Session 09: HBase

Assignment 1

**Problem Statement**

1. What is NoSQL data base?

2. How does data get stored in NoSQl database?

3. What is a column family in HBase?

4. How many maximum number of columns can be added to HBase table?

5. Why columns are not defined at the time of table creation in HBase?

6. How does data get managed in HBase?

7. What happens internally when new data gets inserted into HBase table?

**Solution**

1. **What is NoSQL data base?**

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL.” As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

1. **How does data get stored in NoSQL database?**

TYPES OF NOSQL DATABASES based on data storage:-

Several different varieties of NoSQL databases have been created to support specific needs and use cases. These fall into four main categories:

* **Key-value data stores:** Key-value NoSQL databases emphasize simplicity and are very useful in accelerating an application to support high-speed read and write processing of non-transactional data. Stored values can be any type of binary object (text, video, JSON document, etc.) and are accessed via a key. The application has complete control over what is stored in the value, making this the most flexible NoSQL model. Data is partitioned and replicated across a cluster to get scalability and availability. For this reason, key value stores often do not support transactions. However, they are highly effective at scaling applications that deal with high-velocity, non-transactional data.
* **Document stores:** Document databases typically store self-describing JSON, XML, and BSON documents. They are similar to key-value stores, but in this case, a value is a single document that stores all data related to a specific key. Popular fields in the document can be indexed to provide fast retrieval without knowing the key. Each document can have the same or a different structure.
* **Wide-column stores:** Wide-column NoSQL databases store data in tables with rows and columns similar to RDBMS, but names and formats of columns can vary from row to row across the table. Wide-column databases group columns of related data together. A query can retrieve related data in a single operation because only the columns associated with the query are retrieved. In an RDBMS, the data would be in different rows stored in different places on disk, requiring multiple disk operations for retrieval.
* **Graph stores:** A graph database uses graph structures to store, map, and query relationships. They provide index-free adjacency, so that adjacent elements are linked together without using an index.

Multi-modal databases leverage some combination of the four types described above and therefore can support a wider range of applications.

1. **What is a column family in HBase?**

Columns in Apache HBase are grouped into column families. All column members of a column family have the same prefix. For example, the columns courses:history and courses:math are both members of the courses column family. The colon character (:) delimits the column family from the . The column family prefix must be composed of printable characters. The qualifying tail, the column family qualifier, can be made of any arbitrary bytes. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up an running.

Physically, all column family members are stored together on the filesystem. Because tunings and storage specifications are done at the column family level, it is advised that all column family members have the same general access pattern and size characteristics.

1. **How many maximum number of columns can be added to HBase table?**

There is no hard limit to number of columns in HBase , we can have more than 1 million columns but usually three column families are recommended ( not more than three).

1. **Why columns are not defined at the time of table creation in HBase?**

The columns do not have to be defined at schema definition time and they can be added on the fly while the database is up and running.

A column qualifier is an index for a given data and it is added to a column family. Data within a column family is addressed via the column qualifier. Column qualifiers are mutable and they may vary between rows. They do not have data types and they are always treated as arrays of bytes.

A row key, column family and column qualifier form a cell that has a value and timestamp that represents the value’s version. Values also do not have data types and they are always treated as arrays of bytes. A timestamp is recorded for each value and it is the time on the region server when the value was written.

All cell’s values are stored in a descending order by its timestamp. When values are retrieved and if the timestamp is not provided then HBase will return the cell value with the latest (the most recent) timestamp. If a timestamp is not specified during the write, the current timestamp is used.

The maximum number of versions (timestamps) for a given column to store is part of the column schema. It is specified at table creation. It can be specified via alter table command as well. The default value is 1. The minimum number of versions can be also set up per column family. You can also globally set up a maximum number of versions per column.

1. **How does data get managed in HBase?**

The Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster. The Data Model in HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.

* **Tables** – The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server. The figure above shows a representation of a Table.
* **Rows** – A row is one instance of data in a table and is identified by a rowkey. Rowkeys are unique in a Table and are always treated as a byte[].
* **Column Families** – Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied. Hence it’s important that proper care be taken when designing Column Families in table.
* **Columns** – A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon – example: columnfamily:columnname. There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.
* **Cell** – A Cell stores data and is essentially a unique combination of rowkey, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[].
* **Version** – The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.

7. What happens internally when new data gets inserted into HBase table?

**HBase Write Steps (1)**

When the client issues a Put request, the first step is to write the data to the write-ahead log, the WAL:

- Edits are appended to the end of the WAL (Write Ahead Log) file that is stored on disk.

- The WAL is used to recover not-yet-persisted data in case a server crashes.

**HBase Write Steps (2)**

Once the data is written to the WAL, it is placed in the MemStore. Then, the put request acknowledgement returns to the client.

**HBase MemStore**

The MemStore stores updates in memory as sorted KeyValues, the same as it would be stored in an HFile. There is one MemStore per column family. The updates are sorted per column family.

**HBase Region Flush**

When the MemStore accumulates enough data, the entire sorted set is written to a new HFile in HDFS. HBase uses multiple HFiles per column family, which contain the actual cells, or KeyValue instances. These files are created over time as KeyValue edits sorted in the MemStores are flushed as files to disk.

Note that this is one reason why there is a limit to the number of column families in HBase. There is one MemStore per CF; when one is full, they all flush. It also saves the last written sequence number so the system knows what was persisted so far.

The highest sequence number is stored as a meta field in each HFile, to reflect where persisting has ended and where to continue. On region startup, the sequence number is read, and the highest is used as the sequence number for new edits.

**HBase HFile**

Data is stored in an HFile which contains sorted key/values. When the MemStore accumulates enough data, the entire sorted KeyValue set is written to a new HFile in HDFS. This is a sequential write. It is very fast, as it avoids moving the disk drive head.

**HBase HFile Structure**

An HFile contains a multi-layered index which allows HBase to seek to the data without having to read the whole file. The multi-level index is like a b+tree:

Key value pairs are stored in increasing order

Indexes point by row key to the key value data in 64KB “blocks”

Each block has its own leaf-index

The last key of each block is put in the intermediate index

The root index points to the intermediate index

The trailer points to the meta blocks, and is written at the end of persisting the data to the file. The trailer also has information like bloom filters and time range info. Bloom filters help to skip files that do not contain a certain row key. The time range info is useful for skipping the file if it is not in the time range the read is looking for.