SEMINAR REPORT- NOSQL DATABASE

Group Members :- 1. Vaibhav Kaushik
2. Sujoy Roy
3. Rajat Rai
4. Rachit Trivedi
5. Aayushi Jain

TABLE OF CONTENTS

Contents

NOSQL DATABASE

INTRODUCTION

NoSQL database provides a way for storing and retrieval of data that is modeled in means other than the regular tabular relations used in relational databases. Incentives for this approach include simplicity of design, horizontal scaling, and finer control over availability.

It is also known as "Not Only Sql" to show that they support Sql query. It is an advanced approach to data management and database designing. These database are schema free, support replication, have simple API and are capable of handling huge data.

NoSQL encompasses a wide variety of different database technologies that were developed to respond to the call of rise in the amount of data stored about users, objects and products, the rate at which this data is accessed, the performance and processing requirements. Relational databases, on the other hand, were not designed to keep up with the scale and agility challenges that the modern applications face, nor were they built to efficiently utilize the advent of the cheap storage and processing power available today's world.

HISTORY

The term NoSQL was given by Carlo Strozzi. He coined this term in the year of 1988. It was this term which he used to name his Open Source, Ultra-Light Weight, Database which never had a SQL interface. During the early 2009, last.fm had this desire to organize an event focusing upon open-source distributed databases, Eric Evans, who was a "Rackspace" employee, reused this term to refer databases which were non-relational, distributed, and does not conform to atomicity, consistency, isolation, and durability - four obvious features of traditional relational database systems. In the very same year, the "no:sql(east)" conference took place in Atlanta, USA, NoSQL was discussed and debated over a thousand times. And then, discussion and practice of NoSQL got a push ahead, and NoSQL saw an unprecedented and un-imaginable growth.

NEED OF NOSQL

Over decades of software development, we have been using database in form of SQL (to store relational tables). But in past few years due to the rapid increase in the usage of Web Application like Facebook, WhatsApp, Google, Yahoo, etc. result's in the rise of Database Management which approaches of simple design, faster speed and faster scaling of Database. Example: - Big Data, Massive Real time operations. This

does offer a very big advantage of slicing and dicing of data, efficient querying (which comes in milliseconds), ACID properties, and many more.

ACID provides guidelines as to how changes are applied to a database which in most cases are followed as rules. In a very simplified way, it states (in our own version):

- (A) When you do any operation that changes something into a database the change should work or fail as a whole, ie to say "Everything or Nothing"
- (C) It should be made sure that the database should always remain consistent (this is a very vast topic in itself and hence we do not go into the detail and safely assume that all readers know about this property)
- (I) If other operations are going on at the concurrently they shouldn't be able to see things in the middle of an update
- (D) If the power dies or system fails (hardware or software) the database needs to be able to pick itself back up; and if it says it finished applying an update, it needs to be certain.

After Sir Edgar F. Codd published a research paper on normalization, the relational databases became the de facto standard of storing application data. Here, with this approach the 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 and disadvantages. 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 in the art.

NOSQL CATEGORIES

There are 4 types of NoSQL:-

1. **Document Database –** It allows indexing of documents on the basis of not only it's primary identifier but also it's properties. It pair's 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. Ex:- Mongo.DB, Couch D.B, H. Base, Cassandra, Hypertable, etc.
2. **Graph Based Database –** It uses graph structure with nodes, edges and properties to represent and store data. Provides index free adjency i.e. that every element contains a direct pointer to it's adjacent element and no index lookups are necessary. They are used to store information about networks, such as social connections. Ex: - Neo4J, HyperGraphDB . Facebook open Graph, Flock D.B, etc.

3. **Key Value Database -** 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.
4. **Column Based Database –** It avoids consuming space when storing nulls by simply not storing a column when a value doesn't exist for that data. Ex: - Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

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.

CHALLENGES

The promise of the NoSQL database has generated a lot of enthusiasm, but there are yet many obstacles to overcome before they can appeal to mainstream enterprises. Here are a few of the top challenges.

1:- **Maturity**- Relational Database Management Systems have been around for a very long time. People who advocate NOSQL will argue that their advancing age is a sign of them being obsolete, but for most industries, the maturity of the RDBMS comes with a packaged deal of reassurance. For the most part, RDBMS systems are very dependable, stable and richly multi-functional. In comparison, most NOSQL alternatives are in pre-productive or beta versions with many key features yet to be infused into the system. Living on the technologically leading edge is an exciting prospect for many developers, but the industry/companies should approach it with extreme caution as they might still have some loop holes.

2: **Support** – Industry wants the assurance and dependency that if a keynote system fails, they will be able to get a competent support in a very timely manner. All RDMS productions go to a great deal to fulfill this requirement of the industry and provide a high level of enterprise support. On the contrary, most NOSQL systems are open sourced projects, and though there are one or more firms offering support for each NOSQL database, these very companies often are small start-ups without the global reach, support resources, or credibility of an oracle, Microsoft, or IBM can provide and hence it becomes a little less reliable when it comes to the support sector.

3:- **Analytics and business intelligence** - NOSQL systems have grown enough to meet the scaling demands of modern web 2.0 applications. As a result of which, most of their features are headed towards the needs of these applications. However, the data in an application has value to the business that goes beyond the reach of insert-read-update-delete cycle as in case of a stereotypical web application. Businesses look for more information in corporate databases to improve their productivity, competitive scaling, and business intelligence (bi) which are the key it issues for all medium to large companies dealing in the IT sector. NOSQL databases offer few ways for ad-hoc query and analysis. Even a simple query requires significant programming experience and commonly used (bi) tools do not provide connectivity to NOSQL. Some relief is provided by the newly born solutions such as hive or pig, which can provide easier access to data held in Hadoop clusters and perhaps eventually, other NOSQL databases. Quest soft wares has developed a product — toad for cloud databases — that can provide ad-hoc query capabilities to a variety of NOSQL databases.

4:- **Administration** - The design goals for NOSQL may be to provide a zero-admin solution, but the current reality falls well short of that goal. NOSQL today requires a lot of skill to install and a lot of effort to maintain.

5:- **Expertise** - There are literally lakhs of developers throughout the world, and in every business segmentation, who are familiar with RDBMS concepts and their programming. On the contrary, almost every NOSQL developer is in a learning phase. This situation will address naturally over time, but for now, it's far easier to find experienced RDBMS programmers or administrators than a NOSQL expert.

BENEFITS OF NOSQL OVER RDBMS

- **Schema Less:**

  NoSQL databases are schema-less and hence do not define any strict data structure or need to follow any such rule.

- **Dynamic and Agile:**

  NoSQL databases have good feasibility as far as growth is concerned. They can grow dynamically with the changing requirements. It can handle all the three structured, semi-structured and unstructured data types of data.

- **Scale Horizontally:**

  On the contrary to SQL databases and RDBMS approach which scale vertically, NoSQL scales horizontally by adding more servers and using concepts of sharing and replication. This very behavior of NoSQL suits with the cloud computing services such as Amazon Web Services (AWS) which allow you to handle virtual servers on the cloud which can be expanded horizontally as and when needed.

- **Better Performance:**

  All the NoSQL databases promise to deliver better and faster performance as compared to the normal RDBMS implementations.

Though as far as the limitations are concerned, since NoSQL is an entire set of databases (and not a single database), the limitations differ from different set of database to database. Some of these databases do not support ACID properties while some of them might be lacking in reliability. But each one of them has their own strengths and weakness due to which they are well suited for specific requirements.

DIFFERENCE BETWEEN RDBMS AND NOSQL

RDBMS

- It has more of structured and organized data set

- It has a more structured query language (SQL)

-The data and their relationships are stored in different separate tables.

- The Data Manipulation Language, Data Definition Language are very different as to that in NOSQL

- It has very strict consistency

- It has BASE Transaction

NoSQL

- Abbreviation for Not Only SQL

- There are no declarative query language as compared to SQL

- It has no predefined schema for the database

-There exists Key-Value pair storage, Column Store, Document Store, Graph databases

- The most astonishing feature is that of eventual consistency rather than having ACID properties

-It is more of unstructured and unpredictable data

- It follows CAP Theorem

- This type of databases prioritize high performance, high availability and scalability of data

When compared to RD, NoSQL databases are more scalable and provide better and superior performance ratios, and their data model addresses several issues that the relational models were not designed to address, such as: -

- The large volumes of structured, semi-structured, and unstructured data are not handled in RDBMS

- It has the flexibility and support for agile sprints, quick iteration, and frequent code pushes which is again also, faster as that of RDs.
- It supports object-oriented programming which is easy to use and very flexible.

In some short and small term, it is efficient, cost friendly, scale-out architecture instead of expensive, monolithic architecture.

SOME FEATURES OF NOSQL

1. **Dynamic Schema's –** The Relational databases require schemas to be defined before we can add the data. This particularly does not fit with agile development approaches, because every time you complete new features, the schema of your database is often needed to be changed. And hence if we decide, a few iterations into development, like that when we'd like to store customers' favourite items in addition to their addresses and phone numbers, you'll need to add those columns to the database, and then transport the entire database to the new schema.

   NoSQL databases are built to allow such operations such as the insertion of data without a schema that was previously defined. That makes it pretty easy to make significant application changes based on real-time needs, without worrying about service interruptions – which in turn implies that the development is faster, code integration is more reliable, and less database administration time is needed which is very helpful in every respect.
2. **Auto Sharing -** Because of the way these databases are structured, relational databases generally scale vertically – one server at a time has to host the entire database to make sure the reliability and continuous availability of data. This gets very expensive in very short amount to time, places limit on scales, and creates a comparatively small number of failure points for database infrastructure thus created.

   Whereas NoSQL databases, on the contrary, usually support auto-sharing, meaning to say that they are natively and automatically designed to spread data across an arbitrary number of servers, which is done without requiring the application even to be aware of the composition of the server cluster. Data and query load are automatically balanced on servers, and when one server is down, it can be quickly and transparently replaced with no application disruption which is a very useful procedure.
3. **Replication –** Mostly all NoSQL databases also support automatic replication, meaning to say that we get high availability and disaster recovery methods without involving different applications to manage these heavy tasks. The storage environment is essentially and most importantly virtualized if we look from a developer's perspective.

4. **Integrated Caching -** A lot of products provide a caching tier for SQL RD systems. These systems can improve read performance to a noticeable extent, but they do not improve write performance as much, and they often add complexity to system deployments. If your application is predominated by read operations then a distributed cache should probably be considered and chosen, but if your application is predominated by write operations or if you have a relatively even a mix of read operations and write operations, then in that case a distributed cache may not improve the overall experience of our users.
   Many NoSQL database technologies have extraordinary integrated caching capabilities, keeping frequently-used data in system memory as much as possible and eliminating the need for a separate caching layer that must be maintained which is cost effective as well.

PROS AND CONS

So as far as our words are concerned: -

PROS:-

- It is usually open source.
- They come with horizontal scalability options. There is no need for complex joins and hence data can easily be shared and processed in parallel at the same time.
- They come with built in support for Map/Reduce. This is a very simple paradigm that allows us for scaling of computations on cluster of computing nodes.
- It has no need to develop fine-grained data model – it saves much development time.
- It is very easy to use.
- It acts very fast when adding new data and for simple operations and queries.
- No need to make a lot of changes in code when data structure is modified.
- It possesses the ability to store complex data types (for document based solutions) in a single item of storage.

CONS:-

- It has some immaturity: Still lots of rough edges.
- Possible database administration issues: NoSQL often gives up features that are present in current SQL solutions "by default" for the namesake of performance. For example, someone needs to check different data durability modes and journaling in order not to be caught by awe and surprise after a cold restart of the systems. Memory consumption is one of the most important chapters to read up on in the database manual because it is the memory that is usually heavily used.
- It has no indexing support (Some solutions like MongoDB have indexing but it's not as good and powerful as in SQL solutions).
- Again it has no ACID (Some solutions have just atomicity support on single object level).
- It has bad reporting performances
- Complex consistency models: Talking about CAP theorem it states that it is not possible to achieve consistency, availability and partitioning tolerance at the same time. All the NoSQL vendors are trying to make their solutions as fast as possible and consistency is most typical trade-off.
- Absence of standardization: There are No standard APIs or query language are present till date. It means that the transport to a solution from different vendor is costlier. 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,	Through object-oriented APIs

e.g. SELECT fields FROM table

WHERE…

COMMUNITY CONTRIBUTIONS ON NOSQL

Michael Kopp has contributed in putting together the Apache Cassandra Fastpack providing with a different kind of monitoring for the Cassandra NoSql Database. It has the JMX measures for the Cassandra, servers for Cassandra server and clients, Business Transection, sample system profile and load monitoring.

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.

BIBILOGRAPHY

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
- W3Resource-mongodb-nosql.php
- Tutorial's Point

- International Journal of Advanced Research in Computer Science and Software Engineering (Research Paper)
- http://www.engpaper.net/nosql-research-papers.htm(ResearchPaper)
- http://www.informit.com/articles/article.aspx?p=2247310
- http://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases

_____ -o- _____