MongoDB Design Patterns

MongoDB design patterns help optimize data models based on application queries and usage. However, MongoDB design patterns improve application performance by reducing schema complexity. It helps identify data storage patterns and the type of data returned to the application.

However, MongoDB design patterns are best practices for structuring data and queries to optimize performance, scalability, and maintainability. These patterns arise from common use cases and challenges encountered. Some prominent MongoDB design patterns include approximation, attribute, polymorphic, outliers, and bucket. Let's read about each of these MongoDB design patterns.

Approximation

Approximation MongoDB design pattern helps calculate frequent expenses, where the precision for those calculations is not a priority. For example, Instagram followers are for celebrities.

The approximation design pattern is useful for fewer writes to the database and maintains statistically valid numbers. However, it cannot represent exact numbers.

Attribute

The Attribute MongoDB design pattern helps document various characteristics and similar fields that you cannot identify when raising a query. This attribute is useful when the fields need to be sorted and found in a small subset of documents or when both conditions meet within the documents, such as when selling products on e-commerce.

The attribute MongoDB design pattern is useful in less indexing and generating simple and faster queries.

Polymorphic

The polymorphic MongoDB design pattern helps store different documents in the same collection. However, it often uses a discriminator field to differentiate between types of documents, such as customer communication stored using various channels.

This type of design pattern is easy to implement and runs queries across a single collection.

Outlier

The Outlier MongoDB design pattern is useful for queries or documents that don't fit into the typical data pattern, such as a list of a celebrity's social media followers (which could be in millions). This design pattern prevents documents or queries from determining an application's solution; however, it is tailored for typical use cases but doesn't address ad hoc queries.

Bucket Pattern

The bucket pattern in MongoDB design helps to manage streaming data such as time series, real-time analytics, or Internet of Things (IoT) applications. The bucket MongoDB design pattern reduces the number of documentation while collecting them, improves index performance, and simplifies data access by leveraging pre-aggregation.

For example: Collecting weather data using multiple sensors. This data collection helps to reduce efforts to review temperature and wind speed every minute. However, you would receive a summary per hour, such as:

- Average temperature
- Average wind speed

However, you can review the weather forecast in detail. For example, find the temperature per minute and calculate the median and variance. The benefit of the bucket design pattern is that it doesn't repeat device identifiers and entry data with every record. However, if this didn't happen, there would be 1440 records per device per hour, creating a large amount of data.

Summary

In this reading, you've learned that MongoDB design patterns optimize the data model to access the application's patterns. Such as:

- Approximation
- Attribute
- Polymorphic
- Outliers
- Bucket

Skills Network