CAP Theorem

1.CAP Theorem

Why CAP Theorem is Important?

CAP Theorem is a **basic and very important concept** in **Distributed Databases**.

If you want to:

- Design a **large-scale system**
- Handle **millions of users**
- Run services across **multiple servers and locations**

You **must understand CAP Theorem** to make correct design decisions based on business needs.

2.First, What is a Distributed System?

A **distributed system** means:

- Data is stored on **multiple servers (nodes)**
- Servers may be in **different locations**
- All servers work together as **one system**

Example:

- Facebook
- Amazon
- Banking systems
- Cloud databases

CAP = Consistency + Availability + Partition Tolerance

CAP has **three properties**.
Let's understand each **one by one**.

1.Consistency (C)

Meaning:
All nodes always return the **same and latest data**.

If data is updated on one node:

- It is **immediately replicated** to all other nodes
- Every user sees the **same data at the same time**

Real-World Example (Bank)

- You have ₹10,000 in your account
- You withdraw ₹5,000 from ATM
- Immediately, **every branch** shows balance = ₹5,000

No matter where you check, data is **always correct and updated**

That is **Consistency**.

Key Point:

1.Read always returns the **latest write**
2.But consistency may slow down the system

2.Availability (A)

Meaning:
Every request **always gets a response**, even if some nodes are down.

The system:

- Never stops working
- Responds to every request

But the response **may not be the latest data**

Real-World Example (Social Media)

- You post a photo
- Some users see it immediately
- Some users see it after a few seconds

System is **always available**, but data may not be updated everywhere.

That is **Availability**.

Key Point:

System never goes down
Data may be outdated

3.Partition Tolerance (P)

Meaning:
System continues to work **even if communication breaks** between nodes.

Partition happens when:

- Network fails
- Messages are delayed or dropped
- Nodes cannot talk to each other

Real-World Example (Internet Cable Cut)

- Data center in India
- Data center in USA
- Network cable breaks

System still runs independently on both sides

That is **Partition Tolerance**.

Key Point:

System survives network failures
Very important in real distributed systems

3.What CAP Theorem Says

CAP Theorem Rule:

A distributed system can provide **only TWO** out of:

- Consistency (C)
- Availability (A)
- Partition Tolerance (P)

You **cannot have all three at the same time**.

Why?

Because:

- Network failures **will happen**
- Partition tolerance is **mandatory**
- So you must choose between:
  - Consistency
  - Availability

CAP Theorem and NoSQL Databases

NoSQL databases are designed for:

- Large data
- Horizontal scaling
- Distributed environments

So CAP theorem plays a **big role** in choosing the database.

4.Types of Databases Based on CAP

1.CA Databases (Consistency + Availability)

Consistent
Available
Not partition tolerant

Meaning:

- Data is always correct
- System is always available
- But if network fails → system fails

Example:

- MySQL
- PostgreSQL (with replication)

Used when:

- Single data center
- Network is reliable

Not practical for large distributed systems because **partitions are unavoidable**.

2.CP Databases (Consistency + Partition Tolerance)

Data is always correct
Handles network failures
System may become unavailable

What happens during partition?

- Inconsistent nodes are **shut down**
- System prefers **correct data over availability**

Real-World Example (Banking)

In banking:

- Wrong balance is **not acceptable**
- Temporary unavailability is acceptable

Consistency > Availability

Example:

- **MongoDB**

MongoDB uses:

- One **Primary Node** (writes happen here)
- Multiple **Secondary Nodes** (replicas)

If primary fails:

- One secondary becomes new primary

👉 Ensures **strong consistency**

3.AP Databases (Availability + Partition Tolerance)

Always available
Works during network failure
Data may be inconsistent temporarily

What happens during partition?

- All nodes stay active
- Some nodes may have **old data**
- Data is synced later (eventual consistency)

Real-World Example (Facebook / Instagram)

- Likes count may differ
- Comments appear late
- App never goes down

Availability > Consistency

Example:

- Apache Cassandra
- Amazon DynamoDB

Used in:

- Social media
- Messaging apps
- Recommendation systems

Final Comparison Table

Type	Guarantees	Sacrifices	Use Case
CA	C + A	P	Small systems
CP	C + P	A	Banking, finance
AP	A + P	C	Social media