Checkpoint in PostgreSQL

A **checkpoint** is a crucial event in PostgreSQL that ensures data consistency and reduces recovery time in case of a crash. It works by writing all dirty (modified) pages from the shared memory (buffer cache) to disk and marking a point in the WAL (Write-Ahead Log) where recovery can begin.

Key Features of Checkpoints:

1. **Ensures Durability:** Flushes dirty pages to disk, making sure committed transactions are permanently stored.
2. **Speeds Up Crash Recovery:** The database can start recovery from the last checkpoint instead of replaying the entire WAL.
3. **Controlled via Parameters:**
    1. checkpoint_timeout: Maximum time interval between checkpoints.
    2. checkpoint_completion_target: Percentage of time between checkpoints used to spread out writes.
    3. checkpoint_segments (before PostgreSQL 9.5) / max_wal_size (newer versions): Controls the maximum WAL size before a checkpoint triggers.

Checkpoint Process:

1. Flush all modified data pages to disk.
2. Write a special checkpoint record in the WAL.
3. Remove old WAL files (based on retention settings).

==========

LSN (Log Sequence Number)

A **Log Sequence Number (LSN)** is a unique identifier for each WAL record in PostgreSQL. It represents the exact position in the WAL and helps track database changes efficiently.

Key Features of LSN:

1. **Used in Replication:** Determines how far a replica has caught up with the primary.
2. **Helps in WAL Archiving & PITR:** Ensures recovery starts from a known LSN.
3. **Improves Crash Recovery:** PostgreSQL can replay WAL from a specific LSN to recover data.

LSN Format & Example

- LSN is represented in **HEX format**: 00000002/000000D0
    - **First part (**00000002**)**: WAL segment number.
    - **Second part (**000000D0**)**: Offset within that WAL segment.

Commands to Check LSN:-

Current WAL LSN:
SELECT pg_current_wal_lsn();

Last Checkpoint LSN:
SELECT checkpoint_lsn FROM pg_control_checkpoint();

LSN of a Table:
SELECT pg_relation_size('my_table');

==========

Relation Between LSN (Log Sequence Number) and WAL (Write-Ahead Log) File in PostgreSQL

LSN (Log Sequence Number) is a **pointer** to a specific location in the WAL (Write-Ahead Log). Each WAL file stores database changes, and every record inside the WAL has an associated LSN.

1. WAL File Naming and Structure

PostgreSQL stores WAL files in the pg_wal (or pg_xlog in older versions) directory. Each WAL file has a **fixed size** (usually 16MB by default) and follows a specific naming pattern:

WAL File Name Format:

000000010000000020000003C

This name consists of:

- **Timeline ID (00000001)** → Identifies the current timeline.
- **WAL Segment Number (00000002)** → Corresponds to the first part of the LSN.
- **WAL Offset (0000003C)** → Identifies the WAL file in sequence.

Each WAL file covers a range of LSNs. For example:

- If **one WAL file covers 16MB**, then:
  - **First LSN in the file:** 00000002/00000000
  - **Last LSN in the file:** 00000002/01000000

2. How to Find Which WAL File Contains a Specific LSN

You can determine which WAL file a given LSN belongs to using:

SELECT pg_walfile_name('00000002/000000D0');

Example Output:
000000010000000020000003C

This means the WAL file named 000000010000000020000003C contains changes related to the specified LSN.

3. How to Find the LSN of the Last WAL Record

You can check the latest LSN using:

SELECT pg_current_wal_lsn();

Example Output:
0/16B4F78

This indicates the current WAL insert position.

4.LSN to WAL File Mapping Example:

If you get an LSN like 00000002/000000D0, the corresponding WAL file name can be derived using:

SELECT pg_walfile_name('00000002/000000D0');

which might return something like:
000000010000000020000003C

This tells you that all changes related to this LSN are recorded inside 000000010000000020000003C.

5. LSN to WAL File Mapping Example

If you get an LSN like 00000002/000000D0, the corresponding WAL file name can be derived using:

SELECT pg_walfile_name('00000002/000000D0');
which might return something like:
000000010000000020000003C

This tells you that all changes related to this LSN are recorded inside 000000010000000020000003C.

6.Check the Last Checkpoint LSN:-

SELECT checkpoint_lsn FROM pg_control_checkpoint();

This means PostgreSQL will start recovery from **LSN** 0/16B4A20 in case of a crash.

7. How LSN is Used in Replication

On a standby server (replica), check the last received LSN:

SELECT sent_lsn, write_lsn, flush_lsn, replay_lsn FROM pg_stat_replication;

```
 sent_lsn   | write_lsn  | flush_lsn  | replay_lsn
------------+------------+------------+------------
 0/16B4F78  | 0/16B4F78 | 0/16B4F78 | 0/16B4F00
```

- sent_lsn → Last LSN sent to the replica.
- write_lsn → Last LSN written on the replica.
- flush_lsn → Last LSN flushed to disk on the replica.
- replay_lsn → Last LSN replayed on the replica.

If replay_lsn is behind sent_lsn, it means replication **lag** exists.

Conclusion

- LSN **points** to a specific position in WAL.
- pg_walfile_name() **maps** an LSN to a WAL file.
- WAL files store database changes **sequentially**.
- LSNs are crucial for **replication** and **recovery**.