***NO-SQL***

Introduction

In the computing system (web and business applications), there are enormous data that comes out every day from the web. A large section of these data is handled by Relational database management systems (RDBMS). The idea of relational model came with E.F.Codd’s 1970 paper "A relational model of data for large shared data banks" which made data modeling and application programming much easier. Beyond the intended benefits, the relational model is well-suited to client-server programming and today it is predominant technology for storing structured data in web and business applications. Relational databases use primary and foreign keys and have strict constraints when you manipulate the tables’ data. These databases are good for smaller data storage requirements, but you need “big data” capabilities to manage large queries. This is the goal of NoSQL. NoSQL databases work entirely different than relational databases, so you need to learn how to work with NoSQL to properly manage big data queries. If you don’t implement NoSQL properly, you can actually slow down your website or applications that use the NoSQL database system.

What is Database?

A **database** is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports process requiring information. A **database management system** (**DBMS**) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. we have many database management systems.

1. RDBMS-Structured data, Tables
2. OLAP-Data warehousing, Cubes
3. NoSQL-Structured or unstructured data, Collections

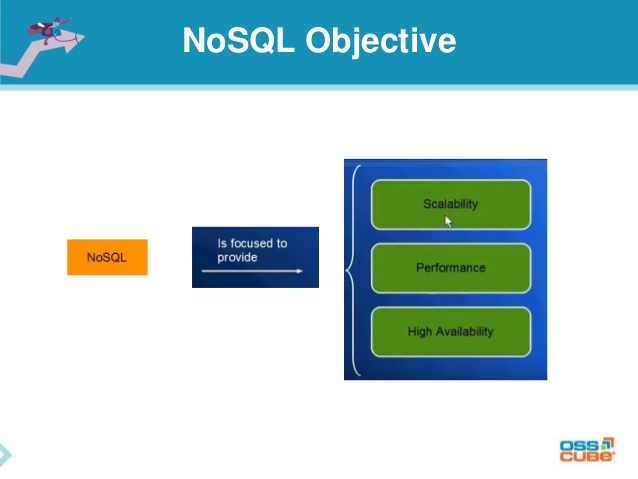
In early days we have Flat file system were created. But there is problem that there is no standard of storing data and no standard of communicating with data. Everyone has implemented their own protocols and it was creating lot of inefficiency, so Relational Data base was introduced in 1970’s which is standardized. When everything is perfect, suddenly we have a big data scenario which is a recent scenario and RDBMS unable to cope with huge amount of data. So the answer was NOSQL databases.

What is NoSQL?

NoSQL stand for Not only SQL. NoSQL is a whole new way of thinking about a database. Though NoSQL is not a relational database, the reality is that a relational database model may not be the best solution for all situations.



OBJECTIVES:



WHY NoSQL?

Relational databases were born in the era of mainframes and business applications – long before the Internet, the cloud, big data, mobile and now, the Digital Economy. In fact, the first commercial implementation was released by Oracle in 1979. These databases were engineered to run on a single server – the bigger, the better. The only way to increase the capacity of these databases was to upgrade the servers – processors, memory, and storage – to scale up.

NoSQL databases emerged as a result of the exponential growth of the Internet and the rise of web applications. Google released the BigTable research in 2006, and Amazon released the Dynamo research paper in 2007. These databases were engineered to meet a new generation of enterprise requirements:

 The need to

1. ***develop with agility***

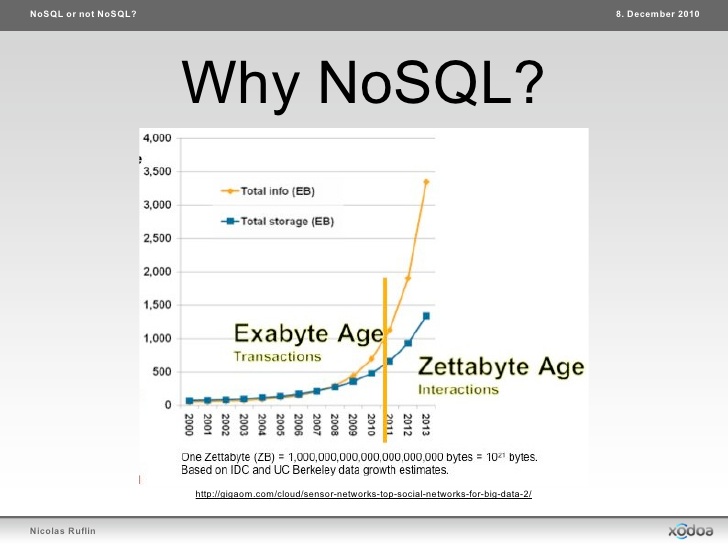
***2 .operate at any scale.***

**Develop with Agility**

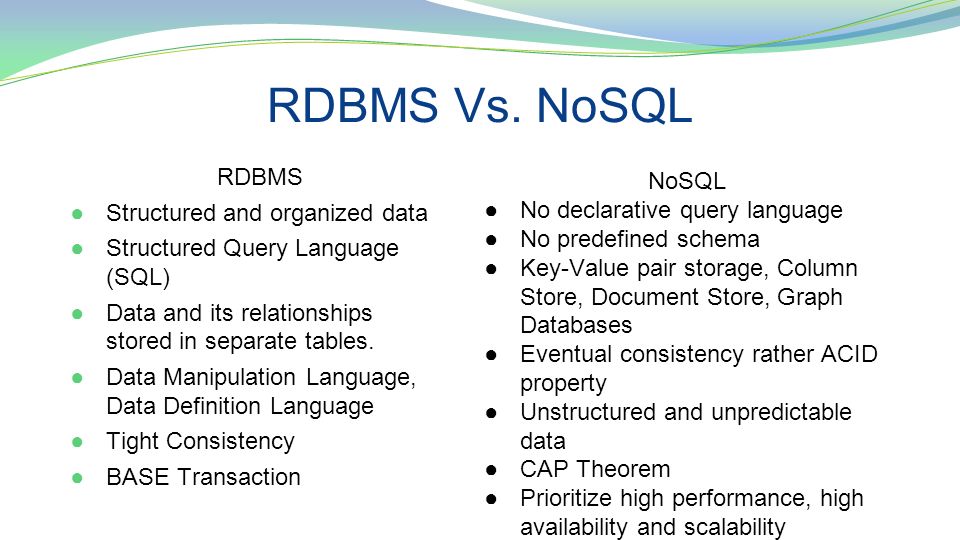
To remain competitive in the Digital Economy, enterprises must innovate – and now they have to do it faster than ever before. As this innovation centers on the development of modern web, mobile, and IoT applications, developers have to deliver applications and services faster than ever before. Speed is critical, but so is agility, since these applications evolve far more rapidly than legacy applications like ERP.  Relational databases are a major roadblock to agility, since they do not support agile development very well due to their fixed data model.

**Flexibility for Faster Development**

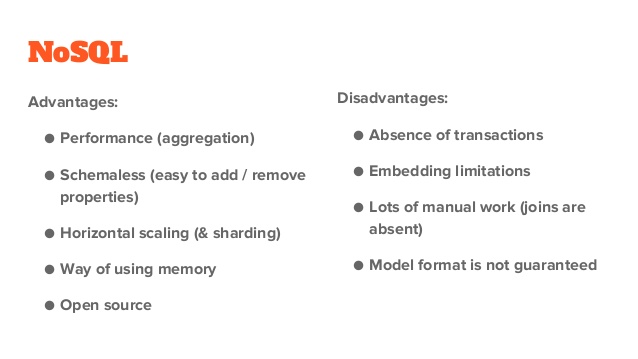
A core principle of agile development is adapting to evolving application requirements: when the requirements change, the data model also changes. This is a problem for relational databases because the data model is fixed and defined by a static schema. So in order to change the data model, developers have to modify the schema, or worse, request a “schema change” from the database administrators. This slows down or stops development, not only because it is a manual, time-consuming process, but it also impacts other applications and services.



NoSQL Vs RDBMS:

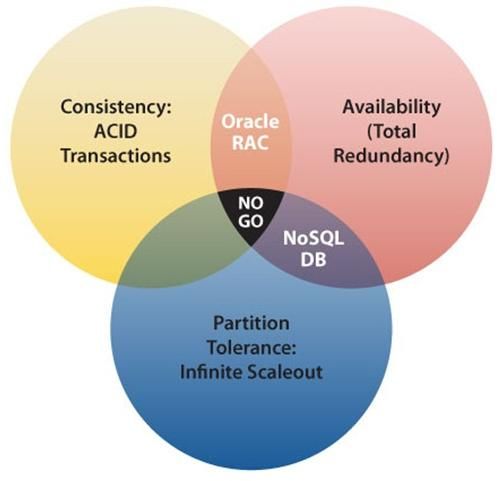


Advantages and Disadvantages:



CAP THEOREM:

* Consistency: every read would get you the most recent write
* Availability: every node (if not failed) always executes queries
* Partition-tolerance: even if the connections between nodes are down, the other two (A & C) promises, are kept.



Distributed systems allow us to achieve a level of computing power and availability that were simply not available in yesteryears. Our systems have higher performance, lower latency, and near 100% up-time in data centers that span the entire globe. Best of all, the systems of today are run on commodity hardware that is easily obtainable and configurable with costs approaching $0.

All of this computing power and benefit comes at a price, however. Distributed systems are more complex than their single-network counterparts. There are many more tools and skills that need to be acquired in order to create a truly scalable, high performance system. Understanding the complexity incurred in distributed systems, making the appropriate trade-offs for the task at hand (CAP), and selecting the right tool for the job are all critical skills in a world where computing systems are moving out, not up.

TYPES OF DATABASES IN NOSQL

NoSQL is four things:

### Document-oriented databases:

The data which is a collection of key value pairs is compressed as a document store quite similar to a key-value store, but the only difference is that the values stored (referred to as “documents”) 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

**Examples:** [**Couchbase**](http://www.couchbase.com/)**,** [**CouchDB**](http://couchdb.apache.org/)**,** [**MongoDB**](http://www.mongodb.org/)**,** [**Riak**](http://sendgrid.com/blog/what-is-nosql/basho.com/riak/)

## Key-Value Store databases

The key value type basically, uses a hash table in which there exists a unique key and a pointer to a particular item of data. A bucket is a logical group of keys – b

**Examples:** [**Memcached**](http://memcached.org/)**,** [**Redis**](http://redis.io/)**, Riak,** [**VoltDB**](http://voltdb.com/)

## Graph databases

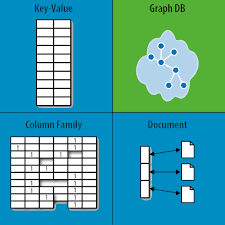
In a Graph Base NoSQL Database, you will not find the rigid format of SQL or the tables and columns representation, a flexible graphical representation is instead used which is perfect to address scalability concerns. Graph structures are used with edges, nodes and properties which provides index-free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

**Examples:** [**InifiniteGraph**](http://www.objectivity.com/infinitegraph#.U3Y0GS_rmVI)**,** [**Neo4J**](http://www.neo4j.org/)**,** [**OrientDB**](http://www.orientechnologies.com/orientdb/)

### Column store databases.

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.

**Examples:** [**Apache, HBase**](http://hbase.apache.org/)**,** [**Cassandra**](http://sendgrid.com/blog/what-is-nosql/cassandra.apache.org)**,** [**Google’s BigTable**](http://research.google.com/archive/bigtable.html)



.