No DB record

Reason 5: Duplicate / Idempotency Handling

- Same request sent twice
- Backend blocks second insert
- Tester expects new record

System is correct, expectation is wrong.

---

6. Tester Golden Rules

- API success ≠ Data persisted

- Always ask: **sync or async?**
- Verify DB **after correct delay**
- Check logs if API and DB mismatch

Final Takeaway

Most critical bugs live:

> Between API response and database write

Understanding this makes you:

- Better API tester
- More respected by backend teams
- Stronger in production debugging

This knowledge is mandatory before advanced API testing.