NoSql vs RDBMS: Why NoSql is hot

What's RDBMS and NoSql

1) Relational Database Management System (RDBMS)

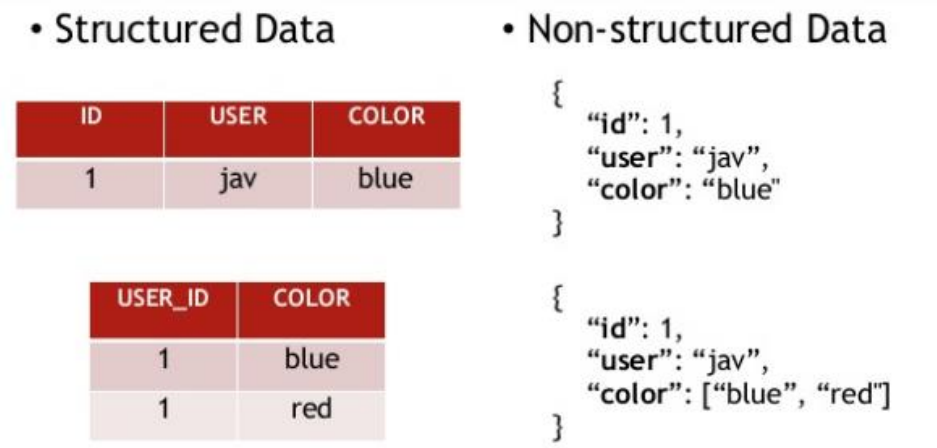
RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields.

According to DB-Engines, in 2016, the most widely used systems are Oracle, MySQL (open source), Microsoft SQL Server, PostgreSQL (open source), IBM DB2, Microsoft Access, and SQLite (open source)

2) NoSql

NoSQL stands for "Not only SQL". It provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

Based on 2015 popularity rankings, the most popular NoSQL databases are MongoDB, Apache Cassandra, and Redis.



Difference between NoSQL and Relational Data Models (RDBMS)

1) Data storage

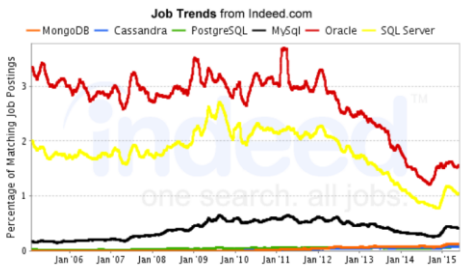
RDBMS is completely structured way of storing data. It takes data and separates it into many interrelated tables that contain rows and columns. Tables reference each other through foreign keys that are stored in columns as well. When looking up data, the desired information needs to be collected from many tables (often hundreds in today's enterprise applications) and combined before it can be provided to the application. Similarly, when writing data, the write needs to be coordinated and performed on many tables.

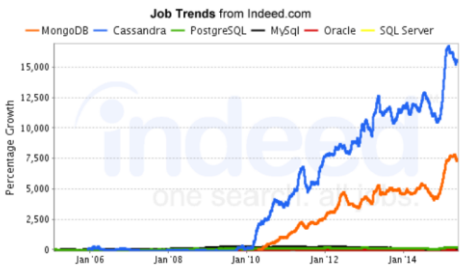
NoSQL is unstructured way of storing the data. For example, a document-oriented NoSQL database takes the data you want to store and aggregates it into documents using the JSON format. Aggregating this information may lead to duplication of information, but since storage is no longer cost prohibitive, the resulting data model flexibility, ease of efficiently distributing the resulting documents and read and write performance improvements make it an easy trade-off for web-based applications.

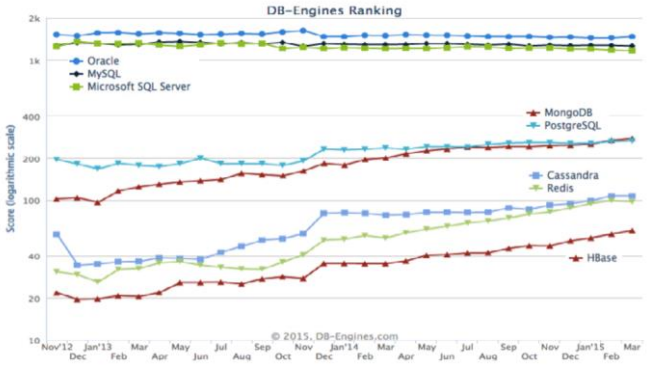
2) Schema

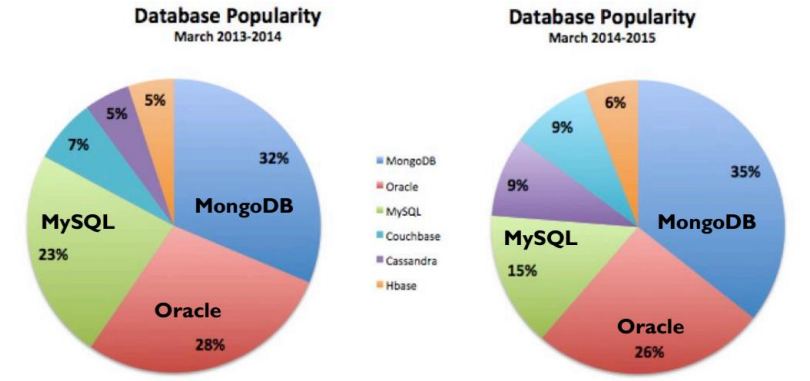
The second difference is schema. **RDMBS** models have rigid schemas while NoSQL models are schemaless. To RDMBS, when you define the database models, you need to write a relation schema for each table. In this word, changing the schema is a big deal if data is inserted. So it's inefficient to deal with the big data. While to **NoSql**, records could add new information on the fly. Unlike SQL table rows, dissimilar data could be stored together as well. It may not provide full ACID (atomicity, consistency, isolation, durability) guarantees as well, but still has a distributed and fault tolerant architecture.

NoSql is very hot







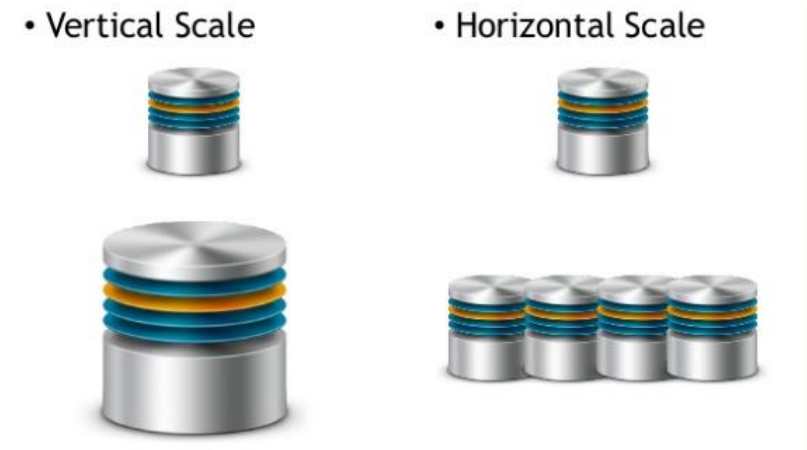


Why it's hot

1) **NoSQL has Flexible Data Model to Capture Unstructured / Semi-structured Big Data**

   Data is becoming easier to capture and access through third parties, like Facebook, D&B, and others. The world is now a big-data world. Data influences the world, and thus influence our surroundings, like communication, shopping, advertising and entertainment. Getting the newest information data is of great importance, so it's not surprising that developers want to enrich existing applications and create new ones made possible by it. If we use RDBMS database model, it's undoubted that the data couldn't be updated quickly due to the limit of the model. NoSQL provides a data model that maps better to these needs.

2) NoSQL is highly and easily scalable (Scale up vs Scale out)



The traditional RDMBS model doesn't fit the situation where a lot of customers use your application. With relational technologies, it's difficult and even impossible to get the dynamic scalability and level of scale they need while also maintaining the performance users demand. You need to switch to NoSQL databases. The main reason is that **RDBMS is a centralized, share-everything technology that scales up rather than out**. This made them have a poor performance on the applications that require easy and dynamic scalability. **NoSQL model is an unstructured, scale-out and distributed technology** and therefore fit better with the highly distributed nature of the three-tier Internet architecture.

At the web/application tier of the three-tier Internet architecture, a scale out approach has been the default for many years and worked extremely well. As more people use an application, more commodity servers should be added to the web/application tier to maintain the performance. For the RDBMS model, the default scaling approach at the database tier was to scale up. To support more concurrent users and/or store more data, you need a bigger server with more CPUs, more memory, and more disk storage to keep all the tables, which means that the servers tend to be highly complex, proprietary, and expensive, in comparison with the low-cost, commodity hardware typically used so effectively at the web/application server tier. With the NoSql model, this issue could be solved since it could scale out.

3) Dyamic schemas

For **relational databases**, we need to define schemas before you can add data. The SQL database needs to know what you have to store in advance, which is fatal for a flexible dataset. Every time we get some new features, we have to change the schema of the dataset.

NoSQL databases, on contrast, 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.

4) Auto-sharding

Relational databases usually scale up vertically because of its structured way of storing data. The single server need to hold the entire dataset to ensure reliability and continuous availability, which means that the cost is very expensive and place limits on scale.

While for NoSql model, data could scale out horizontally. By adding servers instead of concentrating more capacity in a single server, the dataset could be divided into pieces and spread data across multiple servers. Besides, NoSql usually support auto-sharding, meaning that they can realize it automatically. Data and query load are automatically balanced across servers, so if a server goes down, it can be quickly and transparently replaced with no application disruption.

5) Replication

Most NoSQL databases support automatic replication as well, which means that you get high availability and disaster recovery.

NoSQL Database Types

1) Document-based stores

Document databases pair each key with a complex data structure as collections of documents. rather than as structured tables with uniform sized fields for each record. Documents can

contain many different key-value pairs, or key-array pairs, or even nested documents. With these databases, users can add any number of fields of any length to a document.

2) Graph stores

Graph stores are used to store information about networks, such as social connections. Graph stores include Neo4J and HyperGraphDB.

3) Key-value stores

Every single item in the database is stored as an attribute name (or "key"), together with its value. A key-value store is a system that stores values indexed for retrieval by keys. These systems can hold structured or unstructured data. 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) Wide-Column database

Instead of storing all the information in a heavily structured table of columns and rows, as is the case with relational databases, wide-column databases have an extendable column of closely related data. For example, in the HBase database, all data that relates to something is stored on a single row. Columns are as wide as you like and they are designed for queries over wider sets of data. So, the join is almost unnecessary as the data is already pre-joined and stored together.

Reference:

[1] http://theprofessionalspoint.blogspot.in/2014/01/nosql-vs-rdbms-why-and-why-not-to-use.html

[2] https://en.wikipedia.org/wiki/NoSQL

[3] https://en.wikipedia.org/wiki/Relational_database_management_system

[4] https://www.asee.org/documents/zones/zone3/2015/Comparisons-of-Relational-Databases-with-Big-Data-a-Teaching-Approach.pdf

[5] https://www.datastax.com/relational-database-to-nosql

[6] http://www.jamesserra.com/archive/2015/08/relational-databases-vs-non-relational-databases/