**What is CAP Theorem, explain?**

The CAP theorem maintains that a distributed system can deliver only two of three desired characteristics: consistency, availability, and partition tolerance.

A distributed system is a network that stores data on more than one node (physical or virtual machines) at the same time. Because all cloud applications are distributed systems, it’s essential to understand the CAP theorem when designing a cloud app so that you can choose a data management system that delivers the characteristics your application needs most.

**Consistency**

Consistency means that all clients see the same data at the same time, no matter which node they connect to. For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed ‘successful.’

**Availability**

Availability means that any client making a request for data gets a response, even if one or more nodes are down. Another way to state this—all working nodes in the distributed system return a valid response for any request, without exception.

**Partition tolerance**

A partition is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes. Partition tolerance means that the cluster must continue to work despite any number of communication breakdowns between nodes in the system.

NoSQL databases are ideal for distributed network applications. Unlike SQL, NoSQL databases are horizontally scalable and distributed by design—they can rapidly scale across a growing network consisting of multiple interconnected nodes.

NoSQL databases are classified based on the two CAP characteristics they support:

* **CP database:**A CP database delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system must shut down the non-consistent node (i.e., make it unavailable) until the partition is resolved.
* **AP database:**An AP database delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others. (When the partition is resolved, the AP databases typically resync the nodes to repair all inconsistencies in the system.)
* **CA database:**A CA database delivers consistency and availability across all nodes. It can’t do this if there is a partition between any two nodes in the system, however, and therefore can’t deliver fault tolerance.

So, while we can discuss a CA distributed database in theory, **for all practical purposes a CA distributed database can’t exist.** This doesn’t mean you can’t have a CA database for your distributed application if you need one. **Many relational databases, such as PostgreSQL, deliver consistency and availability and can be deployed to multiple nodes using replication.**

**Reference-** [**https://www.ibm.com/topics/cap-theorem**](https://www.ibm.com/topics/cap-theorem)

**What are SOLID principles in Java, explain?**

**SOLID** principles are object-oriented design concepts relevant to software development.

**SOLID is an acronym for five other class-design principles:**

**Single Responsibility Principle**- Each class should be responsible for a single part or functionality of the system.

**Open-Closed Principle**- Software components should be open for extension, but not for modification.

**Liskov Substitution Principle**- Objects of a superclass should be replaceable with objects of its subclasses without breaking the system.

**Interface Segregation Principle**- No client should be forced to depend on methods that it does not use.

**Dependency Inversion Principle**- High-level modules should not depend on low-level modules, both should depend on abstractions.

SOLID is a structured design approach that ensures your software is modular and easy to maintain, understand, debug, and refactor. Following SOLID also helps save time and effort in both development and maintenance. SOLID prevents your code from becoming rigid and fragile, which helps you build long-lasting software.

Reference - <https://www.educative.io/answers/what-are-the-solid-principles-in-java>