SEMINAR REPORT

NO SQL DATABASES

Submitted by: Saurabh Jha Enrollment no: U101113FCS221 Batch: S6

TABLE OF CONTENTS

Contents

INTRODUCTION

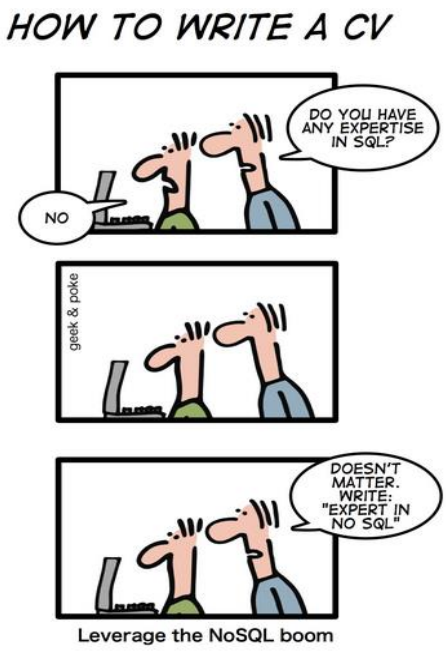
What is NoSQL?

According to Wikipedia, a NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling, and finer control over availability.

Also, as mentioned in MongoDB.org, NoSQL encompasses a wide variety of different database technologies that were developed in response to a rise in the volume of data stored about users, objects and products, the frequency in which this data is accessed, and performance and processing needs. Relational databases, on the other hand, were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the cheap storage and processing power available today.

The need for NoSQL

For the last 20 years or so, relational databases have been the solutions to persistence mechanisms. They had a very firm grounding in Set Theory and were very good at what they were meant to do; storing data in relational models lots of tables. This does offer a very big advantage of slicing and dicing of data, efficient querying (milliseconds), ACID properties etc. After Edgar F. Codd published the paper on normalization, the relational databases became the de facto standard of storing application data.



This model scaled very well until there was a data explosion. The rate at which data was getting generated increased multifold with the ubiquitous presence of internet/3G. After a certain point in time the relational type of databases fail to handle both scale and varied structure of data. At least not at a reasonable cost. Some might disagree, but the truth is that relational databases are good at scaling up, but that comes at an extra cost and there is a limit to how much you can scale vertically, instead you can use commodity hardware to scale out. The reason why relational databases are not good at horizontal scaling is because of the way data stored is distributed in multiple tables and the aggregation might require various joins. So, it is never possible to partition data accurately to store in different partitions across different servers.

Enter NoSQL databases. Here, data that is supposed to be consumed together is stored together. It is no longer stored in a normalized form and data duplicity is no longer considered evil, and this approach has its own advantages. The data can be stored on multiple commodity hardware at a much lower cost. This is what the likes of Google Big Table and Facebook – Cassandra have pioneered.

CURRENT SCENARIO

As of today, there are 150 types of NoSQL databases, all mentioned on NoSQL Database website. They vary among each other and are categorized as:

- Document database: Document databases pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
- Graph Stores: They are used to store information about networks, such as social connections. Graph stores include Neo4J and HyperGraphDB.
- Key Value Stores: They are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or "key"), together with its value. Examples of key-value stores are Riak and Voldemort. Some key-value stores, such as Redis, allow each value to have a type, such as "integer", which adds functionality.
- Wide-column stores: Wide-column stores such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

Are they even used?

The answer, Yes. NoSQL databases are now starting to be used by many big websites/companies. The most prominent example is Cassandra. Currently deployed at companies such as Apple Inc., eBay, Reddit, Hulu, Netflix, GitHub, CERN, Instagram, Twitter, Facebook and more than 1500 others, Cassandra is the most widely used NoSQL database system in the world. It was initially developed by Facebook to power their inbox search feature and was open sourced in 2008. Since then, it has garnered a lot of attention and is currently a top – level project at the Apache software foundation.

Another example of a NoSQL database is the Google's BigTable, which as of April 2015 has been implemented across all the services provided by the search giant including Google Maps, YouTube and Gmail. BigTable is a compressed, high performance, and proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies.

GENERAL FEATURES OF NOSQL

When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address, such as

- Large volumes of structured, semi-structured, and unstructured data
- Agile sprints, quick iteration, and frequent code pushes
- Object-oriented programming that is easy to use and flexible
- Efficient, scale-out architecture instead of expensive, monolithic architecture

Some features of NoSQL databases are:

Dynamic Schemas

Relational databases require that schemas be defined before you can add data. This fits poorly with agile development approaches, because each time you complete new features, the schema of your database often needs to change. So if you decide, a few iterations into development, that you'd like to store customers' favorite items in addition to their addresses and phone numbers, you'll need to add that column to the database, and then migrate the entire database to the new schema.

NoSQL databases are built to allow the insertion of data without a predefined schema. That makes it easy to make significant application changes in real-time, without worrying about service interruptions – which means development is faster, code integration is more reliable, and less database administrator time is needed.

Auto – Sharding

Because of the way they are structured, relational databases usually scale vertically – a single server has to host the entire database to ensure reliability and continuous availability of data. This gets expensive quickly, places limits on scale, and creates a relatively small number of failure points for database infrastructure.

NoSQL databases, on the other hand, usually support auto-sharding, meaning that they natively and automatically spread data across an arbitrary number of servers, without requiring the application to even be aware of the composition of the server pool. Data and query load are automatically balanced across servers, and when a server goes down, it can be quickly and transparently replaced with no application disruption.

GENERAL FEATURES OF NOSQL

Replication

Most NoSQL databases also support automatic replication, meaning that you get high availability and disaster recovery without involving separate applications to manage these tasks. The storage environment is essentially virtualized from the developer's perspective.

Integrated Caching

A number of products provide a caching tier for SQL database systems. These systems can improve read performance substantially, but they do not improve write performance, and they add complexity to system deployments. If your application is dominated by reads then a distributed cache should probably be considered, but if your application is dominated by writes or if you have a relatively even mix of reads and writes, then a distributed cache may not improve the overall experience of your end users.

Many NoSQL database technologies have excellent integrated caching capabilities, keeping frequently-used data in system memory as much as possible and removing the need for a separate caching layer that must be maintained.

PROS AND CONS

Pros

- ✓ Mostly open source.
- ✓ Horizontal scalability. There's no need for complex joins and data can be easily sharded and processed in parallel.
- ✓ Support for Map/Reduce. This is a simple paradigm that allows for scaling computation on cluster of computing nodes.
- ✓ No need to develop fine-grained data model – it saves development time.
- ✓ Easy to use.
- ✓ Very fast for adding new data and for simple operations/queries.
- ✓ No need to make significant changes in code when data structure is modified.
- ✓ Ability to store complex data types (for document based solutions) in a single item of storage.

Cons

- × Immaturity: Still lots of rough edges.
- × Possible database administration issues: NoSQL often sacrifices features that are present in SQL solutions "by default" for the sake of performance. For example, one needs to check different data durability modes and journaling in order not to be caught by surprise after a cold restart of the system. Memory consumption is one more important chapter to read up on in the database manual because memory is usually heavily used.
- × No indexing support (Some solutions like MongoDB have indexing but it's not as powerful as in SQL solutions).
- × No ACID (Some solutions have just atomicity support on single object level).
- × Bad reporting performance.
- × Complex consistency models: CAP theorem states that it's not possible to achieve consistency, availability and partitioning tolerance at the same time. NoSQL vendors are trying to make their solutions as fast as possible and consistency is most typical trade-off.
- × Absence of standardization: No standard APIs or query language. It means that migration to a solution from different vendor is more costly. Also there are no standard tools (e.g. for reporting)

SQL VS NOSQL

PROPERTY	*SQL*	*NOSQL*
TYPES	One type (SQL database) with minor variations	Many different types including key-value stores, document, wide-column stores, and graph databases
EXAMPLES	MySQL, PostgreSQL, 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. Separate data types are 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 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.
SCHEMA	Structure and data types are fixed in advance.	Typically dynamic. Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.
SCALING	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand.	Horizontally, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances.
DATA MANIPULATION	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs

CONCLUSION

Overall, the best platform for one depends greatly on what one needs from his/her database, and what kinds of queries he/she is demanding of the data. It also depends on what kind of database management plan one has in place, whether or not he/she is seeking a renovation on its current state.

One should embrace NoSQL but at the same time one should also understand that there's no need to give up using relational databases. The pragmatic approach is to use the best of both worlds. For transactional data, reporting and business intelligence, SQL solutions can do a more than satisfactory job. At the same time it is recommended to consider NoSQL for logging, caching, session storage and self-sufficient data like incoming email, product reviews, etc. It is also recommended to have evolutional development of applications and refactoring application modules in order to use NoSQL solutions on demand.

BIBLIOGRAPHY

The content used in this report has been gathered from a variety of sources and research papers. The sources have been cited below in no particular order:

- MongoDB Official Website
- Sphere Inc. Research Paper
- NoSQL Wikipedia
- Couch base Reference - NoSQL
- What, when and how of NoSQL databases – by Ashish Subodh Mittal