Replication

Database Replication হলো একটি ডাটাবেসের হুবহু কপি বা প্রতিলিপি অন্য এক বা একাধিক সার্ভারে তৈরি করে রাখা। এটি অনেকটা আপনার গুরুত্বপূর্ণ ফাইলগুলো পেনড্রাইভে ব্যাকআপ রাখার মতো, তবে এখানে ব্যাকআপটি রিয়েল-টাইমে (সাথে সাথে) হয়।

নিচে আপনার উল্লেখ করা বিষয়গুলো বিস্তারিত আলোচনা করা হলো:

১. Replication **কী এবং কেন**?

রেপ্লিকেশন মূলত সিস্টেমের নিরাপত্তা এবং গতি বাড়ানোর জন্য ব্যবহার করা হয়।

- Reliability: যদি একটি সার্ভার নষ্ট হয়ে যায়, অন্যটি থেকে ডাটা পাওয়া যায়।
- Performance: ডাটাবেসের ওপর চাপ কমানোর জন্য কাজগুলোকে ভাগ করে দেওয়া হয়।

২. Master-Slave Replication (**মাস্টার-স্লেভ**)

এটি ডাটাবেস রেপ্লিকেশনের সবচেয়ে জনপ্রিয় আর্কিটেকচার। বর্তমানে একে অনেক ক্ষেত্রে Primary-Secondary বা Leader-Follower ও বলা হয়।

- Master (**প্রাইমারি সার্ভার**): এটি হলো প্রধান ডাটাবেস। এখানে সব ধরনের Write (**লিখা**) অপারেশন হয়। যেমন: নতুন ইউজার রেজিস্ট্রেশন করা, ডাটা আপডেট বা ডিলিট করা।
- Slave (**সেকেন্ডারি সার্ভার**): এটি মাস্টারের একটি কপি। মাস্টারে কোনো পরিবর্তন হলে সেটি স্বয়ংক্রিয়ভাবে স্লেভ সার্ভারে কপি হয়ে যায়। স্লেভ সাধারণত শুধু Read (**পড়া**) করার জন্য ব্যবহৃত হয়।

৩. Read Replicas (**রিড রেপ্লিকা**)

আপনার অ্যাপ্লিকেশনে যদি ইউজারের সংখ্যা অনেক বেড়ে যায়, তবে সবাই যখন একসাথে ডাটা দেখতে চায় (যেমন: ফেসবুক নিউজফিড স্ক্রল করা), তখন মেইন ডাটাবেসের ওপর প্রচুর চাপ পড়ে। এই চাপ কমানোর সমাধান হলো Read Replicas।

- **কাজ**: এটি মূলত স্লেভ সার্ভারগুলোর একটি বিশেষ রূপ যা শুধুমাত্র ডাটা পড়ার (Read queries) জন্য ডেডিকেটেড থাকে।
- **সুবিধা**: আপনার অ্যাপ যখন ডাটা সেভ করবে, তখন সে Master-কে ব্যবহার করবে। আর যখন ডাটা দেখাবে (যেমন প্রোফাইল দেখা), তখন সে Read Replica থেকে ডাটা নিয়ে আসবে। এতে মেইন সার্ভার ফ্রি থাকে।

৪. এটি কীভাবে কাজ করে? (Workflow)

১. ইউজার একটি পোস্ট শেয়ার করল (Write Action)।

২. ডাটাটি Master DB-তে সেভ হলো।

৩. মাস্টার ডাটাবেস সাথে সাথে ওই ডাটাটি তার Read Replicas বা স্লেভদের পাঠিয়ে দিল।

৪. অন্য কোনো ইউজার যখন ওই পোস্টটি দেখল (Read Action), তখন সে মেইন মাস্টার সার্ভারে না গিয়ে কাছের কোনো Read Replica থেকে তথ্যটি পেল।

৫. Replication-এর সুবিধা ও চ্যালেঞ্জ

সুবিধা	চ্যালেঞ্জ
High Availability: মাস্টার সার্ভার ডাউন হলেও রিড রেপ্লিকা থেকে সার্ভিস চালু রাখা যায়।	Consistency Issue: মাস্টারে ডাটা আপডেট হওয়ার কয়েক মিলিসেকেন্ড পর রেপ্লিকাতে আপডেট হয়। একে 'Replication Lag' বলে।

সুবিধা	চ্যালেঞ্জ
Better Performance: রাইট এবং রিড অপারেশন আলাদা হওয়ায় সিস্টেম সুপার ফাস্ট হয়।	Complexity: অনেকগুলো সার্ভার ম্যানেজ করা এবং সেগুলোর মধ্যে সিঙ্ক ঠিক রাখা বেশ জটিল।

সংক্ষেপে সারাংশ:

- Replication: ডাটা কপি করে রাখা।
- Master: যেখানে ডাটা এন্ট্রি হয় (Write)।
- Slave/Read Replica: যেখান থেকে ডাটা শুধু দেখা যায় (Read)।

🧠 **Replication কী?**

👉 **Same data multiple servers-এ copy করে রাখা**

Purpose:

- High availability
- Read scalability
- Fault tolerance

📌 Replication ≠ Sharding
👉 Replication = same data
👉 Sharding = different data

🧱 Replication Architecture (Base)

```
      Writes
Client ──────────▶ Master
          │
          ▼
        Replication
          │
      Read Replicas
      (Slave nodes)
```

1. Master–Slave Replication

কী?

👉 **One master handles all writes**
👉 **Slaves handle reads**

🔁 **Write Flow**

Client → Master → Ack → Replicas sync

🔁 **Read Flow**

Client → Read Replica

✅ Pros

- Simple architecture
- Strong consistency on writes
- Easy to reason

❌ Cons (VERY IMPORTANT)

- Master = bottleneck
- Single point of failure (without failover)
- Replication lag

📌 Interview line:

"Master-slave replication improves read scalability but introduces replication lag."

2. Read Replicas (Deep Dive)

👉 Read replicas = slave nodes optimized for reads

Why Read Replicas?

- Read-heavy workload
- Offload master
- Better latency (geo replicas)

🔁 Read-after-write Problem

Scenario:

User updates profile

Immediately reads profile

❌ Replica may return old data

Solutions:

- Read from master for critical reads
- Session stickiness
- Read-your-write consistency

📌 Interview line:

“Critical reads are routed to the master to avoid stale data.”

3. Replication Types (Advanced)

🔹 **Synchronous**

- Master waits for replicas
- Strong consistency
- Higher latency

🔹 **Asynchronous (Most common)**

- Master doesn’t wait
- Fast writes
- Eventual consistency

4. Failure Handling (Interview MUST)

Master Failure

- Promote replica to master
- Update routing
- Resume writes

📌 Tooling:

- ZooKeeper
- Raft
- Cloud managed DBs

5. Replication vs Sharding (Quick Compare)

Aspect	Replication	Sharding
Data	Same	Split
Goal	Availability	Scalability
Read scale	Yes	Yes
Write scale	No	Yes

6. Real-World Examples

Instagram

- Master DB → writes
- Read replicas → feed reads
- Redis → hot cache

Banking System

- Writes → master only
- Reads → master for critical data

🏆 FAANG Interview Answer Template 💎

Say this 👇

"Replication improves availability and read scalability by copying data across nodes. In a master-slave setup, writes go to the master while read replicas serve read traffic, with careful handling of replication lag and failover."

🧠 One-Line Memory Hook

Replication = same data, many servers