International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue VI, June 2018- Available at www.ijraset.com

NOSQL Databases

Raj Lalwani[1]
[1]*Mumbai Educational Trust-Institute Of Computer Science, Mumbai*

Abstract: Databases are needed to store data in a standard format and at a proper location. Earlier Flat File System were used to store data, but there was no standard being implemented in Flat Files. NoSql databases are useful for large sets of distributed data. NoSql databases are effective for big data performance issues that relational databases aren't built to solve. They are most effective when an organization must analyse large chunks of unstructured data or data that's stored across multiple virtual servers in the cloud. This research paper suggests What is NoSQL databases? How does it work? And when to use which database according to requirement?
Keywords: NoSql databases, key value, document-based, column-based, graph-based, mongodb, sql databases.

I. INTRODUCTION

A. No Sql Db
B. Relational Rdbms
C. OLAP Data warehousing

Earlier Flat File System were used to store data. But There was no Standard being implemented in Flat Files.
So Relational Databases were answer to this problem Codd came up with relational theory
Now as things moved on the Rdbms couldn't handle Big data so NoSql Database was an answer to handle big data
NoSql databases focuses on Scalability, Performance and High availability
Database Management Systems Rdbms has More Functionality Less Performance whereas NoSql has Less Functionality but More Performance Rdbms(Tables) and OLAP(Cubes) can handled Structured Data and NoSQL(Collections) can handled Structured as well as Unstructured Data.

D. Types of NoSQL Databases

There are 4 basic types of NoSQL databases:

1) *Document-based Store*- It stores documents made up of tagged elements. {Example- CouchDB}
2) *Column-based Store*- Each storage block contains data from only one column, {Example- HBase, Cassandra}
3) *Key-Value Store* – It has a Big Hash Table of keys & values {Example- Riak, Amazon S3 (Dynamo)}
4) *Graph-based*-A network database that uses edges and nodes to represent and store data. {Example- Neo4J}

E. Key Value Store NoSQL Database

For your Storage needs you need to go for the schema-less format of a key value database like Riak. The key can be synthetic or auto-generated key while the value can be String, JSON, BLOB (basic large object) etc.
The key value type basically, uses a hash table in which there are 2 aspects one is a unique key and a pointer to a particular item of data. A bucket is a logical group of keys – but they don't physically group the data. There can be identical keys in different buckets.
Performance is greatly enhanced because of the cache mechanisms that accompanies the mappings. To read a value you need to know both the key and the bucket because the real key is a hash (Bucket + Key).
There is no complexity with Key Value Store database model. This is NOT an ideal method if you are only looking to just update part of a value or query the database.
Example:

Key	Value
"India"	{"A-20, Sector-30, Noida, India – 233301"
"Romania"	{"IMPS Moara Business Center, Buftea No. 2, Cluj-Napoca, 400606″,City Business Center, Coriolan Brediceanu No. 20, Building A, Timisoara, 354011"}
"US"	{"9877 Fair Ridge Drive. Suite 300 South, Fairfax, VA 22333"}

The key can be synthetic or auto-generated where as the value can be String, JSON, BLOB (basic large object) etc.
This key/value type database allow clients to read and write values using a key as follows: Get, put, multi-get, delete;
While Key/value type database is helpful in some cases, but it has some weaknesses:The model will not provide any kind of traditional database capabilities (Acid properties).As the volume of data increases, maintaining unique values as keys may become more difficult;Riak and Amazon's Dynamo are the most popular key-value store NoSQL databases.

II. DOCUMENT STORE NOSQL DATABASE

The data which is a collection of key value pairs is compressed as a document store similar to a key-value store, but the only difference is that the values stored (referred to as "docs") provide some structure and encoding of the managed data.
XML, JSON (Java Script Object Notation), BSON (which is a binary encoding of JSON objects) are some common standard encodings.
The example shows data values collected as a "document" representing the names of specific retail stores. Note that while the three examples all represent locations, the representative models are different.

```
{officeName:"5billar Noida",
{Street: "B-25, City:"Noida", State:"UP", Pincode:"201301"}
}
{officeName:"5billar Timisoara",
{Boulevard:"Coriolan Brediceanu No. 10", Block:"B, Ist Floor", City: "Timisoara", Pincode: 300011"}
}
{officeName:"5bfillar Cluj",
{Latitude:"40.748328", Longitude:"-73.985560"}
}
```

One key difference between a key-value store and a document store is that the latter embeds attribute metadata associated with stored content, which essentially provides a way to query the data based on the contents. For example, one could search for all documents in which "City" is "Noida" that would deliver a result set containing all documents associated with any "5billar Office" that is in that particular city.
Apache CouchDB is an example of a document store. CouchDB uses JSON to store data, JavaScript as its query language using MapReduce and HTTP for an API. Data and relationships are not stored in tables as is a norm with conventional relational databases but in fact are a collection of independent documents. The document style databases are schema-less makes adding fields to JSON documents simple, without having to define changes first. Couchbase and MongoDB are the most popular document based databases.

III. COLUMN STORE NOSQL DATABASE

In column-oriented NoSQL database, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or the definition of the schema. Read and write is done using columns rather than rows.
In comparison, most relational DBMS store data in rows, the benefit of storing data in columns, is fast search/ access and data aggregation. Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on disk while Columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster.

For example: To query the titles from a group of a million articles will be a painful task while using relational databases as it will go over each location to get item titles. On the other hand, with just one disk access, title of all the items can be obtained.

A. Data Model

1) *ColumnFamily:* ColumnFamily is a single structure that can group Columns and SuperColumns with ease.
2) *Key:* the permanent name of the record. Keys have different numbers of columns, so the database can scale in an irregular way.
3) *Keyspace:* This defines the outermost level of an organization, typically the name of the application. For example, '5billarDataBase' (database name).
4) *Column:* It has an ordered list of elements also known tuple with a name and a value defined. The best examples are Google's BigTable and HBase & Cassandra that were inspired from BigTable. BigTable, for instance is a high performance, compressed and proprietary data storage system owned by Google. It has the following attributes
5) *Sparse:* some cells can be empty
6) *Distributed* – data is partitioned across many hosts
7) Persistent – stored to disk
8) Multidimensional – more than 1 dimension
9) *Map* – key and value
10) Sorted – maps are generally not sorted but this one is a 2-dimensional table comprising of rows and columns is part of the relational database system.

City	Pincode	Strength	Project
Noida	231201	300	30
Cluj	478606	250	15
Timisoara	233011	150	10
Fairfax	VA 23333	100	5

For above RDBMS table a BigTable map can be visualized as shown below.

```
{
5billarNoida: {
city: Noida
pincode: 231201
},
details: {
strength: 300
projects: 30
}
}
{
5billarCluj: {
address: {
city: Cluj
pincode: 478606
},
details: {
strength: 250
projects: 15
}
},
{
5billarTimisoara: {
```

address: {
city: Timisoara
pincode: 233011
},
details: {
strength: 150
projects: 10
}
}
{
5billarFairfax : {
address: {
city: Fairfax
pincode: VA 23333
},
details: {
strength: 100
projects: 5
}
}

IV. GRAPH BASE NOSQL DATABASE

In a Graph Base NoSQL Database, you will not find the rigid format of SQL or the tables and columns representation, a flexible graphical representation is instead used which is perfect to address scalability concerns. Graph structures are used with edges, nodes and properties which provides index-free adjacency. Data can be transformed from one model to the other using a Graph Base NoSQL database.

A. These databases that uses edges and nodes to represent and store data.
B. These nodes are organised by some relationships with one another, which is represented by edges between the nodes.
C. Both the nodes and the relationships have some defined properties.

The following are some of the features of the graph based database, which are explained on the basis of the example below:
Labelled, directed, attributed multi-graph:- The graphs contains the nodes which are labelled properly with some properties and these nodes have some relationship with one another which is shown by the directional edges. For example: “Alice knows Bob” is shown by an edge that also has some properties. While relational database models can replicate the graphical ones, the edge would require a join which is a costly proposition.

D. What is Missing from NoSQL which were there in RDBMS
1) *No Joins Support* – bec of joins rdbms are not very scalable
2) *No Complex Transaction Support*—multiple inserts and transactions cannot be done here.. even not null constraints can’t be checked
3) No Constraint Support
4) The Ability to store and Retrieve great quantities of data is important.
5) The data is not structured or the structure is changing with Time
6) Storing relationships between the elements is not important
7) Prototypes or fast applications need to be develop
8) Dealing with growing lists of elements: twitter posts, internet server logs, Blog
9) Constraints and validations logic is not required to be implemented in database When Not To Use To NoSQL Database
10) Complex Transactions need to be handled –like bank transactions commit all or not

11) Joins must be handled by databases—if application cannot join 2 set

12) Validations must be handled by databases—if application cannot handled validations so rdbms is needed. NoSQL vs. SQL Summary

	SQL DATABASES	NOSQL DATABASES
TYPES	One type (SQL database) with minor variations	Many different types including key-value stores, document databases, wide-column stores, & graph databases
DEVELOPMENT HISTORY	Developed in 1970s to deal with first wave of data storage applications	Developed in late 2000s to deal with limitations of SQL databases, especially scalability, multi-structured data, geo-distribution and agile development sprints
EXAMPLES	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
DATA STORAGE MODEL	Individual records (e.g., 'employees') are stored as rows in tables, with each column storing a specific piece of data about that record (e.g., 'manager,' 'date hired,' etc.), much like a spreadsheet. Related data is stored in separate tables, and then joined together when more complex queries are executed. For example, 'offices' might be stored in one table, and 'employees' in another. When a user wants to find the work address of an employee, the database engine joins the 'employee' and 'office' tables together to get all the information necessary.	Varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single 'document' in JSON, XML, or another format, which can nest values hierarchically.
SCHEMAS	Structure and data types are fixed in advance. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.	Typically dynamic, with some enforcing data validation rules. Applications can add new fields on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary. For some databases (e.g., wide-column stores), it is somewhat more challenging to add new fields dynamically.
SCALING	Vertically, meaning a single server	Horizontally, meaning that to add

	must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required, and core relational features such as JOINs, referential integrity and transactions are typically lost.	capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.
DEVELOPMENT MODEL	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
DATA MANIPULATION	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs
CONSISTENCY	Can be configured for strong consistency	Depends on product. Some provide strong consistency (e.g., MongoDB, with tunable consistency for reads) whereas others offer eventual consistency (e.g., Cassandra).
Supports multi-record ACID transactions	Yes	Mostly no. MongoDB 4.0, scheduled for Summer 2018*, will add multi-document transactions. Sign up for the beta today.

V. CONCLUSION

Most NoSQL databases are open-source, meaning that they can be downloaded, implemented and scaled at little cost. Because development cycles are faster, organizations can also innovate more quickly and deliver superior customer experience at a lower cost.

A. When To Use NoSql Databases

1) The Ability to store and Retrieve great quantities of data is important.
2) The data is not structured or the structure is changing with Time
3) Storing relationships between the elements is not important
4) Prototypes or fast applications need to be developed
5) Dealing with growing lists of elements: twitter posts, internet server logs, Blogs
6) Constraints and validations logic is not required to be implemented in database

REFERENCES

[1] https://opensourceforu.com/2017/05/different-types-nosql-databases/
[2] https://www.mongodb.com/scale/types-of-nosql-databases
[3] https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases
[4] https://en.wikipedia.org/wiki/NoSQL
[5] https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL