**Question:**

“Before we say COMMIT, where exactly are all the INSERT, UPDATE, or DELETE changes stored?”

**Simple Answer:**

Before COMMIT, all your changes are **not yet permanently written to the actual database table on disk**.  
Instead, they are stored **temporarily in memory (a buffer area)** — managed by the **database engine or transaction log**.

[ **every major database system (MySQL, PostgreSQL, SQL Server, Oracle, etc.)** maintains some form of **temporary memory area and/or transaction log** to keep track of all changes **before the COMMIT**.

But…  
each database engine uses **its own internal mechanism** to manage this — the **concept is the same**, but **implementation differs**. ]

Think of it like writing on a **scratch pad** — your changes are visible to *you* during the transaction, but not to others yet.

Once you say **COMMIT**, the database takes everything from that temporary area and **writes it permanently** to the real data files on disk.

**Step-by-step View:**

1. You run an INSERT, UPDATE, or DELETE.
   * The database does **not immediately update the actual table on disk**.
   * It stores those changes in a **temporary buffer (memory)** or **transaction log**.
2. Only your current transaction (or session) can see those uncommitted changes.
   * Other users connected to the same database **cannot** see them until you commit.
3. When you call COMMIT,
   * All buffered changes are **flushed to the physical table files**.
   * The data becomes **visible to all users** and **permanent**.
4. If you call ROLLBACK instead,
   * The database simply **discards** those temporary changes.
   * It’s as if the operations never happened.

**Real-life Analogy:**

Imagine filling out an **online form**:

* You type your data (insert/update) — this is like working in a transaction buffer.
* But until you **click “Submit” (commit)**, your data isn’t actually saved to the server.
* If you **click “Cancel” (rollback)**, your changes vanish.

**Example in SQL**

START TRANSACTION;

UPDATE students SET marks = 90 WHERE id = 1; -- change only in memory

-- Check from another session: it won’t see this change yet!

COMMIT; -- now it's saved permanently to disk and visible to all

**Example in SQLAlchemy (Python)**

from sqlalchemy.orm import Session

with Session(engine) as session:

s1 = session.get(Student, 1)

s1.marks = 90 # still in memory, not permanent

session.commit() # now permanent in DB

Before the commit, if you close the program or call rollback, the change will **disappear**.

**In short:**

Before COMMIT:

* Changes are in **temporary memory / transaction log**.
* Only your session can see them.
* They are **not written to the actual table file on disk**.

After COMMIT:

* The changes are **flushed to disk**,
* Made **visible to everyone**,
* And become **permanent**.