Replication2:
FSImage: Complete snapshot of meta data.
Lets say we have a library and we have books placed randomly in the shelf. Say someone read a book and placed it at a different location from where they picked up, we will again capture the entire data. In FSImage, we are capturing a lot of redundant data, which is not at all necessary. That's why we have this EditLogs, whenever a position of the book is changed only that's tracked. When the book say changed from place p1 to p2, then we will only keep track of that and put that back into p1.

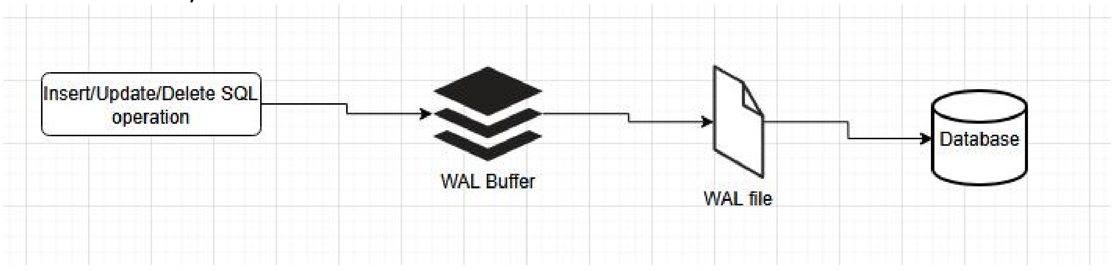
What are Replication logs?
Logs are maintained for sending the changed info

Types of logs

1. Statement based logs/Replication:
Update Users
Set name="Akshay"
Where id=101
that's the update query for Leader. We can have Update/Insert/Delete queries all in Leader, then we capture what was changed in Leader in the LOGS then forward the Logs to make changes in Followers. The downside is that, say if we use some function like now() or rand(), they could differ in the Followers.
now() --> TimeSensitive
rand() --> Different data

MySQL used to use this type of Logs approach for Replication prior to 5.1 version

2. WAL (Write Ahead Logs):
When Insert/Update/Delete operation happens, we wont directly change our Database.
I/U/D operation ---> we create a WAL buffer ---> saves in WAL file ---> This works like memory and from this memory we will store our data into Database



Even if our Leader crashes, we will have backups of all queries performed on Leader in this WAL Database and we can recover from that. We can replicate queries from this store onto other systems.
WAL file holds only the changes not the exact query
Set age=30 where id=101 ==> Query
WAL file we are saving like this ==>
Modify page#123 at offset
: write 30
Downside:
- Maintaining the change in physical location
- Tightly coupled with our database storage engine that means if storage format changes between different databases then it is of no use. Storage engine should have the same configuration
- If the Leader and Follower are using different versions then it is of no use

3. Logical replication / Row-based replication:
{ table: "orders",
  action: "INSERT",
  data: {"order_id"=123, item="Shirt"},
  timestamp: "2025-12-04T18:00:00Z",

```
},
{
…….
……….
……..
}
```

Like this we are having content of a row and we will have multiple JSON and that will be our Log file

- Implemented on storage engine level
- here JSON is a neutral format and data across storage engine could be different
- Doesn't have any random value
- Doesn't depend on storage location
- Complexity and Heavy

4. Triggered-based Replication:
Similar to click-events in web applications, we have Triggers in database

```
Create Trigger Replicate-data
After Insert On orders
For each row
Begin
        - Insert into Replication_Log (id, item)
          Values (New.id, New.item)
```

We create a Trigger, so whenever an Update/Insert/Delete happens in Database, we will update Logs
Humans can create Logs according to their system. It is risky because Human intervention is there
Software using it: Oracle GoldenGate, Postgres