Introduction to NoSQL

NoSQL – A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

NoSQL databases are typically **open source** software. The software is free, and most of them are free to use in commercial products.
The **open source** codebases can be modified to solve business needs.

NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages.

NoSQL is generally 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 type of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

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. Which SQL databases were never designed. The evolution of NoSql databases is to handle these huge data properly.

NoSQL is

1. Stands for Not Only SQL
2. document based model which is non-relational in nature.
3. Schema free document storage.
4. Support Create, Delete, Update and Read
5. Supports indexing and querying
6. Supports concurrency and transactions
7. Highly optimized for retrieve and append
8. Great performance and scalability.
9. Open Source

RDBMS is completely structured way of storing data. While the **NoSQL** is unstructured way of storing the data. And another main **difference** is that the amount of data stored mainly depends on the Physical memory of the system. While in the **NoSQL** you don't have any such limits as you can scale the system horizontally.

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

NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot-persistent architectures.

NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data

NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages.

A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve. Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables.

Many NoSQL stores compromise consistency in favor of availability, speed and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases. Most NoSQL stores lack true ACID(Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made

Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss. For distributed transaction processing across multiple databases, data consistency is an even bigger challenge. This is difficult for both NoSQL and relational databases. Even current relational databases do not allow referential integrity constraints to span databases. There are few systems that maintain both X/Open XA standards and ACID transactions for distributed transaction processing.

Advantages of NoSQL : There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

Advantages

1. **High scalability –**
2. NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.

Disadvantages

1. **Narrow focus –** NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
2. **Open-source –** NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.
3. **Management challenge –**The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.

Disadvantages (contd…)

4. **GUI is not available –** GUI mode tools to access the database is not flexibly available in the market.
5. **Backup –** Backup is a great weak point for some NoSQL databases. There is no approach for the backup of data in a consistent manner.
6. **Large document size –** Some database systems like MongoDB and CouchDB store data in JSON (JavaScript Object Notation) format. Which means that documents are quite large, and having descriptive key names actually hurts, since they increase the document size.

Types of NoSQL database:

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 solutions which is better than all the others, however there are some databases that are better to solve specific problems. To clarify the NoSQL databases, lets discuss the most common categories :

1. Key-value stores
2. Column-oriented
3. Graph
4. Document oriented

Key Value Stores

- Key-value stores are most basic types of NoSQL databases.
- Designed to handle huge amounts of data.
- 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 color schemes, a landing page URI, or a default account number.

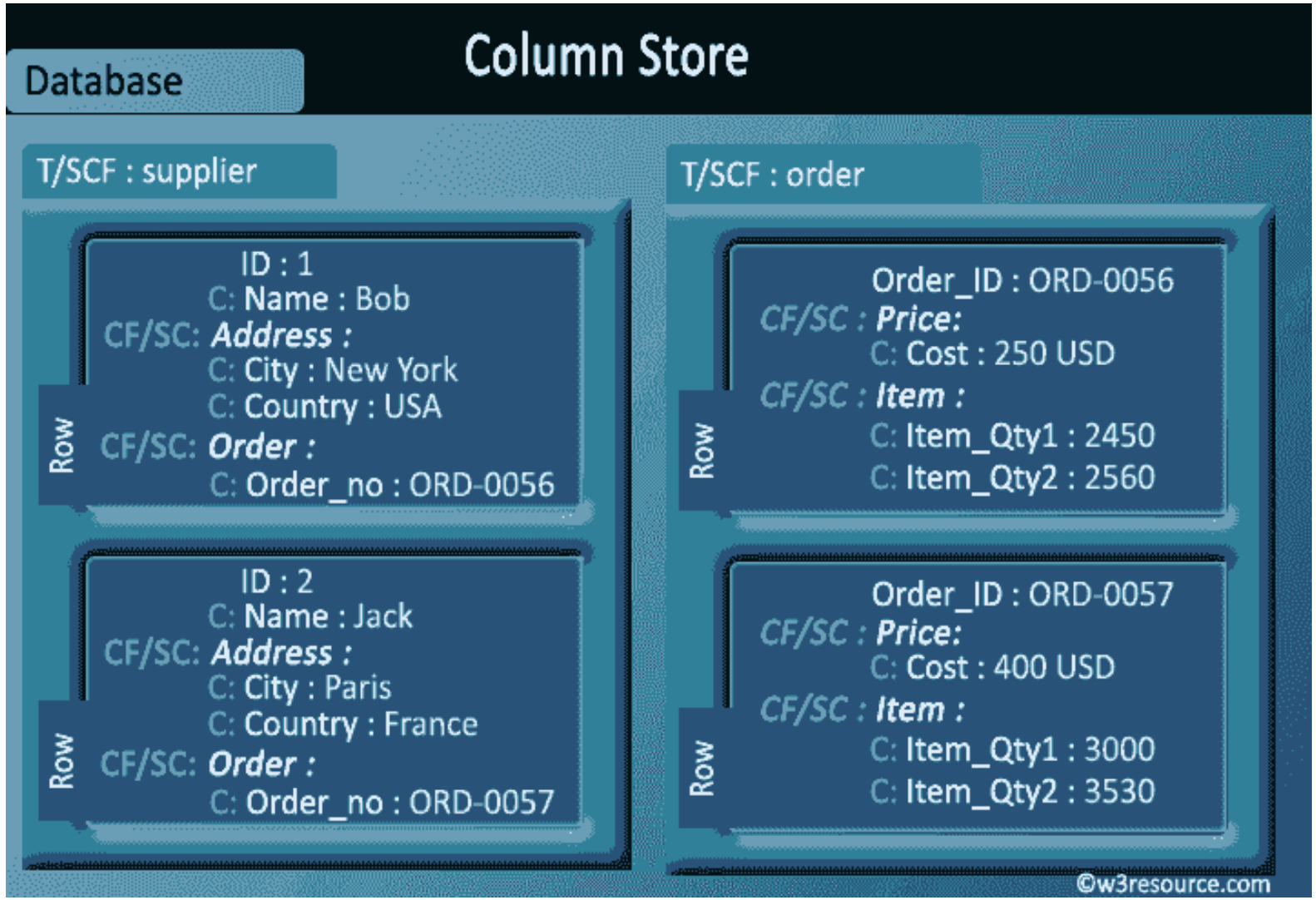
Example of Key-value store DataBase : Redis, Dynamo, Riak. etc.


Key - Value Store
Database
Table : supplier
ID : 1
Name : Bob
City : New York
Country : USA
Order_no : ORD-0056
Row
ID : 2
Name : Jack
City : Paris
Country : France
Order_no : ORD-0057
Row
Table : order
Order_ID : ORD-0056
Cost : 250 USD
Item_Qty1 : 2450
Item_Qty2 : 2560
Row
Order_ID : ORD-0057
Cost : 400 USD
Item_Qty1 : 3000
Item_Qty2 : 3530
Row

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 datafile have the same type which makes it ideal for compression.
- 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.

Example of Column-oriented databases : BigTable, Cassandra, SimpleDB etc.


Column Store
Database
T/SCF : supplier
Row
ID : 1
C: Name : Bob
CF/SC: Address :
C: City : New York
C: Country : USA
CF/SC: Order :
C: Order_no : ORD-0056
Row
ID : 2
C: Name : Jack
CF/SC: Address :
C: City : Paris
C: Country : France
CF/SC: Order :
C: Order_no : ORD-0057
T/SCF : order
Row
Order_ID : ORD-0056
CF/SC : Price:
C: Cost : 250 USD
CF/SC : Item :
C: Item_Qty1 : 2450
C: Item_Qty2 : 2560
Row
Order_ID : ORD-0057
CF/SC : Price:
C: Cost : 400 USD
CF/SC : Item :
C: Item_Qty1 : 3000
C: Item_Qty2 : 3530
©w3resource.com

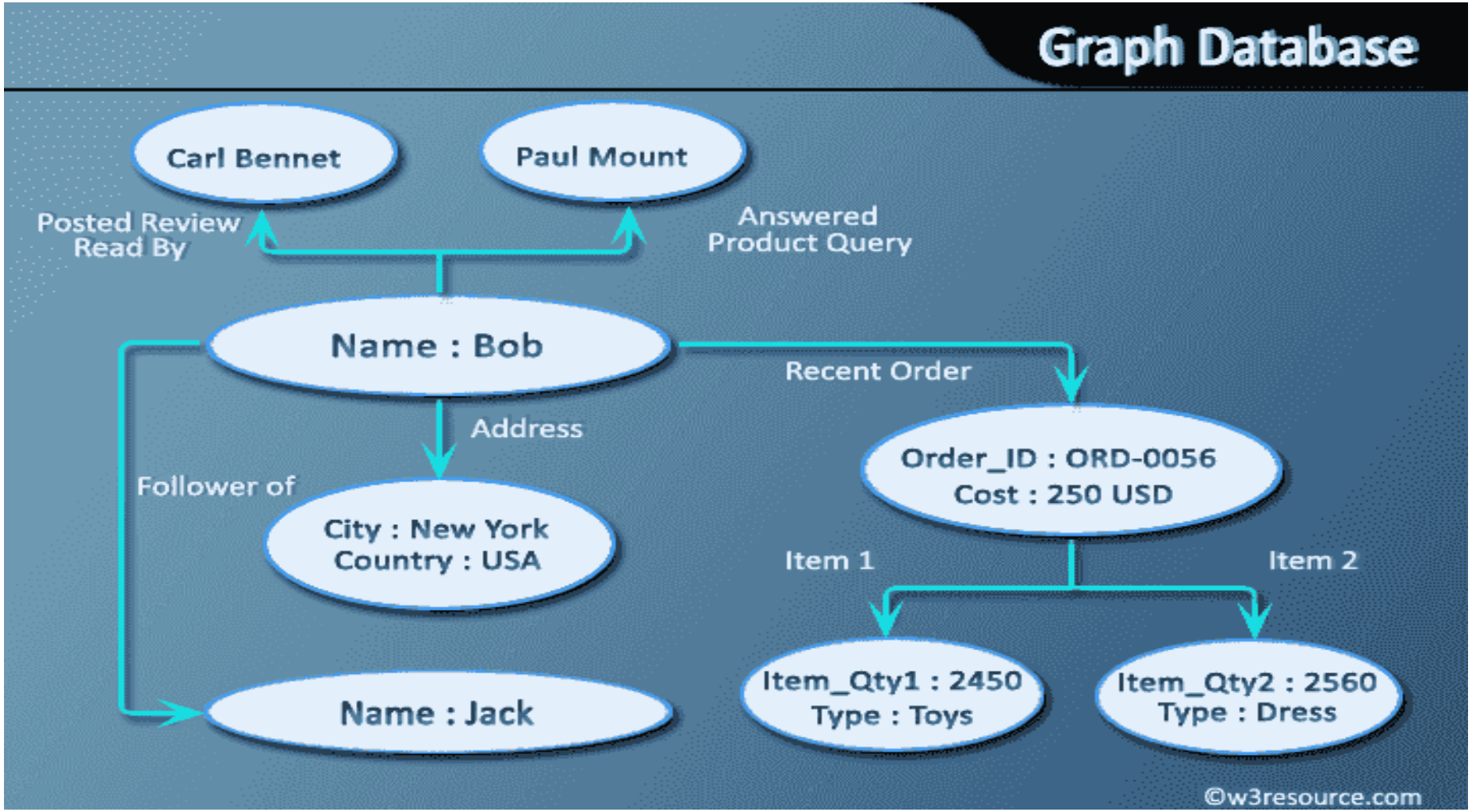
Graph databases

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 labeled 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 are 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.

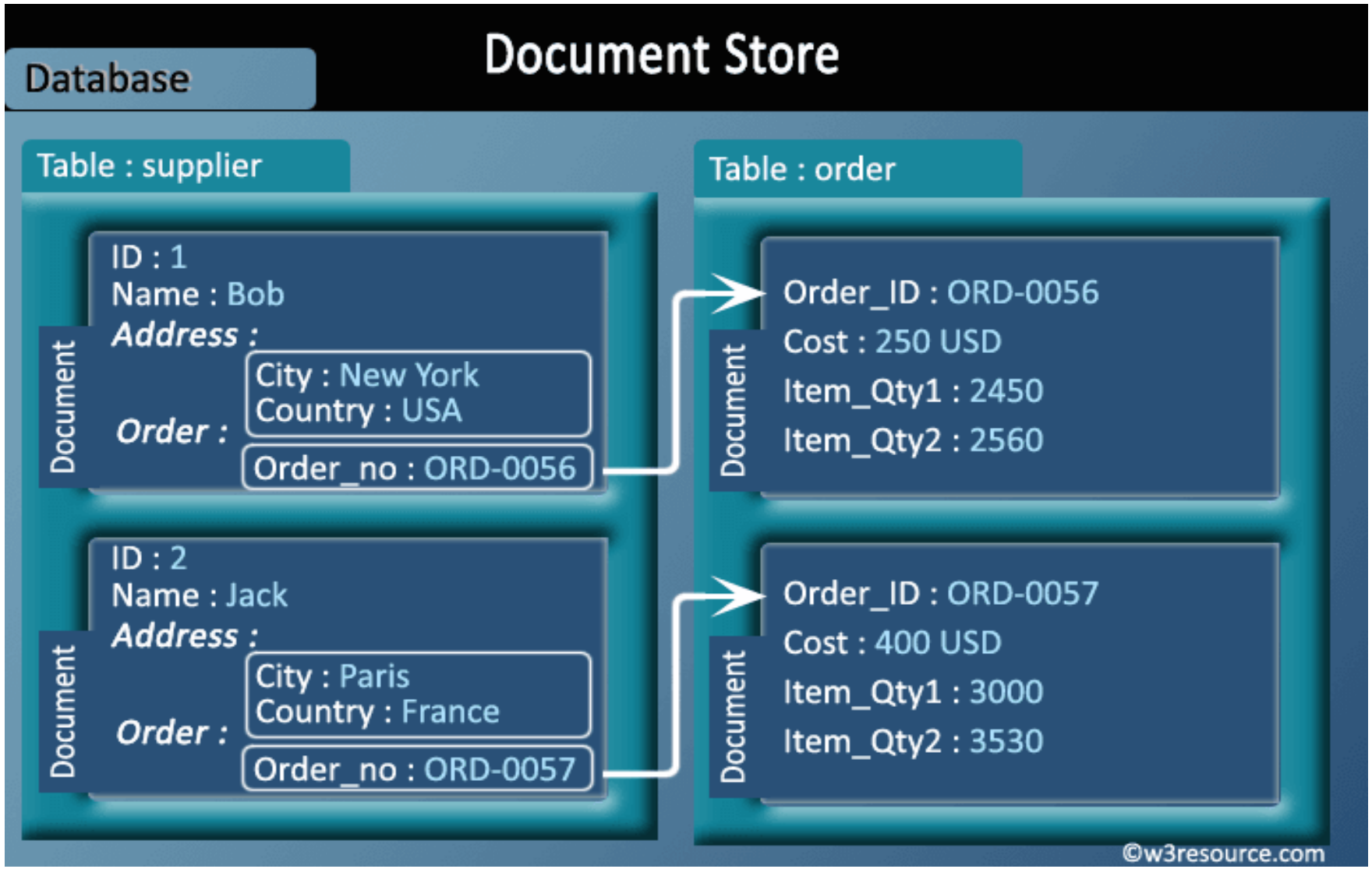
Example of Graph databases : OrientDB, Neo4J, Titan.etc.


Graph Database
Carl Bennet
Paul Mount
Posted Review
Read By
Answered
Product Query
Name : Bob
Recent Order
Address
Follower of
City : New York
Country : USA
Order_ID : ORD-0056
Cost : 250 USD
Item 1
Item 2
Name : Jack
Item_Qty1 : 2450
Type : Toys
Item_Qty2 : 2560
Type : Dress
©w3resource.com

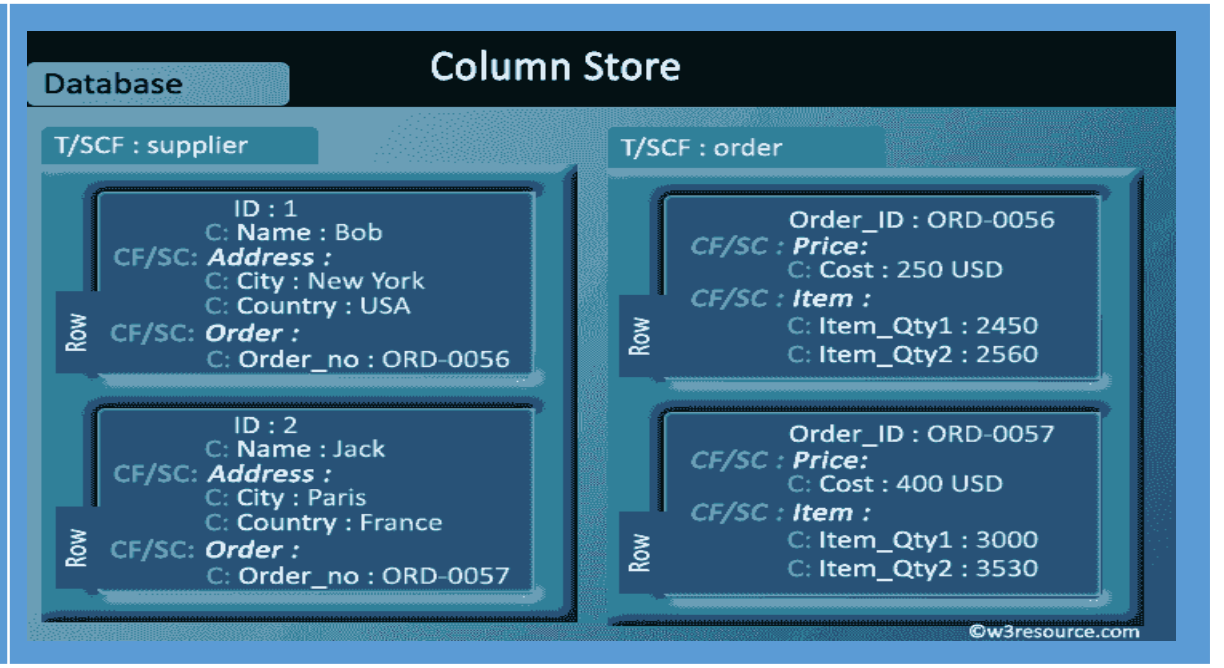
Document Oriented databases

- 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.

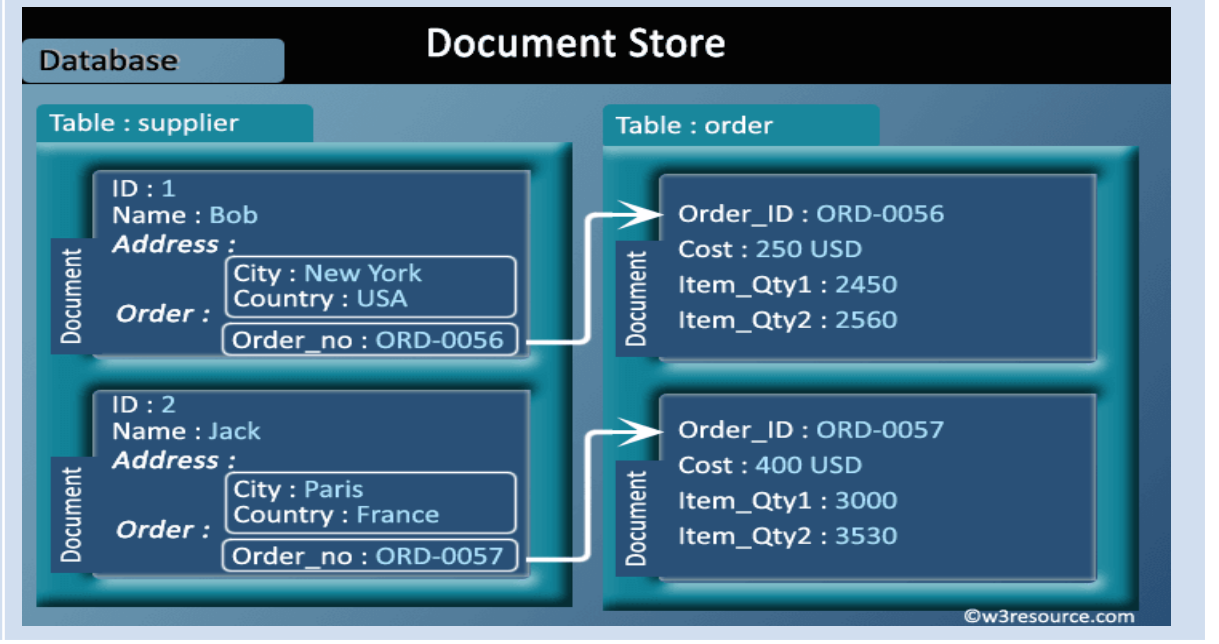
Example of Document Oriented databases : MongoDB, CouchDB etc.


Document Store
Database
Table : supplier
Table : order
Document
ID : 1
Name : Bob
Address :
City : New York
Country : USA
Order :
Order_no : ORD-0056
Document
ID : 2
Name : Jack
Address :
City : Paris
Country : France
Order :
Order_no : ORD-0057
Document
Order_ID : ORD-0056
Cost : 250 USD
Item_Qty1 : 2450
Item_Qty2 : 2560
Document
Order_ID : ORD-0057
Cost : 400 USD
Item_Qty1 : 3000
Item_Qty2 : 3530
©w3resource.com


Database
Key - Value Store
Table : supplier
Row
ID : 1
Name : Bob
City : New York
Country : USA
Order_no : ORD-0056
Row
ID : 2
Name : Jack
City : Paris
Country : France
Order_no : ORD-0057
Table : order
Row
Order_ID : ORD-0056
Cost : 250 USD
Item_Qty1 : 2450
Item_Qty2 : 2560
Row
Order_ID : ORD-0057
Cost : 400 USD
Item_Qty1 : 3000
Item_Qty2 : 3530


Database
Column Store
T/SCF : supplier
Row
ID : 1
C: Name : Bob
CF/SC: Address :
C: City : New York
C: Country : USA
CF/SC: Order :
C: Order_no : ORD-0056
Row
ID : 2
C: Name : Jack
CF/SC: Address :
C: City : Paris
C: Country : France
CF/SC: Order :
C: Order_no : ORD-0057
T/SCF : order
Row
Order_ID : ORD-0056
CF/SC : Price:
C: Cost : 250 USD
CF/SC : Item :
C: Item_Qty1 : 2450
C: Item_Qty2 : 2560
Row
Order_ID : ORD-0057
CF/SC : Price:
C: Cost : 400 USD
CF/SC : Item :
C: Item_Qty1 : 3000
C: Item_Qty2 : 3530
©w3resource.com


Database
Column Store
T/SCF : supplier
Row
ID : 1
C: Name : Bob
CF/SC: Address :
C: City : New York
C: Country : USA
CF/SC: Order :
C: Order_no : ORD-0056
Row
ID : 2
C: Name : Jack
CF/SC: Address :
C: City : Paris
C: Country : France
CF/SC: Order :
C: Order_no : ORD-0057
T/SCF : order
Row
Order_ID : ORD-0056
CF/SC : Price:
C: Cost : 250 USD
CF/SC : Item :
C: Item_Qty1 : 2450
C: Item_Qty2 : 2560
Row
Order_ID : ORD-0057
CF/SC : Price:
C: Cost : 400 USD
CF/SC : Item :
C: Item_Qty1 : 3000
C: Item_Qty2 : 3530
©w3resource.com


Database
Document Store
Table : supplier
Document
ID : 1
Name : Bob
Address :
Order :
City : New York
Country : USA
Order_no : ORD-0056
Document
ID : 2
Name : Jack
Address :
Order :
City : Paris
Country : France
Order_no : ORD-0057
Table : order
Document
Order_ID : ORD-0056
Cost : 250 USD
Item_Qty1 : 2450
Item_Qty2 : 2560
Document
Order_ID : ORD-0057
Cost : 400 USD
Item_Qty1 : 3000
Item_Qty2 : 3530
©w3resource.com

When should NoSQL be used:

- When huge amount of data need to be stored and retrieved .
- The relationship between the data you store is not that important
- The data changing over time and is not structured.
- Support of Constraints and Joins is not required at database level
- The data is growing continuously and you need to scale the database regular to handle the data.