****

# **SEMINAR REPORT**

**Nosql Database**

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

# **Table of contents**

**Contents**

**Introduction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1  
History\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1  
Need Of NoSql\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2  
NoSql Categories\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3  
Challenges\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4  
Benefits Of NoSql Over RDBMS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5  
Difference Between NpSql And RDBMS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6  
Some features of NoSql\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7  
Pros And Cons\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8  
Sql Vs NoSql\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9  
Conclusion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10  
Bibilography\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11**

**NoSql Database**

# **Introduction**

NoSQL database provides a way for storing 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.

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

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.

# History

The term NoSQL was coined by Carlo Strozzi in the year 1998. He used this term to name his Open Source, Light Weight, DataBase which did not have an SQL interface. In the early 2009, when last.fm wanted to organize an event on open-source distributed databases, Eric Evans, a Rackspace employee, reused the term to refer databases which are non-relational, distributed, and does not conform to atomicity, consistency, isolation, durability - four obvious features of traditional relational database systems. In the same year, the "no:sql(east)" conference held in Atlanta, USA, NoSQL was discussed and debated a lot.And then, discussion and practice of NoSQL got a momentum, and NoSQL saw an unprecedented growth.

# Need of nosql - Motivation

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 (milliseconds), ACID properties, etc.

**ACID** provides principles governing how changes are applied to a database. In a very simplified way, it states (my own version):

* (A) when you do something to change a database the change should work or fail as a whole.
* (C) the database should remain consistent (this is a pretty broad topic)
* (I) if other things are going on at the same time they shouldn't be able to see things mid-update
* (D) if the system blows up (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 Edgar F. Codd published the paper on normalization, the relational databases became the de facto standard of storing application data.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.

# 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 dosen’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 many obstacles to overcome before they can appeal to mainstream enterprises. Here are a few of the top challenges.

1 – **Maturity** - RDBMS systems have been around for a long time. NOSQL advocates will argue that their advancing age is a sign of their obsolescence, but for most CIOs, the maturity of the RDBMS is reassuring. For the most part, RDBMS systems are stable and richly functional. In comparison, most NOSQL alternatives are in pre-production versions with many key features yet to be implemented. Living on the technological leading edge is an exciting prospect for many developers, but enterprises should approach it with extreme caution.

2 – **Support -** Enterprises want the reassurance that if a key system fails, they will be able to get timely and competent support. All RDMS vendors go to great lengths to provide a high level of enterprise support. In contrast, most NOSQL systems are open source projects, and although there are usually one or more firms offering support for each NOSQL database, these companies often are small start-ups without the global reach, support resources, or credibility of an oracle, Microsoft, or IBM.

**3 - Analytics and business intelligence** - NOSQL databases have evolved to meet the scaling demands of modern web 2.0 applications. Consequently, most of their feature set is oriented toward the demands of these applications. However, data in an application has value to the business that goes beyond the insert-read-update-delete cycle of a typical web application. Businesses mine information in corporate databases to improve their efficiency and competitiveness, and business intelligence (bi) is a key it issue for all medium to large companies. NOSQL databases offer few facilities for ad-hoc query and analysis. Even a simple query requires significant programming expertise and commonly used (bi) tools do not provide connectivity to NOSQL. Some relief is provided by the emergence of solutions such as hive or pig, which can provide easier access to data held in Hadoop clusters and perhaps eventually, other NOSQL databases. Quest software 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 millions of developers throughout the world, and in every business segment, who are familiar with RDBMS concepts and programming. In contrast, almost every NOSQL developer is in a learning mode. 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 being schema-less do not define any strict data structure.
* **Dynamic and Agile:**  
  NoSQL databases have good tendency to grow dynamically with changing requirements. It can handle structured, semi-structured and unstructured data.
* **Scale Horizontally:**  
  In contrast to SQL databases which scale vertically, NoSQL scales horizontally by adding more servers and using concepts of sharding and replication. This behavior of NoSQL fits with the cloud computing services such as Amazon Web Services (AWS) which allows you to handle virtual servers which can be expanded horizontally on demand.
* **Better Performance:**  
  All the NoSQL databases claim to deliver better and faster performance as compared to traditional RDBMS implementations.

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

# difference between rdbms and nosql

**RDBMS**  
- Structured and organized data   
- Structured query language (SQL)   
- Data and its relationships are stored in separate tables.   
- Data Manipulation Language, Data Definition Language   
- Tight Consistency   
- BASE Transaction

**NoSQL**  
- Stands for Not Only SQL  
- No declarative query language  
- No predefined schema   
- Key-Value pair storage, Column Store, Document Store, Graph databases  
- Eventual consistency rather ACID property   
- Unstructured and unpredictable data  
- CAP Theorem   
- Prioritizes high performance, high availability and scalability

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

1. **Dynamic Schema’s -** 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.
2. **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.
3. **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.
4. **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 shared 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 |

# **Community contribution on NOsql**

**Michael Kopp has put together the**[**Apache Cassandra Fastpack**](https://community.dynatrace.com/community/pages/viewpage.action?pageId=65732667)**, providing out-of-the-box monitoring for the Cassandra NoSQL database. It consists of predefined JMX Measures for Cassandra, sensors for Cassandra server and clients, business transactions, a sample system profile, and a dashboard for 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>
* <https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&cad=rja&uact=8&ved=0ahUKEwiBhcDl16nMAhVW1I4KHR09CWAQFghBMAY&url=http%3A%2F%2Fwww.datastax.com%2Fwp-content%2Fuploads%2F2012%2F10%2FWP-DataStax-WhyNoSQL.pdf%3F2&usg=AFQjCNHwpM2VVom9xd54wszQBG2z586IWQ&sig2=WTlYqxjCgeIgJEslvzYW-A&bvm=bv.119745492,d.c2E>
* **SilvanWeber, “ NoSQLDatabases ” http://www.christof-strauch.de /nosqldbs.pdf**