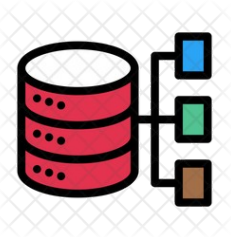
Practical Design of NoSQL Database

Tushar B. Kute,
http://tusharkute.com



NoSQL Design Principles

- NoSQL databases require a different design approach than that used with traditional relational database management systems (RDBMS).
- The rise in popularity of NoSQL databases paralleled the adoption of agile and DevOps practices. Unlike RDBMSs, NoSQL databases encourage ‘application-first’ or API-first development patterns.
- Before considering the data models and entities, developers first consider queries that support the functionality specific to an application.
- This developer-friendly architecture paved the path to the success of the first generation of NoSQL databases.

NoSQL Design Principles

- Relational databases impose fairly rigid, schema-based structures to data models; tables consisting of columns and rows.
- Each table typically defines an entity. Each row in a table holds one entry, and each column contains a specific piece of information for that record.
- The relationships among tables are clearly defined and usually enforced by schemas and database rules.

NoSQL Design Principles

- In the world of relational databases, schemas are usually managed by a centralized team of database administrators, who ensure that data models and data types are consistent across multiple applications.
- This situation can often introduce friction between administrators and development teams. This friction often translates into very long, non-agile application development lifecycles.
- Such highly structured data requires normalization to reduce redundancy and improve reliability.
- The data model is based on the entity being represented; query patterns are a secondary consideration.

NoSQL Design Principles

- Non-relational data models are flexible, and schema management is often delegated to application developers, who are relatively free to adapt data models independently.
- Such a decentralized approach accelerates development cycles and provides a more agile approach to addressing user requirements.
- Application developers can easily add properties and attributes, and the changes can be applied to existing data sets.
- This enables developers to add new features more quickly than if a schema migration were required.

Oracle NoSQL Database

- Oracle NoSQL Database enables the creation of innovative applications that engage customers and create business value for constantly changing business requirements.
- Your organization can quickly respond to new opportunities that require very fast storage and retrieval of data with extremely low latency.
- Oracle NoSQL Database is a scalable, distributed NoSQL database, designed to provide highly reliable, flexible and available data management across a configurable set of storage nodes.

Oracle NoSQL Database

- Oracle NoSQL Database has been designed to be flexible in a number of areas:
  - Developers can create innovative applications using a number of popular programming languages and are able to model the data in a number of ways.
  - Administrators can plan for varying workloads by scaling both vertically and horizontally, using industry standard servers.

Oracle NoSQL Database: Modelling

- As a true multi-model data store, the Oracle NoSQL Database provides several different options for data modeling:
  - Key/value pairs – Using this modeling option, developers specify string keys and opaque byte arrays as values. The Oracle NoSQL Database does not interpret the byte arrays. These are serialized and de-serialized by the application.
  - Tables – Using this modeling option, developers specify table definitions using a SQL data definition language, very similar to the relational database. Command line SQL for querying the Oracle NoSQL Database or the Oracle NoSQL Database APIs can be used for table data access.

Oracle NoSQL Database: Modelling

- Graph – Using this modeling option, developers define any number of name/value pairs for the edges or vertices of the graph. Then, edges are created that draw “labeled” relationships between the graph vertices.
- JSON documents – Using this modeling option, developers can store JSON objects in the Oracle NoSQL Database and use SQL to access these objects.
  - The Oracle NoSQL Database provides a rich SQL dialect with specialized operators designed specifically for slicing through JSON objects.
  - Secondary indexes specified by JSON path expressions are also supported via this SQL dialect.

ACID vs. BASE

- Deciding on the right database management system (DBMS) can be a difficult task. The number of available options is huge.
- Before you start searching for an adequate database solution, consider your requirements.
- A recommended first step in this process is to choose a database transaction model.
- This is a set of rules which determine how a database organizes, stores, and manipulates data.

ACID Model

- The ACID database transaction model ensures that a performed transaction is always consistent.
- This makes it a good fit for businesses which deal with online transaction processing (e.g., finance institutions) or online analytical processing (e.g., data warehousing).
- These organizations need database systems which can handle many small simultaneous transactions. There must be zero tolerance for invalid states.

ACID Model

ACID Model

- Atomic – Each transaction is either properly carried out or the process halts and the database reverts back to the state before the transaction started. This ensures that all data in the database is valid.
- Consistent – A processed transaction will never endanger the structural integrity of the database.
- Isolated – Transactions cannot compromise the integrity of other transactions by interacting with them while they are still in progress.
- Durable – The data related to the completed transaction will persist even in the cases of network or power outages. If a transaction fails, it will not impact the manipulated data.

ACID Model – Use Cases

- Financial institutions will almost exclusively use ACID databases. Money transfers depend on the atomic nature of ACID.
- An interrupted transaction which is not immediately removed from the database can cause a lot of issues.
- Money could be debited from one account and, due to an error, never credited to another.

ACID Model – Which Databases?

- One safe way to make sure your database is ACID compliant is to choose a relational database management system. These include MySQL, PostgreSQL, Oracle, SQLite, and Microsoft SQL Server.
- Some NoSQL DBMSs, such as Apache's CouchDB or IBM's Db2, also possess a certain degree of ACID compliance.
- However, the philosophy behind the NoSQL approach to database management goes against the strict ACID rules.
- Hence, NoSQL databases are not the recommended choice for those who need strict environments.

BASE Model

- The rise of NoSQL databases provided a flexible and fluid way to manipulate data.
- As a result, a new database model was designed, reflecting these properties.
- The acronym BASE is slightly more confusing than ACID.
- However, the words behind it suggest ways in which the BASE model is different.

BASE Model


BASICALLY AVAILABLE
SOFT STATE
BASE
EVENTUALLY CONSISTENT

BASE Model

- Basically Available – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.
- Soft State – Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers.
- Eventually Consistent – The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

BASE Model – Use Cases

- Marketing and customer service companies who deal with sentiment analysis will prefer the elasticity of BASE when conducting their social network research.
- Social network feeds are not well structured but contain huge amounts of data which a BASE-modeled database can easily store.

BASE Model – Which Databases?

- Just as SQL databases are almost uniformly ACID compliant, NoSQL databases tend to conform to BASE principles.
- MongoDB, Cassandra and Redis are among the most popular NoSQL solutions, together with Amazon DynamoDB and Couchbase.

Which is best?

- It is not possible to give a straight answer to the question of which database model is better. Therefore, a decision must be reached by considering all the aspects of the project.
- Given their highly structured nature, ACID-compliant databases will be a better fit for those who require consistency, predictability, and reliability.
- Those who consider growth to be among their priorities will likely want to choose the BASE model, because it enables easier scaling up and provides more flexibility.
- However, BASE also requires developers who will know how to deal with the limitations of the model.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com