

## NOSQL

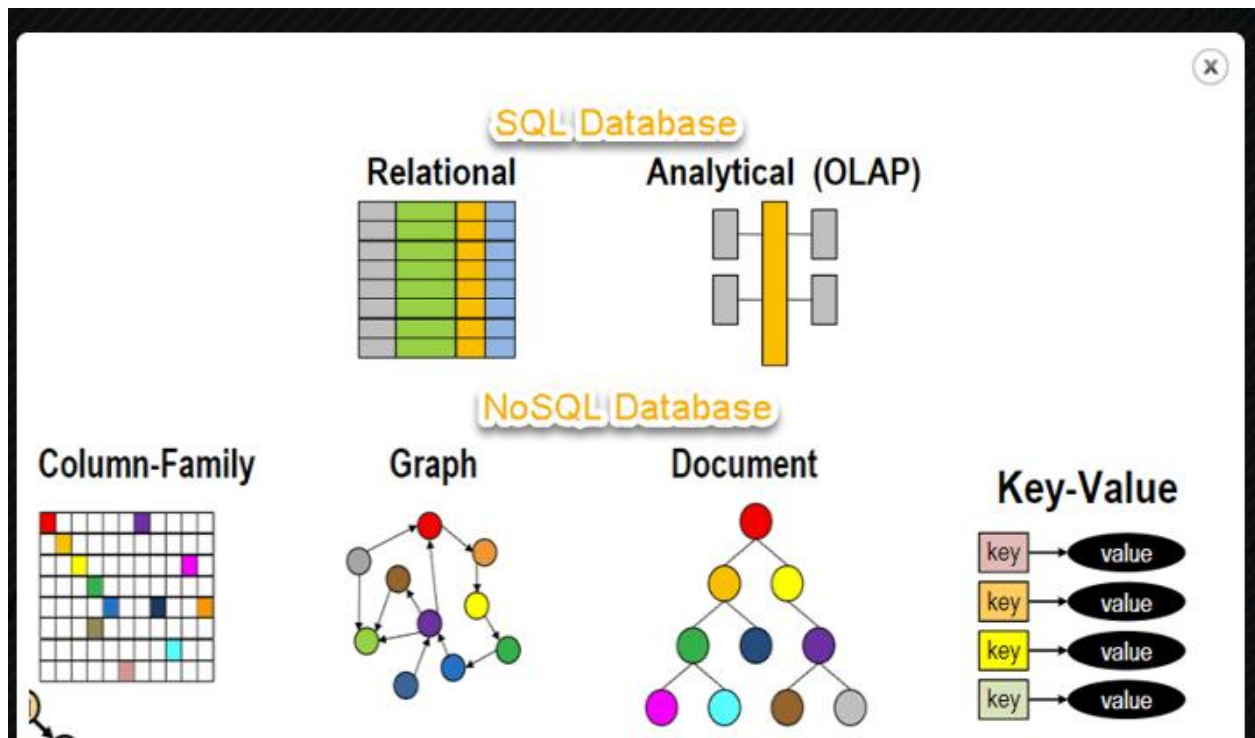
Relational databases require that schemas be defined before you can add data. For example, you might want to store data about your customers such as phone numbers, first and last name, address, city and state – a SQL database needs to know this in advance.

This fits poorly with agile development approaches, because each time you complete new features, the schema of your database often needs to change. So if you decide, a few iterations into development, that you'd like to store customers' favorite items in addition to their addresses and phone numbers, you'll need to add that column to the database, and then migrate the entire database to the new schema.

If the database is large, this is a very slow process that involves significant downtime. If you are frequently changing the data your application stores – because you are iterating rapidly – this downtime may also be frequent. There's also no way, using a relational database, to effectively address data that's completely unstructured or unknown in advance.

NoSQL databases are built to 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, and less database administrator time is needed.

### **NoSQL Database Types**



Since "NoSQL" just means non-relational and not SQL, there are many different ways to implement NoSQL technology. Generally, NoSQL databases include the following families:

- **Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name, or key, together with its value. 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.

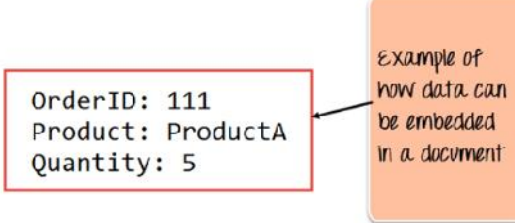
Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

- **Document databases** pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents. eg: MongoDB, Couch DB etc

```

{
  _id : <ObjectId> ,
  CustomerName : Guru99 ,
  Order:
    {
      OrderID: 111
      Product: ProductA
      Quantity: 5
    }
}

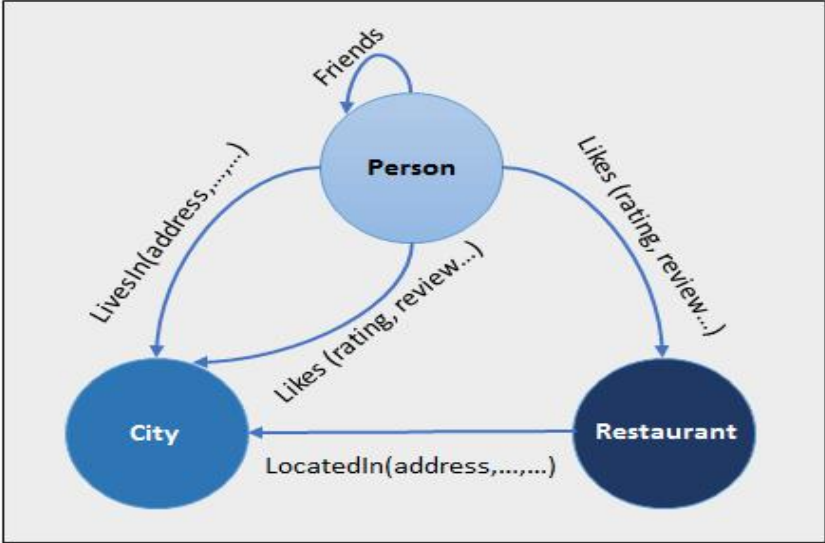
```



- **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

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



## Advantages of NoSQL

- NoSQL is Non-relational

Non-relational, in other words, you can call it as table-less, these NoSQL databases vary from SQL databases. In this sense, they provide the ease of management while ensuring a high level of flexibility with data models that are new.

- NoSQL is Low Cost

While being low cost, NoSQL is also an [open-source database](#), that provides an awesome solution for smaller enterprises to opt this at affordable prices.

The various kinds of NoSQL databases available in the market include Couchbase, Amazon's Dynamo Db, MongoDB and MarkLogic to provide for the processing of big data apps that are cost-effective.

- Scalability is Easier

NoSQL has been gaining popularity because of the elasticity and scalability that it offers over the other kinds of databases that are available. It has been designed to perform exceptionally well under any conditions including low-cost hardware.

Detailed database model structuring is unnecessary here: You can easily create a database without actually developing any detailed database models when using a NoSQL database. This will help to save a lot of your time and effort.

## Disadvantages of NoSQL

- Less Community Support

Though the NoSQL has been expanding at an unbelievable pace, the community support is relatively less as its new.

- Standardization

It lacks a standardized platform like SQL, which is preventing it from further expanding. This has been creating concerns during migration. Standardization is what helps the database industry to unify.

- Interfaces and Interoperability

Interfaces and interoperability is another concern that is faced by NoSQL, which needs fixing immediately.

