Tutorial 8a
Non-Relational Databases: Transaction

SOLUTIONS

Discuss the following questions:

1. What are the main objectives of Databases and what are the things the database management systems must do in order to achieve those objectives?

   Databases are designed to do two things: to store data and to retrieve data.
   In order to achieve these objectives, the database management systems must do three things:
   - Store data persistently
   - Maintain data consistency
   - Ensure data availability

2. Describe a two-phase commit. Does it help ensure consistency or availability?

   A two-phase commit is a transaction that requires writing data to two separate locations. In the first phase of the operation, the database writes, or commits, the data to the disk of the primary server. In the second phase of the operation, the database writes data to the disk of the backup server.
   Two-phase commits help ensure consistency.

3. What is CAP Theorem?

   CAP Theorem states that it is impossible for a distributed system to provide following three guarantees at the same time:
   - **Consistency**: All nodes see the same data, and the same versions of these data, at the same time.
   - **Availability**: Every request receives a response indicating a success or failure result.
   - **Partition Tolerance**: The system continues to work even if nodes go down or are added. The distributed system can cope with it being divided into two or more disjoint network partitions due to node or network failure.

4. What do the C and A in the CAP theorem stand for? Give an example of how designing for one of those properties can lead to difficulties in maintaining the other.

   C stands for consistency; A stands for availability.
   When using a two-phase commit, the database favours consistency but at the risk of the most recent data not being available for a brief period of time. While the two-phase commit is executing, other queries to the data are blocked. The updated data is unavailable until the two-phase commit finishes. This favours consistency over availability.

5. What does the BASE stand for? Explain each principle of BASE.

BASE stands for **B**asically **A**vailable, **S**oft state and **E**ventual consistency.

- **Basically Available**: Measures are in place to guarantee availability under all circumstances, if necessary, at the cost of consistency.
- **Soft state**: The state of the database may evolve, even without external input, due to the asynchronous propagation of updates throughout the system.
- **Eventual Consistency**: The database will become consistent over time, but may not be consistent at any moment and especially not at transaction commit.

6. How does a BASE system differ from a traditional distributed database system?

A traditional database system enforces the ACID properties as to ensure that all database transactions yield a database in a consistent state. In a centralized database system, all data resides in a centralized node. However, in a distributed database system data are located in multiple geographically disperse sites connected via a network. In such cases, network latency and network partitioning impose a new level of complexity. In most highly distributed systems, designers tend to emphasize availability over data consistency and partition tolerance. This trade-off has given way to a new type of database systems in which data are basically available, soft state and eventually consistent (BASE).

7. The E in BASE stands for eventually consistent. What does that mean?

E stands for eventually consistent, which means that some replicas might be inconsistent for some period of time but will become consistent at some point.

8. What are the types of eventual consistency?

- Casual consistency
- Read-your-writes consistency
- Session consistency
- Monotonic read consistency
- Monotonic write consistency

9. Describe monotonic write consistency. Why is it so important?

Monotonic write consistency ensures that if you were to issue several update commands, they would be executed in the order you issued them. This ensures that the results of a set of commands are predictable. Repeating the same commands with the same starting data will yield the same results.

The End