|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Model** | **Scalability** | **Flexibility** | **Performance** | **Complexity** | **Structure** |
| **Key-Value Store** | High | High | High | None | Primary key with some value |
| **Document-Oriented Store** | Variable | High | High | Low | Tree structured JSON Object Form |
| **Column-Based Store** | High | Moderate | High | Low | Rows and Columns |
| **Graph-Based Store** | Variable | High | Variable | High | Graph entities and relations |

|  |  |  |
| --- | --- | --- |
| **Basis** | **RDBMS** | **NoSQL** |
| **Software Distribution** | Both Open-source and Closed-source | Mostly Open source |
| **Cost** | Expensive | Cheaper |
| **Volume of Data** | Can Limited Data | Handles large volumes of data |
| **Scalability** | By upgrading hardware of a single server | Horizontal scaling using commodity servers |
| **Complexity** | Complex data is difficult to convert into tables | Relatively less difficult to store complex data |
| **Consistency** | Strong since uses strict schemas design | Poor since uses schema less design |
| **Performance** | Slow to process information | Better performance to process information |
| **Data Manipulation** | Most RDBMS only use SQL as the query language | NoSQL databases have its own manipulation language |
| **Security** | Strong mechanisms to protect the data | No built-in security protocols but can be handled by middleware programs. |

Abstract

2 Introduction

Data is produced and prepared more quickly now than any other time before. Lately as more than 2.5 trillion of Data is created consistently. Data will keep on expanding in volumes later at an outstanding level. This developing interest brought about various NoSQL Database Management Systems (DBMS’s), with an emphasis on execution, depend-ability, and consistency. Various existing ordering structures were reused and refined to upgrade search and read execution.

The SQL scalability issue was perceived by organizations with tremendous, developing information and foundation needs, for example, Google, Amazon, and Facebook. They concocted their own answers for the issue – advancements like Big Data, DynamoDB, and Cassandra. NoSQL stands for “Not only SQL”. The data in the NoSQL is not stored as tables that is in Relational Databases. NoSQL has a SQL like syntax. NoSQL Databases offers straight forward plan, better control on accessing data by sup-porting horizontal scaling for many machines.

Sometimes, database structure of NoSQL is flexible than Relational Database models. The utilization of NoSQL databases is expanded essentially because of its use which incorporates factors like low-level query language, high normalized interfaces and less ventures when contrasted with relational databases, it additionally offers consistency regarding speed, segment resistance and accessibility.

Advantages of working with NoSQL databases are Scalability and Availability.

Sharding is a technique where data is partitioned and distributed to several servers. Vertical scaling is where resource capacity of the existing server is increased. Most of the NoSQL databases are horizontally scalable using sharding technique.

Availability NoSQL databases are profoundly accessible as highlight to auto replication which recreates to past steady state if there should be an occurrence of data failure.

3 Background

MongoDB is a horizontally scalable, document-oriented NoSQL Database. The software is written in C++ and uses JSON-like documents. MongoDB architecture consists of collections and documents. Tables are known as collections and records are known as documents. Document is a data structure where data is composed of field and value pairs.

MongoDB can store any sort of information and hence-forth does not need any predefined patterns. This gives the client greater adaptability while making new fields in document. Document-oriented databases are substantially more adaptable. The structure of every document does not need to be consistent. Indeed, even enormous volumes of unstructured information can be obliged in the database.

4 Analysis of RDBMS and NoSQL

4.1 Design principles and theorems

NoSQL

NoSQL gives an API and utilizes decentralized engineering for the replication of data.

BASE

NoSQL databases follow BASE (Basically Available, Soft State, Eventual consistency) principles.

Basically Available: This states that the system ensures the accessibility of the data.

Soft State: Data are in changing state over the long haul without a user’s potential contribution.

Eventual consistency: The system becomes eventually consistent as it is state will be modified when receiving input is stopped. Eventually the information is updated maintaining consistent database.

CAP

CAP stands for Consistency, Availability, and Partition tolerance.

Consistency: Data existing on all machines must be indistinguishable after all update operations.

Availability: Data ought to be made open forever rather than transitory access.

Partition tolerance: Partition Tolerance is an assurance that the framework keeps on working notwithstanding subjective message loss or failure of part of the framework. Regardless of whether there is a network outage in the server and a portion of the PCs are in accessible, still, the system keeps on performing.

Lacking Schema:

NoSQL databases require no defined schemas before data entry. Without affecting the applications, schema can be constructed at any time. We can also change the schema at any time.

Features of MongoDB

Schema-less Database: Different types of documents can be held by the collection. Documents can be of varying number of fields and size. This provides great flexibility to database.

Document-Oriented: Data is stored in documents in key-value pairs. Each document is assigned a unique object id.

Indexing: Each field in the document is ordered with indices. This makes it simpler and sets aside less effort to get or look through information from the huge amount of data.

Scalability: Using sharding technique, data is distributed to multiple servers using shard key. New servers can be added to the running database.

Replication: MongoDB duplicates data and send them to multiple servers increasing data availability.

Aggregation: MongoDB allows aggregation operations which takes grouped data and produces single output. It provides three different aggregations i.e., aggregation pipeline, map-reduce function, and single-purpose aggregation methods

High Performance: MongoDB has high performance and persistence of data because of above features.

4.2 Data models

NoSQL

There are 4 types of data models in NoSQL.

Key-value store: Uses key value pairs to store the data in the database. Keys must be unique, and values can be of any data type including URLs. Key length varies from database to database. The size of the key is significant here, as a long key can cause execution issues though too short a key can cause coherence issues. Key-value database models are useful for Ecommerce. Examples are Oracle NoSQL database and Redis.

Document-oriented store: It is a semi-structured data where records and its metadata are stored in the same document. Data encoding is done like XML or JSON. This type of databases is used in big data analysis. Examples are MongoDB and Couchbase.

Column-based store: Data is stored on columns instead of rows. All operations are done on columns. Storing data in columns results in fast access and retrieval of data. They are highly scalable and are used in Data mining. Examples are HBase, Big Table, Cassandra.

Graph-based store: Graphical representation is used to store data. Object is stored as node and properties associated. Connection to other objects/nodes are represented by relationship. Data can be efficiently transformed from one model to another. Rule of graph-based database is that node cannot be deleted unless all its relationships are removed. These databases are used in analyzing social media networks. Examples are Info Grid and Infinite Graph.

4.3 Indexing techniques

Indexing is associating a key to its data record. Below are the commonly used indexing methods.

B-Tree Indexing: B-Tree is one of the self-balancing trees which implement dynamic multi-level indexing. B-Tree reduces the frequency of accessing disk. A range is set for each node there by setting the height low. For the most part, the B-Tree node size is held equivalent to the disk block size.

Most popular B- Tree is B+-Tree where all keys are stored in leave nodes. Leaf nodes are connected to give requested access to the records. Some of the values of the key values show up in the internal nodes, to just go about as a medium to control the looking of a record. B+-Tree of n elements and t degree must satisfy below properties\*.

Each node contains at least (t-1)/2 keys.

All nodes contain t -1 keys.

Each node has number of keys in that node plus 1 child.

All leaves are at same level.

Degree of the tree depends on the block size in the disk.

T-Tree Indexing: The structure of T-Tree is the combination of both AVL-Trees and B-Trees. AVL-Trees are self-balancing binary trees while B-Trees are unbalanced trees with varying child node for each node.

There are three types of nodes in T-Trees. T-node has a right child and left child, half node has only one child, and a leaf node has no children. Each node has more than one pair of tuples {key-value, pointer}. While there is comparability in query operations with that of AVL-Tree, T-Trees offer better execution over them.

O2-Tree Indexing: It is the combination of Red-Black Tree and Binary Search Tree where leaf node contains the tuples {key-value, pointer}. O2-Tree enhances the current indexing method. O2-Tree of order m (m greater than or equal to 2) must satisfy below properties\*.

Each node is either red or black. Root node is always black.

Red node has both children black.

Each leaf node is black and contains tuple.

Leaf nodes have tuples between m/2 to m.

Leaf nodes are doubly linked both in forward and backward directions.

4.4 Security features

Making the database secure is most important. Below are the security controls that help to deal with the security protocol.

Identification and Authentication requirements: Databases can be accessed by users through various means. Authentication can be done by operating system or database.

Reviewing DB users and passwords: Get a rundown of all User Accounts and role of each user account. Roles of user account must be reviewed to make sure users are having appropriate privileges. There should be a few rules for picking the passwords and clients ought to be confined.

Application system connections: Enterprise applications are directly connected to the database with hardcoded passwords that change rarely. Those passwords must be encrypted and stored in user inaccessible areas.

Logging and Monitoring: Data usage and user activities must be logged and monitored.

Backup and Recovery: Guarantee that a proper recovery and backup technique exists.

Vulnerability Analysis: Known vulnerability assessment must be performed at regular time intervals. Proper action must be taken to mitigate.

MongoDB:

Security was not an essential worry of MongoDB's designers. Thus, there are many "loopholes" in its design.

Data files in MongoDB are unencrypted and there are no techniques that would automatically encrypt sensitive information.

Explicit encryption of sensitive data must be implemented at operating system level.

JavaScript is the internal scripting language in MongoDB that is vulnerable to SQL injection attack.

Auditing actions on database in MongoDB is not provided. MongoDB stored information only about database creation but does not store about inserts, updates or deletes.

4.5 Comparisons

5 Working on Big Data

6 Conclusions

7 References

Manageability:

PostgreSQL is easy to manipulate. Each table can be updated without disturbing other tables or the database. The superuser can also share certain groups of data with others and can also limit their access to others.

Flexibility:

if you need to update your data, you only have to do it once – so no more having to change multiple files one at a time.

And it’s pretty simple to extend your database. If your records are growing, a relational database is easily scalable to grow with your data.

Avoid Errors: there’s no room for mistakes in a relational database because it’s easy to check for mistakes against the data in other parts of the records. And since each piece of information is stored at a single point, you don’t have the problem of old versions of data clouding the picture.