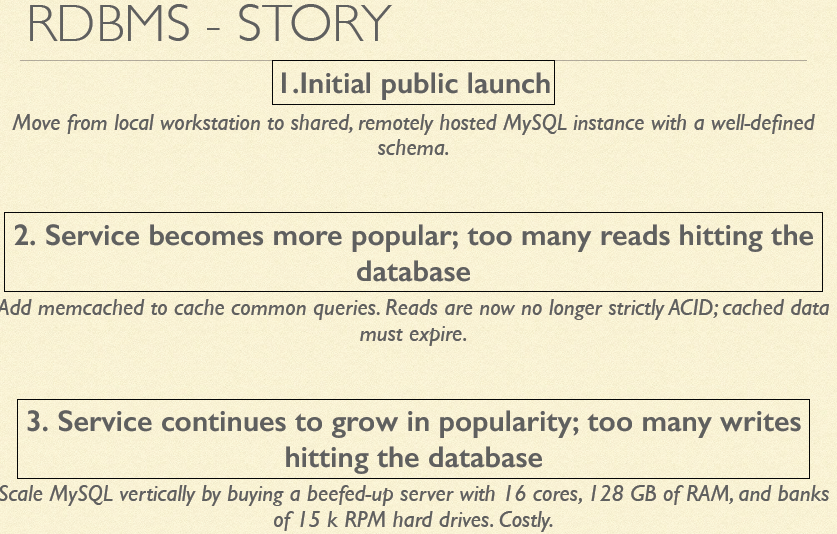
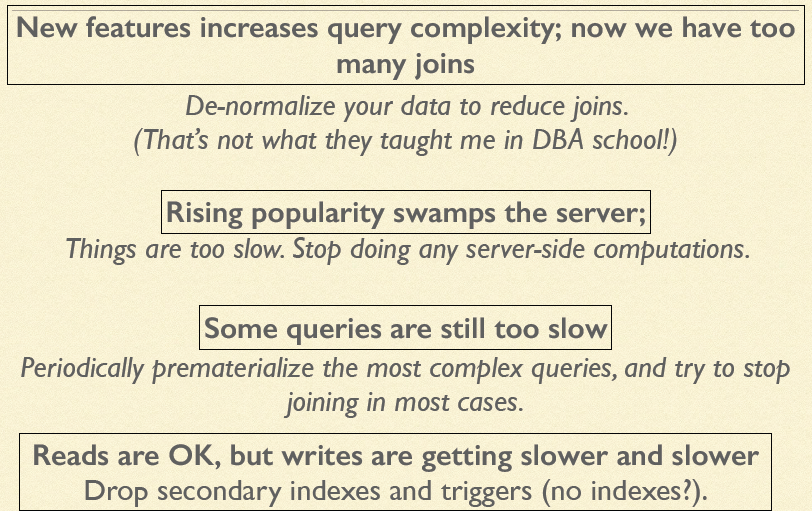
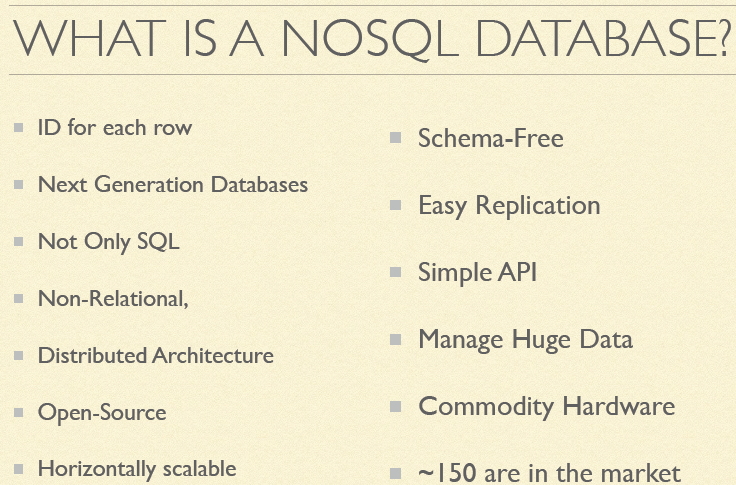
**NoSQL**



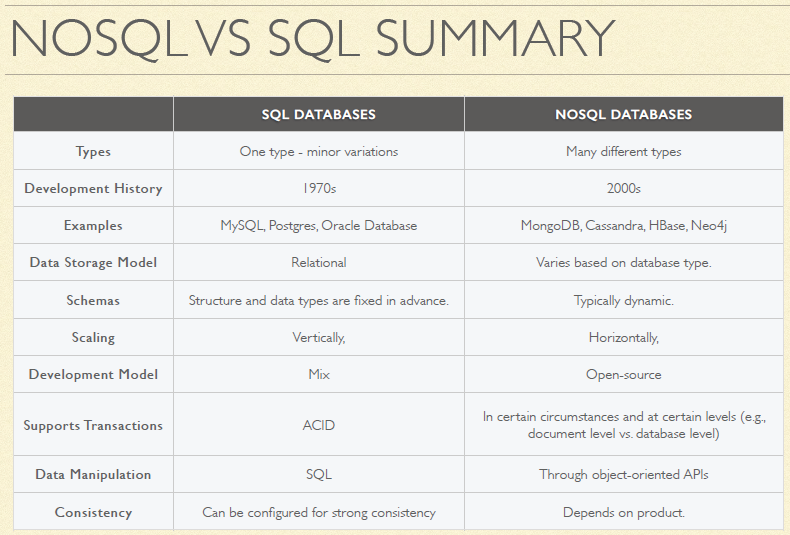


A **NoSQL** (often interpreted as **Not only SQL**[[1]](http://en.wikipedia.org/wiki/NoSQL#cite_note-1)[[2]](http://en.wikipedia.org/wiki/NoSQL#cite_note-2)) database provides a mechanism for [storage](http://en.wikipedia.org/wiki/Computer_data_storage) and [retrieval](http://en.wikipedia.org/wiki/Data_retrieval) of data that is modeled in means other than the tabular relations used in [relational databases](http://en.wikipedia.org/wiki/Relational_database). Many NoSQL stores compromise consistency (in the sense of the [CAP theorem](http://en.wikipedia.org/wiki/CAP_theorem)) in favor of availability and partition tolerance



SQL vs NoSQL: High-Level Differences

* SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL database are primarily called as non-relational or distributed database.
* SQL databases are table based databases whereas NoSQL databases are document based, key-value pairs, graph databases or wide-column stores. This means that SQL databases represent data in form of tables which consists of n number of rows of data whereas NoSQL databases are the collection of key-value pair, documents, graph databases or wide-column stores which do not have standard schema definitions which it needs to adhered to.
* SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data.
* SQL databases are vertically scalable whereas the NoSQL databases are horizontally scalable. SQL databases are scaled by increasing the horse-power of the hardware. NoSQL databases are scaled by increasing the databases servers in the pool of resources to reduce the load.
* SQL databases uses SQL ( structured query language ) for defining and manipulating the data, which is very powerful. In NoSQL database, queries are focused on collection of documents. Sometimes it is also called as UnQL (Unstructured Query Language). The syntax of using UnQL varies from database to database.
* SQL database examples: MySql, Oracle, Sqlite, Postgres and MS-SQL. NoSQL database examples: MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb
* For complex queries: SQL databases are good fit for the complex query intensive environment whereas NoSQL databases are not good fit for complex queries. On a high-level, NoSQL don’t have standard interfaces to perform complex queries, and the queries themselves in NoSQL are not as powerful as SQL query language.
* For the type of data to be stored: SQL databases are not best fit for hierarchical data storage. But, NoSQL database fits better for the hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data. NoSQL database are highly preferred for large data set (i.e for big data). Hbase is an example for this purpose.
* For scalability: In most typical situations, SQL databases are vertically scalable. You can manage increasing load by increasing the CPU, RAM, SSD, etc, on a single server. On the other hand, NoSQL databases are horizontally scalable. You can just add few more servers easily in your NoSQL database infrastructure to handle the large traffic.
* For high transactional based application: SQL databases are best fit for heavy duty transactional type applications, as it is more stable and promises the atomicity as well as integrity of the data. While you can use NoSQL for transactions purpose, it is still not comparable and sable enough in high load and for complex transactional applications.
* For support: Excellent support are available for all SQL database from their vendors. There are also lot of independent consultations who can help you with SQL database for a very large scale deployments. For some NoSQL database you still have to rely on community support, and only limited outside experts are available for you to setup and deploy your large scale NoSQL deployments.
* For properties: SQL databases emphasizes on ACID properties ( Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem ( Consistency, Availability and Partition tolerance )
* For DB types: On a high-level, we can classify SQL databases as either open-source or close-sourced from commercial vendors. NoSQL databases can be classified on the basis of way of storing data as graph databases, key-value store databases, document store databases, column store database and XML databases.
* SQL is not suitable for handling terabytes for data, where as NoSQL can handle tera bytes of data easily.



Types of NoSQL Stores

1. Column Oriented Database

A **column-oriented DBMS** is a [database management system](http://en.wikipedia.org/wiki/Database_management_system) (DBMS) that stores data tables as sections of columns of data rather than as rows of data. Data in columns stored nearby as opposed to the rows being nearby. Unlike the row-based systems (Key Value and Document Oriented DBs) these are as the name implies oriented to storing data in columns.  Where Document Oriented DBs excel at OLTP, Column Oriented DBs excel at OLAP (on line analytic processing).

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RowId** | **EmpId** | **Lastname** | **Firstname** | **Salary** |
| 001 | 10 | Smith | Joe | 40000 |
| 002 | 12 | Jones | Mary | 50000 |
| 003 | 11 | Johnson | Cathy | 44000 |
| 004 | 22 | Jones | Bob | 55000 |

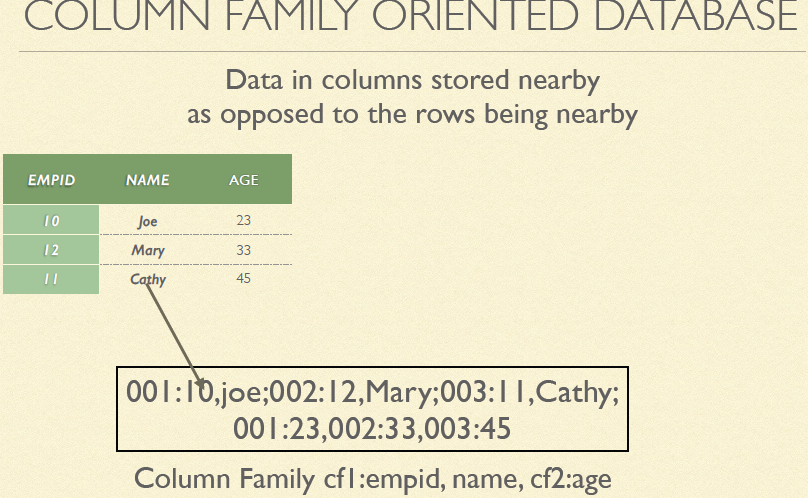
A column-oriented database will store the data as follows:

10:001,12:002,11:003,22:004;Smith:001,Jones:002,Johnson:003,Jones:004;Joe:001,Mary:002,Cathy:003,Bob:004;40000:001,

50000:002,44000:003,55000:004;

Column Family Oriented Database:

Data in columns stored nearby as opposed to the rows being nearby



Ex: Accumulo, Cassandra, HBase

1. Document Oriented data store:

A **document-oriented database** is a [computer program](http://en.wikipedia.org/wiki/Computer_program) designed for storing, retrieving, and managing document-oriented information, also known as [semi-structured data](http://en.wikipedia.org/wiki/Semi-structured_model). Here an entire document is treated as a record.  While these can accommodate completely unstructured text, they excel at semi-structured text.  That is text that has been encoded according to a known schema such as XML, YAML, JSON, PDF, email, or even MS Office.

Ex: [Lotus Notes](http://en.wikipedia.org/wiki/Lotus_Notes), [Clusterpoint](http://en.wikipedia.org/wiki/Clusterpoint" \o "Clusterpoint), [Apache couchDB](http://en.wikipedia.org/wiki/Apache_CouchDB), [Couchbase](http://en.wikipedia.org/wiki/Couchbase" \o "Couchbase),  [MarkLogic](http://en.wikipedia.org/wiki/MarkLogic), [MongoDB](http://en.wikipedia.org/wiki/MongoDB" \o "MongoDB), [OrientDB](http://en.wikipedia.org/wiki/OrientDB" \o "OrientDB), [Qizx](http://en.wikipedia.org/wiki/Qizx" \o "Qizx)

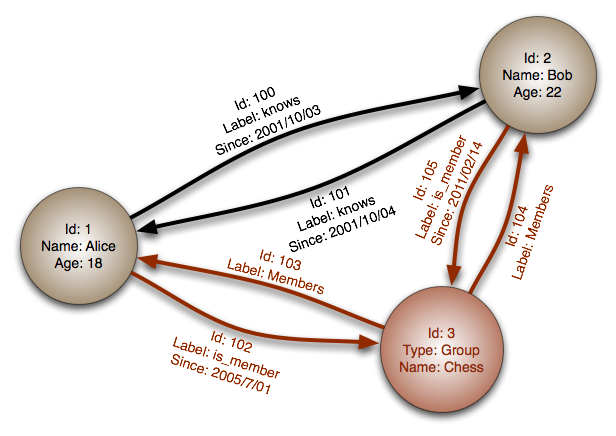
1. Key-Value data Store

Key value stores allow the application developer to store schema-less data. This data is usually consisting of a string which represents the key and the actual data which is considered to be the value in the "key - value" relationship

Ex:  [CouchDB](http://en.wikipedia.org/wiki/CouchDB" \o "CouchDB), [Dynamo](http://en.wikipedia.org/wiki/Dynamo_(storage_system)), [FoundationDB](http://en.wikipedia.org/wiki/FoundationDB" \o "FoundationDB), [MemcacheDB](http://en.wikipedia.org/wiki/MemcacheDB" \o "MemcacheDB), [Redis](http://en.wikipedia.org/wiki/Redis" \o "Redis), [Riak](http://en.wikipedia.org/wiki/Riak" \o "Riak), FairCom [c-treeACE](http://en.wikipedia.org/wiki/C-treeACE), [Aerospike](http://en.wikipedia.org/wiki/Aerospike_database" \o "Aerospike database), [OrientDB](http://en.wikipedia.org/wiki/OrientDB" \o "OrientDB), [MUMPS](http://en.wikipedia.org/wiki/MUMPS)

1. Graph Data Store

In [computing](http://en.wikipedia.org/wiki/Computing), a **graph database** is a [database](http://en.wikipedia.org/wiki/Database) that uses [graph structures](http://en.wikipedia.org/wiki/Graph_(data_structure)) for [semantic queries](http://en.wikipedia.org/wiki/Semantic_Query) with nodes, edges, and properties to represent and store data.



Nodes represent entities such as people, businesses, accounts, or any other item you might want to keep track of.

Properties are pertinent information that relate to nodes. For instance, if "Wikipedia" were one of the nodes, one might have it tied to properties such as "website", "reference material", or "word that starts with the letter 'w'", depending on which aspects of "Wikipedia" are pertinent to the particular database.

Edges are the lines that connect nodes to nodes or nodes to properties and they represent the relationship between the two. Most of the important information is really stored in the edges. Meaningful patterns emerge when one examines the connections and interconnections of nodes, properties, and edges.

**Ex:** Allegro, Neo4J, OrientDB, Virtuoso

