e-PGPathshala
Subject : Computer Science
Paper: Data Analytics
Module No 31: CS/DA/31 -NoSQL
Quadrant 1 – e-text

1.1 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.

1.2 Learning Objectives

- To make good data design decisions by understanding the differences between relational and Non-Relational systems.
- To know about different type of NoSQL databases.
- To compare NoSQL databases with each other and relational systems

1.3 Classical relation database follow the ACID Rules

A database transaction must be atomic, consistent, isolated and durable. Below we have discussed these four points.

Atomic: A transaction is a logical unit of work which must be either completed with all of its data modifications, or none of them is performed.

Consistent: At the end of the transaction, all data must be left in a consistent state.

Isolated: Modifications of data performed by a transaction must be independent of another transaction. Unless this happens, the outcome of a transaction may be erroneous.

Durable: When the transaction is completed, effects of the modifications performed by the transaction must be permanent in the system.

Often these four properties of a transaction are acronymed as **ACID**.

1.4 Distributed Systems

A distributed system consists of multiple computers and software components that communicate through a computer network (a local network or by a wide area network). A distributed system can consist of any number of possible configurations, such as mainframes, workstations, personal computers, and so on. The computers interact with each other and share the resources of the system to achieve a common goal.

Advantages of Distributed Computing

Reliability (fault tolerance):
The important advantage of distributed computing system is reliability. If some of the machines within the system crash, the rest of the computers remain unaffected and work does not stop.

Scalability:
In distributed computing the system can easily be expanded by adding more machines as needed.

Sharing of Resources:
Shared data is essential for many applications, such as banking, reservation system etc. As data or resources are shared in distributed system, other resources can be also shared (e.g. expensive printers).

Flexibility:
As the system is very flexible, it is very easy to install, implement and debug new services.

Speed:
A distributed computing system can have more computing power and it's speed makes it different than other systems.

Open system:
As it is open system, every service is equally accessible to every client i.e. local or remote.

Performance:
The collection of processors in the system can provide higher performance (and better price/performance ratio) than a centralized computer.

Disadvantages of Distributed Computing

Troubleshooting:
Troubleshooting and diagnosing problems**.**

Software:
Less software support is the main disadvantage of distributed computing system.

Networking:
The network infrastructure can create several problems such as transmission problem, overloading, loss of messages.

Security:
Easy access in distributed computing system increases the risk of security and sharing of data generates the problem of data security

Scalability

In electronics (including hardware, communication and software), scalability is the ability of a system to expand to meet your business needs. For example scaling a web application is all about allowing more people to use your application. You scale a system by upgrading the existing hardware without changing much of the application or by adding extra hardware. There are two ways of scaling horizontal and vertical scaling:

Vertical scaling

To scale vertically (or scale up) means to add resources within the same logical unit to increase capacity. For example to add CPUs to an existing server, increase memory in the system or expanding storage by adding hard drive.

Horizontal scaling

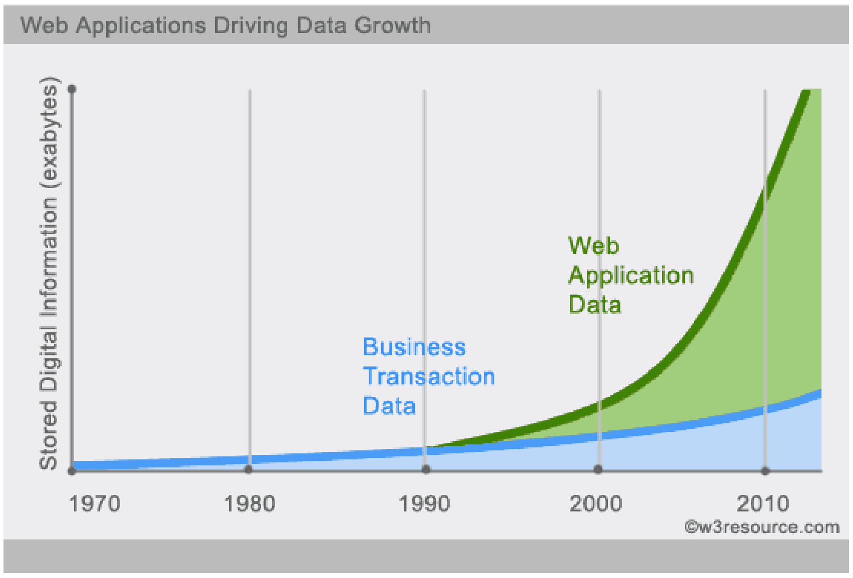
To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. In NoSQL system, data store can be much faster as it takes advantage of "scaling out" which means to add more nodes to a system and distribute the load over those nodes.

1.5 What is NoSQL?

NoSQL is a non-relational database management system, different from traditional relational database management systems in some significant ways. It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users). These types of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

Why NoSQL?

In today's time, data is becoming easier to access and capture through third parties such as Facebook, Google+ and others. Personal user information, social graphs, geo location data, user-generated content and machine logging data are just a few examples where the data has been increasing exponentially. To avail the above service properly, it is required to process huge amount of data. The evolution of NoSql databases is to handle these huge data properly.



Example:

Social-network graph:

Each record: UserID1, UserID2
Separate records: UserID, first_name,last_name, age, gender,...
Task: Find all friends of friends of friends of ... friends of a given user.

Wikipedia pages:

Large collection of documents
Combination of structured and unstructured data
Task: Retrieve all pages regarding athletics of Summer Olympic before 1950.

RDBMS vs 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

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
- BASE Transaction

1.6 Brief history of NoSQL

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, and 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.

1.7 CAP Theorem (Brewer's Theorem)

You must understand the CAP theorem when you talk about NoSQL databases or in fact when designing any distributed system. CAP theorem states that there are three basic requirements which exist in a special relation when designing applications for a distributed architecture.

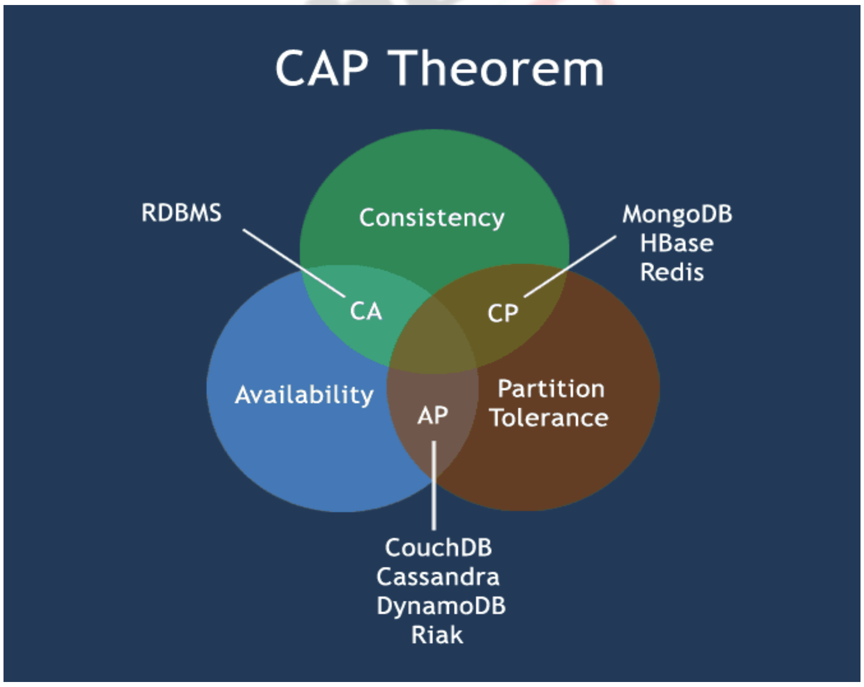
Consistency - The data in the database remains consistent after the execution of an operation. For example after an update operation all clients see the same data.

Availability - The system is always on (service guarantee availability), no downtime.

Partition Tolerance - The system continues to function, even the communication among the servers is unreliable. That is the servers may be partitioned into multiple groups that cannot communicate with one another.

In theoretically it is impossible to fulfil all 3 requirements. CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements. Therefore the entire NoSQL database follow the different combinations of the C, A, P from the CAP theorem. Here is the brief description of three combinations CA, CP, AP:

CA - Single site cluster, therefore all nodes are always in contact. When partitioning occurs, the system blocks.
CP - Some data may not be accessible, but the rest is still consistent /accurate.
AP - System is still available under partitioning, but some of the data returned may be inaccurate.



NoSQL pros/cons

Advantages:

- High scalability
- Distributed Computing
- Lower cost
- Schema flexibility, semi-structure data
- No complicated Relationships

Disadvantages

- No standardization
- Limited query capabilities (so far)
- Eventual consistent is not intuitive to program for

1.8 The BASE

The BASE acronym was defined by Eric Brewer, who is also known for formulating the CAP theorem.

The CAP theorem states that a distributed computer system cannot guarantee all of the following three properties at the same time:

- Consistency
- Availability
- Partition tolerance

A BASE system gives up on consistency.

- **B**asically **A**vailable indicates that the system *does* guarantee availability, in terms of the CAP theorem.
- **S**oft state indicates that the state of the system may change over time, even without input. This is because of the eventual consistency model.
- **E**ventual consistency indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

ACID vs BASE

ACID	**BASE**
Atomic	**B**asically **A**vailable
Consistency	**S**oft state
Isolation	**E**ventual consistency
Durable	

1.9 NoSQL Categories

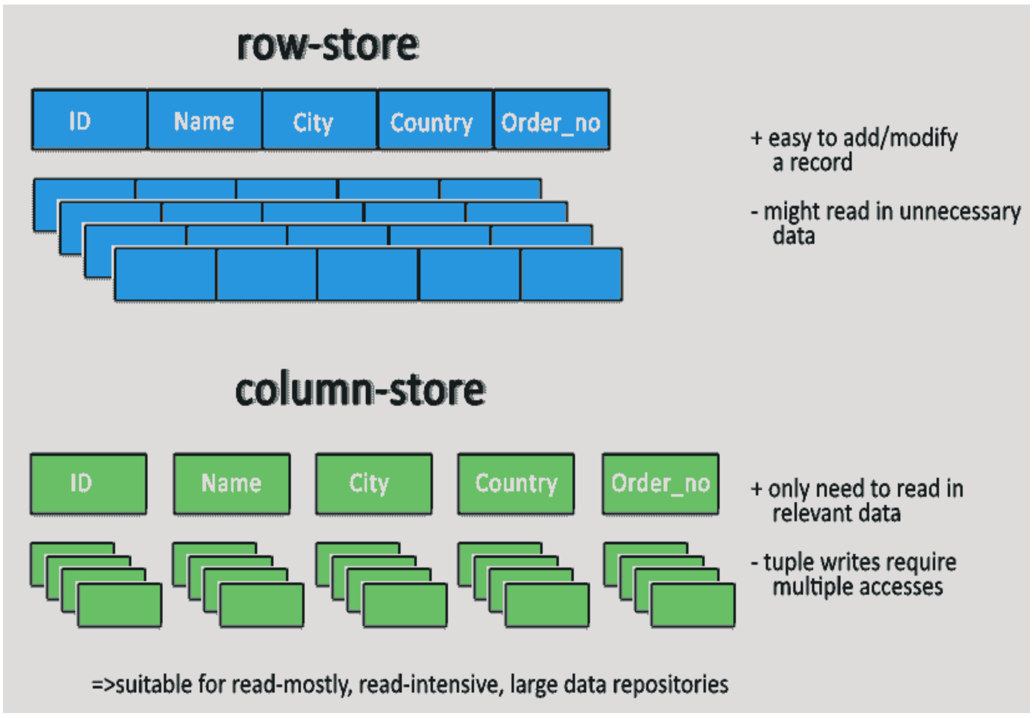
There are four general types (most common categories) of NoSQL databases. Each of these categories has its own specific attributes and limitations. There is not a single solution which is better than all the others; however there are some databases that are better to solve specific problems. To clarify the NoSQL databases, let's discuss the most common categories:

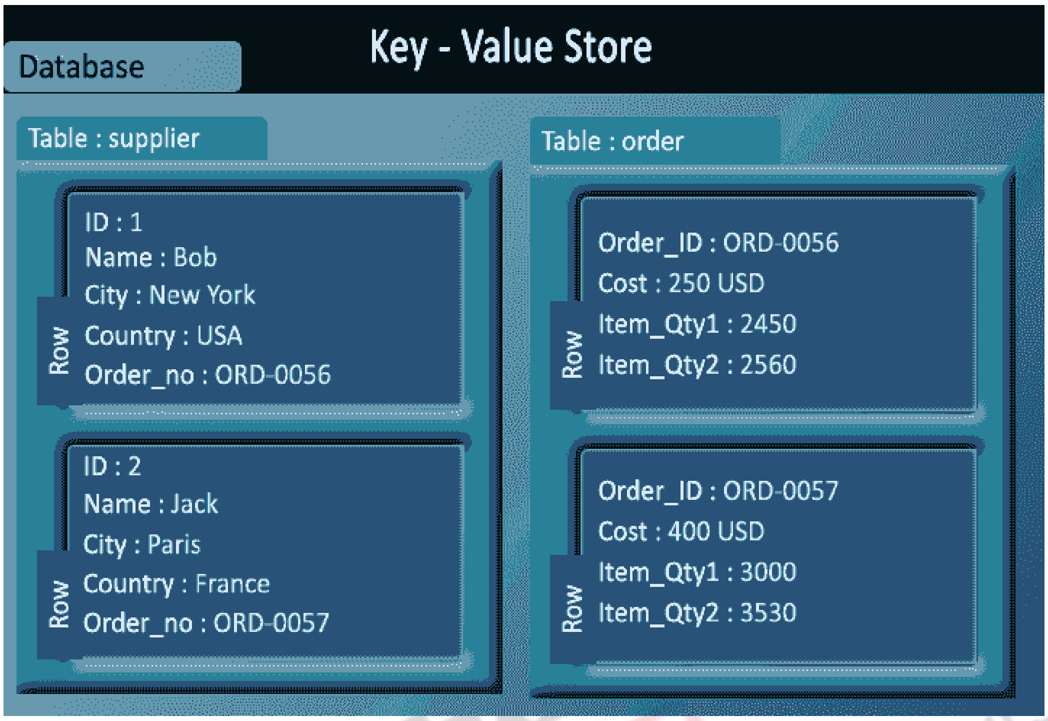
- Key-value stores
- Column-oriented
- Graph
- Document oriented

1.9.1 Key-value stores

- Key-value stores are most basic types of NoSQL databases.
- Designed to handle huge amounts of data.
- Based on Amazon's Dynamo paper.
- Key value stores allow developer to store schema-less data.
- In the key-value storage, database stores data as hash table where each key is unique and the value can be string, JSON, BLOB (Binary Large OBjec) etc.
- A key may be strings, hashes, lists, sets; sorted sets and values are stored against these keys.
- For example a key-value pair might consist of a key like "Name" that is associated with a value like "Robin".
- Key-Value stores can be used as collections, dictionaries, associative arrays etc.
- Key-Value stores follow the 'Availability' and 'Partition' aspects of CAP theorem.
- Key-Values stores would work well for shopping cart contents, or individual values like colour schemes, a landing page URI, or a default account number.

Pictorial Presentation:

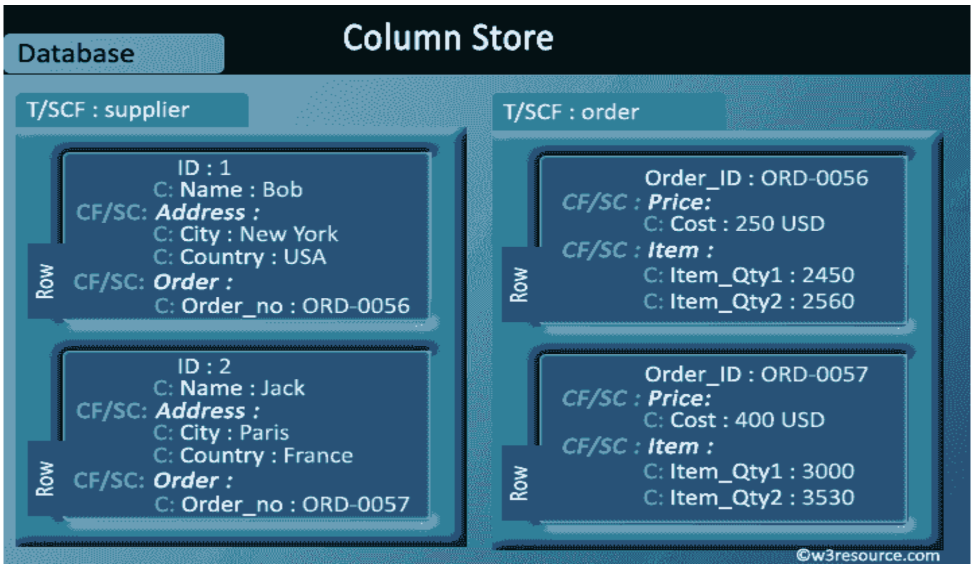

row-store
ID
Name
City
Country
Order_no
+ easy to add/modify a record
- might read in unnecessary data
column-store
ID
Name
City
Country
Order_no
+ only need to read in relevant data
- tuple writes require multiple accesses
=>suitable for read-mostly, read-intensive, large data repositories



1.9.2 Column-oriented databases

- Column-oriented databases primarily work on columns and every column is treated individually.
- Values of a single column are stored contiguously.
- Column stores data in column specific files.
- In Column stores, query processors work on columns too.
- All data within each column data file have the same type which makes it ideal for compression.

Pictorial Presentation:

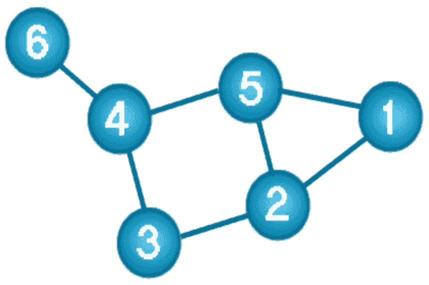


- Column stores can improve the performance of queries as it can access specific column data.
- High performance on aggregation queries (e.g. COUNT, SUM, AVG, MIN, MAX).
- Works on data warehouses and business intelligence, customer relationship management (CRM), Library card catalogs etc.

1.9.3 Graph databases

A graph data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices.

The following picture presents a labelled graph of 6 vertices and 7 edges.



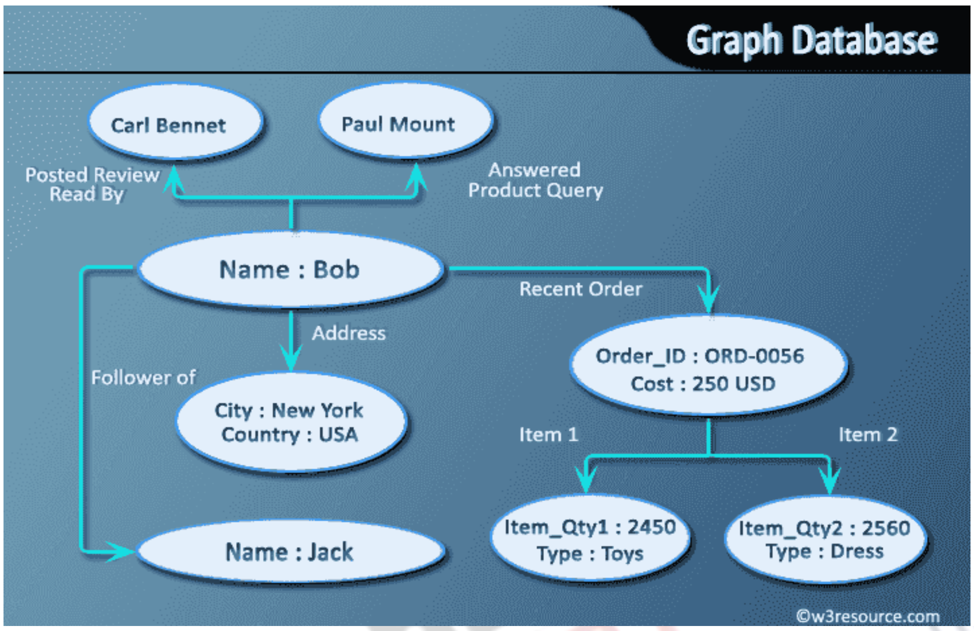
What is a Graph Databases?

- A graph database stores data in a graph.
- It is capable of elegantly representing any kind of data in a highly accessible way.
- A graph database is a collection of nodes and edges
- Each node represents an entity (such as a student or business) and each edge represents a connection or relationship between two nodes.
- Every node and edge is defined by a unique identifier.
- Each node knows its adjacent nodes.
- As the number of nodes increases, the cost of a local step (or hop) remains the same.
- Index for lookups.

Here is a comparison between the classic relational model and the graph model:

Relational model	**Graph model**
Tables	Vertices and Edges set
Rows	Vertices
Columns	Key/value pairs
Joins	Edges

Pictorial Presentation:



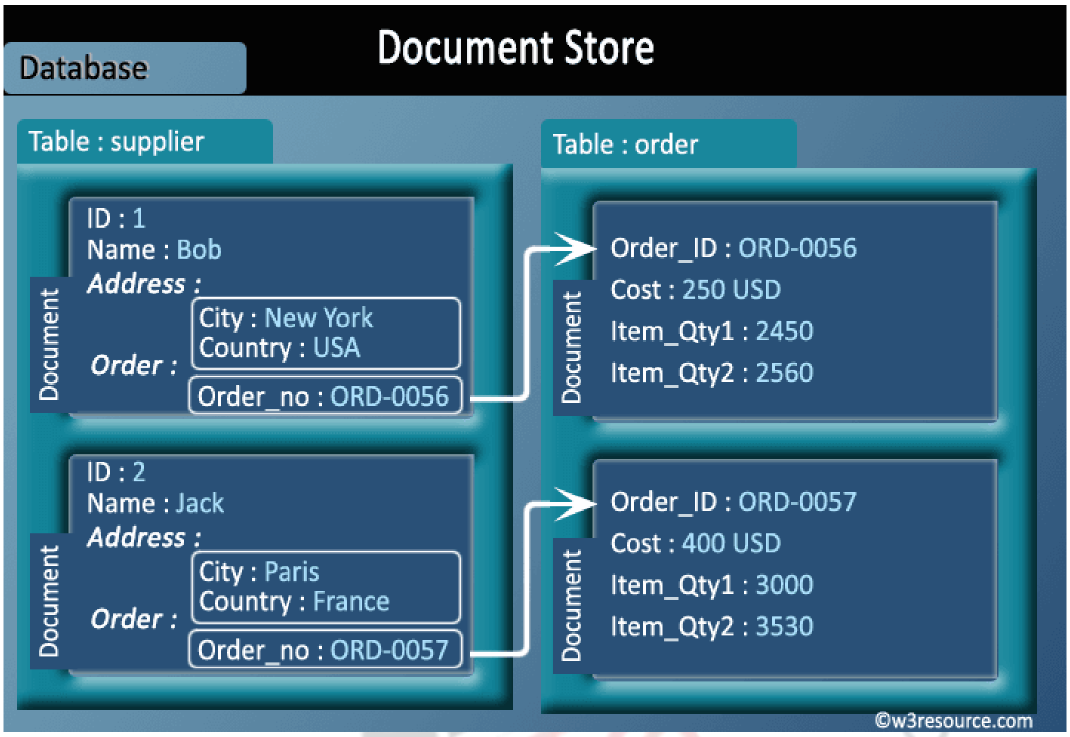
1.9.4 Document oriented databases

- A collection of documents
- Data in this model is stored inside documents.
- A document is a key value collection where the key allows access to its value.
- Documents are not typically forced to have a schema and therefore are flexible and easy to change.
- Documents are stored into collections in order to group different kinds of data.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

Here is a comparison between the classic relational model and the document model:

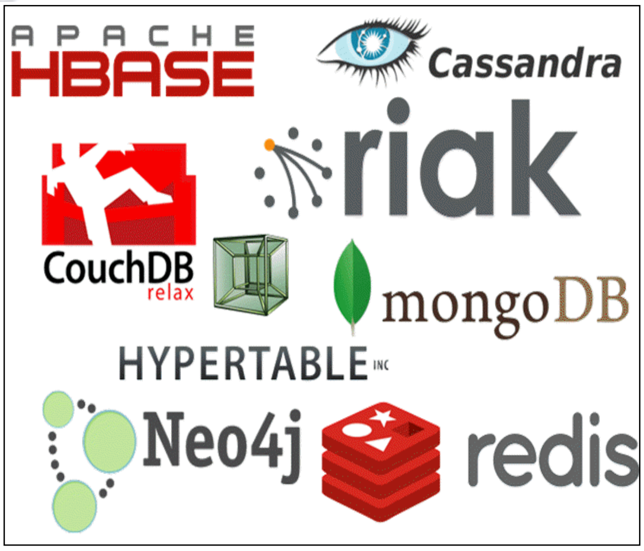
Relational model	**Document model**
Tables	Collections
Rows	Documents
Columns	Key/value pairs
Joins	not available

Pictorial Presentation:



To summarize some of the popular NoSQL databases that falls in the above categories respectively.

- **Key Value Databases -** Membase, Redis, MemcacheDB, etc.
- **Column Based Databases** – CouchDB, OrientDB, etc.
- **Graph Based Databases** – Neo4j, OrientDB, Facebook Open Graph, FlockDB, etc.
- **Document Oriented Databases** – MongoDB, HBase, Cassandra, Amazon SimpleDB, Hypertable, etc.



1.10 Production deployment

There are a large number of companies using NoSQL. To name a few:

- Google
- Facebook
- Mozilla
- Adobe
- Foursquare
- LinkedIn
- Digg
- McGraw-Hill Education
- Vermont Public Radio

Many enterprises have successfully introduced NoSQL by identifying a single application or service to start with. Some common examples of good fits for NoSQL:

- ➢ Product Catalog Service
- ➢ Asset Tracking Service
- ➢ Content Management Service
- ➢ Application Configuration Service
- ➢ Customer Management Service
- ➢ File or Streaming Metadata Service

SUMMARY

- NoSQL is the successful database system to handle huge amount of data.
- A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations.
- The data structure differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS.