Assignment 5

|  |  |
| --- | --- |
| SQL | No-SQL |
| Relational database | Non-relational database |
| Pre-defined schema | Dynamic schema |
| Vertical scaling | Horizontal scaling |
| ACID | CAP |
| Not suited for hieratical data store | Suited for hieratical data store |

1. **Relational database:** A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database.
2. **Non-relational database:** non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored.
3. Pre-defined schema:
4. Dynamic schema: a dynamic schema changes as you add data. There is no need to define the schema beforehand. When data is inserted, updated, or removed, the database builds a schema dynamically.
5. Vertical scaling: vertical scaling refers to increasing the processing power of a single server or cluster. Both relational and non-relational databases can scale up, but eventually, there will be a limit in terms of maximum processing power and throughput. Additionally, there are increased costs with high-performance hardware, as costs do not scale linearly.

Shape, arrow

Description automatically generated

1. Horizontal scaling: Horizontal scaling, also known as scale-out, refers to bringing on additional nodes to share the load. This is difficult with relational databases due to the difficulty in spreading out related data across nodes. With non-relational databases, this is made simpler since collections are self-contained and not coupled relationally. This allows them to be distributed across nodes more simply, as queries do not have to “join” them together across nodes.

Icon

Description automatically generated with medium confidence

1. ACID: The SQL ACID is an acronym for Atomicity, Consistency, Isolation, Durability.
   1. Atomicity: the atomicity acid property in SQL. It means either all the operations (insert, update, delete) inside a transaction take place or none. Or you can say, all the statements (insert, update, delete) inside a transaction are either completed or rolled back.
   2. Consistency: This SQL ACID property ensures database consistency. It means, whatever happens in the middle of the transaction, this acid property will never leave your database in a half-completed state.
      1. If the transaction completed successfully, then it will apply all the changes to the database.
      2. If there is an error in a transaction, then all the changes that already made will be rolled back automatically. It means the database will restore to its state that it had before the transaction started.
      3. If there is a system failure in the middle of the transaction, then also, all the changes made already will automatically rollback.
   3. Isolation: Every transaction is individual, and One transaction can’t access the result of other transactions until the transaction completed. Or, you can’t perform the same operation using multiple transactions at the same time. We will explain this SQL acid property in a separate article.
   4. Durability: Once the transaction completed, then the changes it has made to the database will be permanent. Even if there is a system failure, or any abnormal changes also, this SQL acid property will safeguard the committed data.
2. CAP: consistency, availability, and partition tolerance.
   1. Consistency: all clients always have the same view of the data
   2. Availability: each client can always read and write
   3. Partition tolerance: the system works despite physical network partitions
   4. AP: Dynamo, Cassandra, SimpleDB, CouchDB…
   5. CP: BigTable, MongoDB, Hbase, Redis…
3. Hieratical data store: