Crash Recover Database

1️⃣ What WAL files normally do

- WAL files are continuous logs of all changes in the database.
- Every insert, update, delete is written into a WAL file.
- WAL files are usually 16 MB each, so when one fills up, PostgreSQL moves to the next WAL file.

2️⃣ What a checkpoint is

- A checkpoint is like saving all changes from memory to the real database files.
- It ensures that all previous changes are safely stored on disk.

3️⃣ Checkpoint and WAL

- When a checkpoint happens, PostgreSQL does not create a new WAL file just for the checkpoint.
- Instead, it writes a special "checkpoint record" inside the current WAL file.
- This special record says:

"All changes before this point are safely written to the database files."

- PostgreSQL also updates pg_control to remember the LSN (Log Sequence Number) of this checkpoint.

So basically:

Before checkpoint:
WAL files: [000000010000000000000016] → recording normal changes

During checkpoint:
WAL files: [000000010000000000000016-CHECKPOINT] → special marker added

The file is the same WAL file, only a checkpoint marker is added inside it.

Note: Checkpoint does not create a new WAL file. It just adds a special record inside WAL.

4️⃣ Why this matters

On crash recovery, PostgreSQL looks for this checkpoint record and the LSN in pg_control.

It knows:

"I only need to replay WAL after this checkpoint."

Everything before this checkpoint is already safely in data files (base/).

Simple analogy

Think of WAL files as a **running diary**:

WAL file	Analogy
Normal WAL	You write down every change you do in your diary.
Checkpoint record	You put a bookmark in your diary: “Everything before this is saved in the notebook.”

You don’t need a new diary page for the bookmark; you just mark the current page.

Exmaple:-

1 Setup

- Suppose your PostgreSQL data directory is:

/var/lib/pgsql/15/data

Inside it:

```
base/       → Actual table and index files
pg_wal/     → WAL (Write-Ahead Log) files
global/pg_control → Stores last checkpoint info
```

2 You run some transactions

Example:

```
INSERT INTO employee VALUES (1, 'Gaurav', 'DBA');
INSERT INTO employee VALUES (2, 'Ravi', 'Developer');
UPDATE employee SET role = 'Senior Developer' WHERE id = 2;
```

- What happens internally:

- Changes are first written in memory (shared buffers).

- PostgreSQL writes WAL records in pg_wal before applying them to the actual table files.

Example WAL file content:

```
000000010000000000000016
 ├─ Insert row (1, 'Gaurav', 'DBA')
 ├─ Insert row (2, 'Ravi', 'Developer')
 └─ Update row id=2, role='Senior Developer'
```

- WAL ensures that even if the database crashes now, these changes can be recovered.

3 Checkpoint occurs

- Suppose a checkpoint is triggered:
- PostgreSQL flushes all dirty memory pages to the actual table files in base/.
- Adds a checkpoint record inside the current WAL file (same file, no new file created).
- Updates pg_control with checkpoint LSN, e.g., 0/1600000.

After checkpoint:

pg_wal/
└─ 000000010000000000000016 ← WAL records + checkpoint record

Checkpoint record means:

"All changes before this LSN are safely stored in the actual table files."

4 More transactions

- You insert more rows:

```
INSERT INTO employee VALUES (3, 'Anita', 'Manager');
```

- This goes to current WAL file or a new WAL file if the old one is full.

WAL files now look like:

```
000000010000000000000016  ← Previous records + checkpoint
000000010000000000000017  ← New changes (insert Anita)
```

5 Crash happens

- Suppose the server crashes before the new changes are fully written to base/.
- Database files may not have the latest data, but WAL has all changes safely.

6 Recovery after restart

- When PostgreSQL starts:
- Reads pg_control → last checkpoint LSN = 0/1600000.
- Opens WAL files starting from that checkpoint LSN (000000010000000000000016).
- Replays all WAL records after checkpoint:

```
000000010000000000000017
 ├─ Insert row (3, 'Anita', 'Manager')
```

- After replay, database is fully consistent again.

✅Key Points From Example

- WAL files are physical files in pg_wal, created automatically.
- Checkpoint record is a special marker inside WAL, not a new file.
- pg_control stores the LSN of last checkpoint, so PostgreSQL knows where recovery starts.
- Recovery = replay WAL records from checkpoint LSN to bring database consistent.

🧠 Analogy

- WAL = diary of all changes
- Checkpoint record = bookmark: “Everything before this is safe”
- Recovery = start reading diary from the bookmark to catch up

If you want, I can draw a simple diagram showing:

Transactions → WAL → Checkpoint → Crash → Recovery