Difference between WAL flush and dirty page flush

confusion between WAL flush and dirty page flush, and go through a clean, ordered step-by-step flow to explain exactly what happens first, why, and when.

High-Level Principle:

PostgreSQL uses "Write-Ahead Logging (WAL)"

 Which means:
 WAL must be flushed to disk BEFORE any dirty page is flushed to disk.
This ensures **durability** and **crash recovery** — if the system crashes, PostgreSQL can replay WAL and restore consistent state.

Query Execution Flow (INSERT/UPDATE/DELETE)

Let's take the example:

```
UPDATE employees SET name = 'Gaurav' WHERE emp_id = 1001;
```

✅ Step-by-Step Internal Flow:

Step	Description	Memory/Disk Area
1	Query reaches backend	Client → PostgreSQL
2	Planner/Optimizer generates plan	Memory
3	Page needed for UPDATE is checked in shared_buffers	Memory
4	If page is not in shared_buffers, load it from disk	Disk → shared_buffers
5	PostgreSQL updates the tuple in shared_buffers	In-memory page
6	Page is now marked as dirty (has unsaved changes)	Shared_buffers
7	WAL record is generated for the update	WAL buffer (in memory)

Step	Description	Memory/Disk Area
8	**WAL is flushed to disk** in pg_wal/ via WAL Writer or immediately during COMMIT	Disk (pg_wal)
9	**COMMIT happens only after WAL is safely flushed to disk** (fsync)	Disk
10	Dirty page still in shared_buffers (not yet written to data file)	Memory
11	**Later**, Checkpointer writes dirty pages to actual table file on disk	Disk (heap)

❗ What Comes First?

Action	Timing	Why?
WAL Flush	**Before COMMIT**	To ensure changes are durable and can be recovered
Dirty Page Flush (Checkpointer)	**After COMMIT (later)**	For performance – actual data writes are delayed

---

Important Concepts:

Concept	Description
Dirty Page	A modified page in shared_buffers not yet written to table file
WAL Flush	Happens on COMMIT – WAL record must be written to disk first
Checkpointer	Periodically writes dirty pages to disk (not immediately)
Crash Recovery	Uses WAL to redo changes if dirty pages weren't written yet
fsync	Ensures data is physically written to disk (not cached by OS)

Example Timeline Visualization

```
Time      Action
----      ----------------------------
T0        Query executed
T1        Page loaded in shared_buffers (if needed)
```

```
T2        Row updated in shared_buffers
T3        Page marked dirty
T4        WAL record created
T5        WAL flushed to disk (pg_wal)
T6        COMMIT acknowledged ✓
T7        Dirty page remains in memory
T8        Checkpoint triggers
T9        Dirty page written to table file on disk
```

Summary: Final Answer

WAL is flushed to disk FIRST, even **before COMMIT is acknowledged** to the client.

Dirty page is flushed LATER, by checkpointer or background writer.

WAL is the **source of truth** for crash recovery.

PostgreSQL relies on this **"log first, write later"** method for data safety and speed.