International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 02, Issue – 10, October – 2017, PP – 48-53

Multi-model Databases, New SQL, Time Series Databases

Dr. Deepak Chahal[1], Dr. Latika Kharb[2], Prachi Julka[3]
[1,2]Associate Professor (IT), Jagan Institute of Management Studies, New Delhi. India
[3]MCA 5th Sem Student, Department of IT,Jagan Institute of Management Studies, New Delhi. India

Abstract: This chapter provides the information about multi model databases and its characteristics. There is a brief detail about various No SQL data models such as key value data store, document, graph, column data model, which can be used in multi model databases. The difference between polyglot persistence and multi model databases is described. It provides the information about New SQL and its characteristics. Why New SQL is Emerged and the principle of New SQL architecture. It introduces the time series databases, its types and importance. It also describes the traditional and modern method of storing time series databases.

Keywords: Schema, Model.

1. Introduction

In Today's scenario there is a need of multiple type of databases to work for a single project, databases are not only expected to be flexible but they also need to provide high performance and a scale to handle huge data volumes. Each module needs a different type of database, so in order to fulfil these needs the multi model database, New SQL and non traditional databases are designed. To store the time series data we need to have time series databases.

Multi-model Database is designed to organise different NO SQL data models with single backend. A unifying query language and API is available which covers all the NO SQL models and this also mix them in a single query. It has some characteristic such as unification, parallel scalability, keeps the data consistent, powerful software can be developed.

New SQL is a concept of modern RDBMS[Relational Database Management System] which has all the concept of NO SQL systems for online transaction processing but it also holds the ACID guarantees of a traditional database system. Basically it is an extension of old RDBMS.

Time series databases are used for handling time series data. Time series data is a collection of data items which can be taken through different repeated measured time. The time series database can be called as profiles, curves, or traces.

2. Introduction to Multi-Model Databases

Before the Evolution of multi-model databases concept our data was organised with single data model. Today also in many organisations a single data model is used to organise the Database.

Now-a-days many organisations are moving towards large projects which requires different NO SQL data model to be used for different modules with single backend. So to fulfil this need, multi-model concept is evolved.

No SQL Data models are different from the relational database models. In relational database model the data can only be stored in one format that is in the form of rows and columns (tables). But in No SQL we can store the data in key value store, document data store, graph data store and column oriented database.

SQL has already defined schema but in No SQL the schema is dynamic, it is not predefined.

SQL is not good for the storage of hierarchal data but No SQL is sufficient for storing the hierarchal data.

2.1 MULTI-MODEL DATABASES

Multi-model Database is designed to organise different NO SQL data models with single backend. A unifying query language and API is available which covers all the NO SQL models and this also mix them in a single query. Multi-model Databases basically eliminates the fragmentation and it supplies a consistent backend that supports varieties of products and applications. Multi-model Databases approach can be implemented with polyglot persistence which we will discuss in further section.

Characteristics

- **Unification**

  While using NO SQL, the engineers have many choices of data models. So to decide which model should be used to store and organise data becomes quite difficult, especially when there is a need of different data models in a project. A multi-model database provides different models to be integrated with a single backend.

- **PARALLEL SCALABILITY**

  Performance of database needs to be increased according to the use of an application increases. But sometimes exact need of performance of database may change. In many Database System the user has only one option of vertical scaling. But in multi-model systems different components are allowed to be scaled independently according to needs. Due to increased throughput or storage requirements, the different parts of the backend system can be scaled out horizontally. It's simple to scale down the backend system, when the performance demands decrease.

- **KEEPS THE DATA CONSISTENT**

  If an application receives large amount of data to store and we have to store it in different data models. We will be required that these models reflect the consistent state, without ACID transaction across models. This requirement would be difficult without the use of single backend which stores different data models based on software requirements.

- **POWERFUL SOFTWARES CAN BE DEVELOPED**

  To run different databases in software to make it more powerful is very difficult without the use of multi-model database. Software when supported by multi-model database gets the advantage of scalability, fault tolerance as well as high performance built into the product. Programmer can focus on developing software instead of developing logic to handle the database communications and potential failure conditions. Due to above mention benefits the more powerful software can be developed.

2.2 DATA MODELS

Data models are basically the structure of a database with the conceptual tools which describes data, data relationships, data semantics and constraints. They provide the design to database.
There are many data models that are supported by multi-model databases which include NOSQL data models also. They are as follows.

- **DOCUMENT DATA MODEL**

  According to this model, each record and its related data is considered as a "Document". This model stores data in a XML or JSON formats as document database is a non relational database, but they can implement ACID transactions as well as can acquire some characteristics of RDBMS. It is easy and quick to execute many different optimized queries due to its very powerful query engines and indexing technology.

  XML databases are the part of document databases that extract their Meta data from XML documents. In traditional RDBMS, each and every instance of data has same format so to change that format is very difficult. A document database allows every instance of data to be different from each other and stores all related information together. This reduces database size and makes them more flexible in dealing with the change. This makes the software much better specially web applications on which the changes are constant.
  A unique key in the document database is required to represent the document and can be used to retrieve the document as well. The key can be a string or a path.

  Some examples of document data model are MongoDB, OrientDB, MarkLogic and CouchDB etc.
  The first multimodal database was OrientDB developed on 2010.

Graph Store Data Model

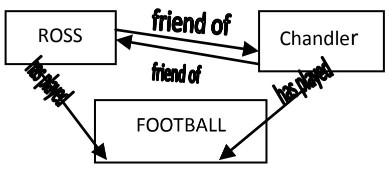
Graph database holds graph structure that represents the queries, properties with the nodes, edges and store data. There is a no root node concept in graph store data model i.e. there is no parent node. If any type of relationship exists between any nodes than that is represented in the form of a graph. Every node has equal importance.

An example to illustrate the graph model.

Let there be two persons, Ross and Chandler plays football and they are friends. Now we have to draw a graph database on the above described situation.
First of all we have to identify the components such as labels, nodes, and relationships.
Nodes represent entities, properties and relationships. Nodes can also be labelled with zero or more labels.
From the information given to us we have the following nodes:
Ross
Chandler
Football
Now, we have to assign the labels to our nodes.
To assign the labels first we need to understand the role of objects. As we know this example is about two different types of objects i.e. two persons, Ross and Chandler are friends. They both play the game, Football.
So the labels will be
Person
Game
As we know the nodes and the labels now we can assign node to their labels. The person label can be given to Ross and Chandler and for graph database we apply the role game.
After this we have to describe the relationship between them. It is as follows:-
Ross is a **friend** of Chandler.
Chandler is a **friend** of Ross**.**
Ross **plays** Football.
Chandler **plays** Football.
Now we can connect the nodes together to describe their connections.
The two nodes labelled **Person** can be connected to each other by the relationship of friend.
The two nodes labelled **Game** and **Person** can be connected to each other by "**has"** played relationship.

Data Model of the Given Statement

Now we can draw our graph model.



- **Key Value Store**

A Key-value store or database is basically designed for storing, retrieving, and managing an associative arrays, dictionary and hash kind of data structures. It considers the data as a single collection which can contain different fields for each record. This follows the concept of **OOPS [Object Oriented Programming].**
Key-value store useless memory to store the database, which can perform large gain performance in certain pressurized work. Document database is a subclass of key value database. Some graph databases are also internally the key data store, which adds the concepts of the relationships between records as a first class data type.

Redis is the most popular key value data store that is used now a day.

- **COLUMN DATA STORE**

A column of column data store is basically an object of No SQL has the lowest level in a keyspace. It is a row which consists of three elements.

- **UNIQUE NAME: -** It refers the column.
- **VALUE: -** The content is placed in this column. It can have different types of data such as Integer type, Long type, ASCII Type among others.
- **TIMESTAMP: -** This is used to ascertain the valid content in a column.

The column is taken as a store to hold the values and it has a timestamp that differentiate the valid contents from ***old*** ones. In distributed databases the consistency, availability and partition tolerance are the major important issues. So to remove this problem programmer will use the timestamp element to check the up to date backup nodes.

2.3 COMPARISON OF POLYGLOT PERSISTENCE AND MULTIMODEL DATABASE

Polyglot persistence is a technology which is used to store data in multiple data storage technologies, which is chosen based upon the way data is used in applications/software. Multi-model Database is designed to organise different NO SQL data models with single backend.

Polyglot Persistence has some disadvantages such as increase in complexity and no support for maintaining data consistency.

Multi model database is developed to remove the disadvantage of the polyglot persistence by keeping all the advantages of polyglot persistence. In polyglot persistence different databases are connected to every other database engine, which will require taking care of the different database engines.

3. Introduction to New Sql

New SQL is a concept of modern RDBMS [Relational Database Management System] which has all the concept of NO SQL systems for online transaction processing but it also holds the ACID guarantees of a traditional database system. Basically it is an extension of old RDBMS.

New SQL contains all the features of RDBMS such as ACID and SQL, but they overcome overheads in the performance of RDBMS. The overheads of RDBMS are caused by Buffer Management, Logging, Locking, Index management and Latching. It is also called as fast in memory SQL. This all overheads are removed with the help of in memory concept.

3.1 IN MEMORY DATABASE

In this concept the data is stored in main memory. Before this concept the database management system used to store data by applying disk storage mechanism.

In memory it is faster to access data from main memory as compared to disk storage mechanism; this performs faster than the disk. Hence the millions of transactions can be performed in a second.

Characteristics of NEW SQL

a. In New SQL we can use the SQL language. SQL language is basically a very simple language which can be taught easily to others. So NEW SQL retains the SQL language because of this reason.

b. There is an ACID support for the transaction. As this property is very useful in providing the security in the transaction process. The ACID property is as follows:-

Atomicity

This property states that "the transaction if occurs than all the operation should occur otherwise nothing occurs".

Consistency

Consistency should be maintained in the database. It states that "The affected changes should be done in the allowed rules. All the rules such as constraints, triggers, and any other combinations should be followed."

Isolation

Isolation states that "The changes after the transaction integrity should be visible to others." If a person is doing the transaction, at the same time other person is also doing the transaction, than at that time each person should think that he/she is the only one who is performing the transaction.

Durability

Durability states that "If a transaction is completed, than all the changes made on the database will be permanent." For an example, If a person 'A' has booked the railway ticket than that ticket is booked and a seat is reserved. This should not be changed.

c. There is a concept of non locking concurrency control mechanism, the real time analysis will read and the conflict of read and write will not occur. This will cause them to stall.

d. NEWSQL database provide much higher performance to each node than available from traditional Relational Database Management System solutions. This improves the performance of the database.

e. There is a scale out architecture which helps in maintaining the performance of the server. A shared-nothing architecture is capable of running on a large number of nodes without suffering bottlenecks.

f. It is based on the concept of in memory database.

g. It is basically single threaded; there is no locking as well as no latching.

h. It can convert live data into business related data.

i. It is much faster than the traditional databases.
j. It analyzes and acts on transferring the data.

4. Introduction to Time Series Databases

Time series databases are used for handling time series data. Time series data is a collection of data items which can be taken through different repeated measured time. The time series database can be called as profiles, curves, or traces.

4.1 IMPORTANCE OF TIME SERIES DATABASES

Time series database is important in various fields. They are as follows.

- Each month weather report of last year can be generated.
- Air pollution level in a city or country
- Need To Companies
- Traditional Methods For Time Series Databases
- Time Series Database Types

Time series databases are basically classified into two categories. They are as follows:-

4.2 OPEN SOURCE TIME SERIES DATABASE

Open source databases are available freely. There is no need to pay the money to use it. Open source time series data uses open source tools like R and Apache spark. Open TSDB is far more flexible than the others because the database engine is based on Apache HBase or Map R-DB, which allows a high flexibility schema. It supports very high performing data recording.

ARCHITECTURE OF OPEN TSDB

Open TSDB has many supporting components which loads and access data from the database tier of time series database. The components include data collectors, time series demons (TSD), and various functions related to user interface.
In a server measurements are made, collector is a process that sends data to the daemons using the protocol i.e. TSD RPC protocol. The daemons are the main components which are responsible for altering and inserting each data point as it is received into the database tier. One can run multiple TSD without worrying of overlapping because the TSD store the data in the database tier immediately without keeping anything in the memory state.
To retrieve the data user interface TSD directly communicates with the TSD. Than the TSD will retrieve the data from the database tier and according to the request it will properly summarize it and returns the result.

Examples of Open Source TSDBs are: - Open TSDB is already explained above.

- **InfluxDB**
  It is a time series database written in Go, created by influxDB. It provide a great query language.
- **KairosDB**
  It can be used as in memory or in Cassandra. It is a fork of Open TSDB.
- **Warp 10**

It is also a time series data base. It has the Go time series platform. It is developed by Cityzen Data and Open Source.

There are many more open source examples are available.

Proprietary time series database
Proprietary software is sold by the company. They are not freely available like open source software. There are many TSD software available. They are as follows:-

- **Axisbase time series database: -** They are designed to store and analyze time series data. It uses Hbase storage. Its architecture is based on No SQL architecture.
- **Infiniflux: -** It is basically used for big data and IoT (Internet of things). There is a multi version concurrency control. The technology which is used in infiniflux is in memory storage concept, Columnar DBMS and a real time search engine technology.
- **Informix: -** It is developed by Informix Company. It is now acquired by IBM Company. It is mostly used in OLTP (Online transaction processing systems) related applications such as retail, finance etc.

There are many more available in proprietary category.

Conclusion

Now a day many databases are available to store different type of data. For an example mangoDB is available for document data model and Graph data model is available to store data in graph. They are more flexible and efficient than the old databases.

References:

[1]. List of NoSQL Databases [currently 225]. (n.d.). Retrieved May 17, 2016, from http://nosql-database.org/

[2]. Main Page. (n.d.). Retrieved May 16, 2016, from http://en.wikipedia.org/wiki/Main_Page

[3]. NoSQL, no security? (n.d.). Retrieved May 19, 2016, from http://www.slideshare.net/wurbanski/nosql-no-security

[4]. Cassandra. (n.d.). Retrieved May 15, 2016, from http://cassandra.apache.org/

[5]. MongoDB for GIANT Ideas. (n.d.). Retrieved May 18, 2016, from http://mongodb.com/

[6]. Retrieved May 19, 2016, from http://www.youtube.com/?app=desktop

[7]. Ron, A., Shulman-Peleg, A., & Puzanov, A. (2016). Analysis and Mitigation of NoSQL Injections. IEEE Security & Privacy IEEE Secur. Privacy, 14(2), 30-39. doi:10.1109/msp.2016.36