Different types of NoSQL Datases.

Types of NoSQL databases:-

There are 4 basic types of NoSQL databases:

1. **Key-value store –** It has a Big Hash Table of keys & value {Example – Riak, Amazon S3 (Dynamo)}
2. **Document-based Store-It** stores documents made up of tagged elements. {Example- CouchDB}
3. **Column-based Store –** Each storage block contains data from only one column, {Example- HBase, Cassandra}
4. **Graph-based –** A network database that uses edges and nodes to represent and store data. {Example-Neo4J}

1. **Key Value Store NoSQL Database**

   The schema-less format of a key value database like Riak is just about what you need for your storage needs. The key can be synthetic or auto-generated while the value can be String, JSON, BLOB (basic large object) etc.

   The key value type basically, uses a hash table in which there exists a unique key and a pointer to a particular item of data. A bucket is a logical group of keys – but they don't physically group the data. There can be identical keys in different buckets.

   Performance is enhanced to a great degree because of the cache mechanisms that accompany the mappings. To read a value you need to know both the key and the bucket because the real key is a hash (Bucket+ Key).

   There is no complexity around the Key Value Store database model as it can be implemented in a breeze. Not an ideal method if you are only looking to just update part of a value or query the database.

   When we try and reflect back on the CAP theorem, it becomes quite clear that key value stores are great around the Availability and Partition aspects but definitely lack in Consistency.

   Example: Consider the data subset represented in the following table. Here the key is the name of the 3Pillar country name, while the value is a list of addresses of 3PiIllar centers in that country.

Key	**Value**
"India"	{"B-25, Sector-58, Noida, India – 201301"
"Romania"	{"IMPS Moara Business Center, Buftea No. 1, Cluj-Napoca, 400606″,City Business Center, Coriolan Brediceanu No. 10, Building B, Timisoara, 300011"}
"US"	{"3975 Fair Ridge Drive. Suite 200 South, Fairfax, VA 22033"}

The key can be synthetic or auto-generated while the value can be String, JSON, BLOB (basic large object) etc.

This key/value type database allow clients to read and write values using a key as follows:

- Get(key), returns the value associated with the provided key.
- Put(key, value), associates the value with the key.
- Multi-get(key1, key2, .., keyN), returns the list of values associated with the list of keys.

- Delete(key), removes the entry for the key from the data store.

While Key/value type database seems helpful in some cases, but it has some weaknesses as well. One, is that the model will not provide any kind of traditional database capabilities (such as atomicity of transactions, or consistency when multiple transactions are executed simultaneously). Such capabilities must be provided by the application itself.

Secondly, as the volume of data increases, maintaining unique values as keys may become more difficult; addressing this issue requires the introduction of some complexity in generating character strings that will remain unique among an extremely large set of keys.

- Riak and Amazon's Dynamo are the most popular key-value store NoSQL databases.

2. **Document Store NoSQL Database**

The data which is a collection of key value pairs is compressed as a document store quite similar to a key-value store, but the only difference is that the values stored (referred to as "documents") provide some structure and encoding of the managed data. XML, JSON (Java Script Object Notation), BSON (which is a binary encoding of JSON objects) are some common standard encodings.

The following example shows data values collected as a "document" representing the names of specific retail stores. Note that while the three examples all represent locations, the representative models are different.

```
{officeName:"3Pillar Noida",
{Street: "B-25, City:"Noida", State:"UP", Pincode:"201301"}
}
{officeName:"3Pillar Timisoara",
{Boulevard:"Coriolan Brediceanu No. 10", Block:"B, Ist Floor", City: "Timisoara",
Pincode: 300011"}
}
{officeName:"3Pillar Cluj",
{Latitude:"40.748328", Longitude:"-73.985560"}
}
```

One key difference between a key-value store and a document store is that the latter embeds attribute metadata associated with stored content, which essentially provides a way to query the data based on the contents. For example, in the above example, one could search for all documents in which "City" is "Noida" that would deliver a result set containing all documents associated with any "3Pillar Office" that is in that particular city.

Apache CouchDB is an example of a document store. CouchDB uses JSON to store data, JavaScript as its query language using MapReduce and HTTP for an API. Data and relationships are not stored in tables as is a norm with conventional relational databases but in fact are a collection of independent documents.

The fact that document style databases are schema-less makes adding fields to JSON documents a simple task without having to define changes first.

- Couchbase and MongoDB are the most popular document based databases.

3. **Column Store NoSQL Database**–

In column-oriented NoSQL database, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited

number of columns that can be created at runtime or the definition of the schema. Read and write is done using columns rather than rows.

In comparison, most relational DBMS store data in rows, the benefit of storing data in columns, is fast search/ access and data aggregation. Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on disk while Columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster.

For example: To query the titles from a bunch of a million articles will be a painstaking task while using relational databases as it will go over each location to get item titles. On the other hand, with just one disk access, title of all the items can be obtained.

Data Model

- **ColumnFamily**: ColumnFamily is a single structure that can group Columns and SuperColumns with ease.
- **Key**: the permanent name of the record. Keys have different numbers of columns, so the database can scale in an irregular way.
- **Keyspace**: This defines the outermost level of an organization, typically the name of the application. For example, '3PillarDataBase' (database name).
- **Column**: It has an ordered list of elements aka tuple with a name and a value defined.

The best known examples are Google's BigTable and HBase & Cassandra that were inspired from BigTable.

BigTable, for instance is a high performance, compressed and proprietary data storage system owned by Google. It has the following attributes:

- **Sparse** – some cells can be empty
- **Distributed** – data is partitioned across many hosts
- **Persistent** – stored to disk
- **Multidimensional** – more than 1 dimension
- **Map** – key and value
- **Sorted** – maps are generally not sorted but this one is

A 2-dimensional table comprising of rows and columns is part of the relational database system.

City	Pincode	Strength	Project
Noida	201301	250	20
Cluj	400606	200	15
Timisoara	300011	150	10
Fairfax	VA 22033	100	5

For above RDBMS table a BigTable map can be visualized as shown below.

```
{
3PillarNoida: {
city: Noida
pincode: 201301
},
details: {
strength: 250
```

```
projects: 20
}
}
{
3PillarCluj: {
address: {
city: Cluj
pincode: 400606
},
details: {
strength: 200
projects: 15
}
},
{
3PillarTimisoara: {
address: {
city: Timisoara
pincode: 300011
},
details: {
strength: 150
projects: 10
}
}
{
3PillarFairfax : {
address: {
city: Fairfax
pincode: VA 22033
},
details: {
strength: 100
projects: 5
}
}
```

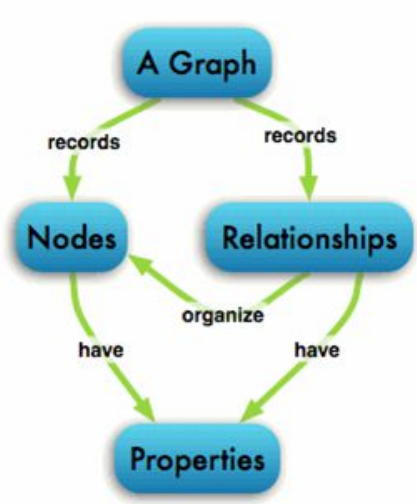
- The outermost keys 3PillarNoida, 3PillarCluj, 3PillarTimisoara and 3PillarFairfax are analogues to rows.
- 'address' and 'details' are called **column families**.
- The column-family 'address' has **columns** 'city' and 'pincode'.
- The column-family details' has **columns** 'strength' and 'projects'.

Columns can be referenced using CloumnFamily.

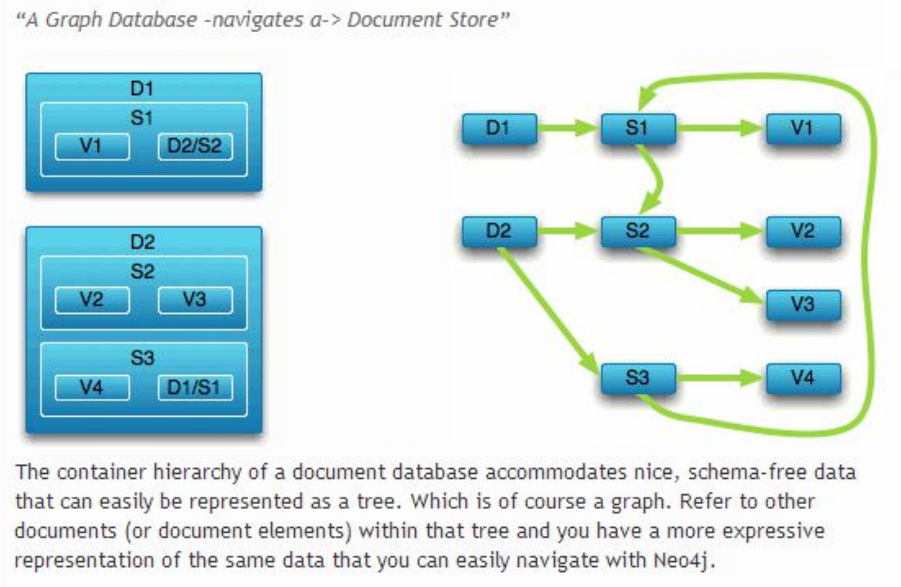
- Google's BigTable, HBase and Cassandra are the most popular column store based databases.

4. Graph Base NoSQL Database

In a Graph Base NoSQL Database, you will not find the rigid format of SQL or the tables and columns representation, a flexible graphical representation is instead used which is perfect to address scalability concerns. Graph structures are used with edges, nodes and properties which provides index-free adjacency. Data can be easily transformed from one model to the other using a Graph Base NoSQL database.



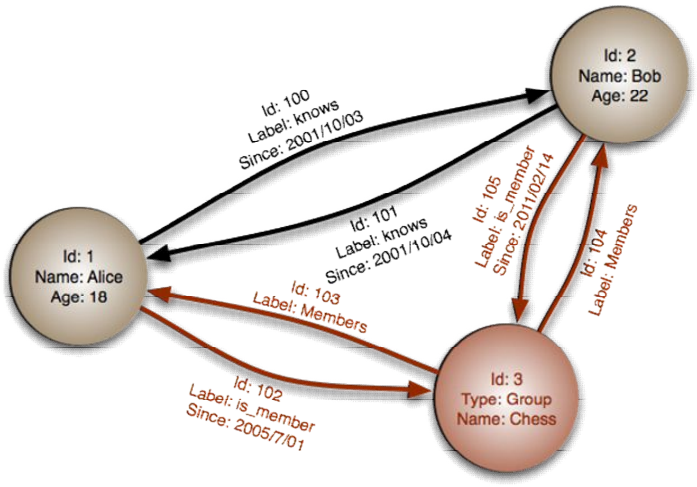
- These databases that uses edges and nodes to represent and store data.
- These nodes are organised by some relationships with one another, which is represented by edges between the nodes.
- Both the nodes and the relationships have some defined properties.



The following are some of the features of the graph based database, which are explained on the basis of the example below:

Labeled, directed, attributed multi-graph : The graphs contains the nodes which are labelled properly with some properties and these nodes have some relationship with one another which is shown by the directional edges. For example: in the following representation, "Alice knows Bob" is shown by an edge that also has some properties.

While relational database models can replicate the graphical ones, the edge would require a join which is a costly proposition.



UseCase –

Any 'Recommended for You' rating you see on e-commerce websites (book/video renting sites) is often derived by taking into account how other users have rated the product in question. Arriving at such a UseCase is made easy using Graph databases.

InfoGrid and Infinite Graph are the most popular graph based databases. InfoGrid allows the connection of as many edges (Relationships) and nodes (MeshObjects), making it easier to represent hyperlinked and complex set of information.

There are two kinds of GraphDatabase offered by InfoGrid, these include the following:

MeshBase – It is a perfect option where standalone deployment is required.

NetMeshBase – It is ideally suited for large distributed graphs and has additional capabilities to communicate with other similar NetMeshbase.

This concludes the second post exemplifying the value in a NoSQL implementation. In this blog post we discussed in detail the different types of NoSQL databases. Watch out for the concluding part of the series which will cover important factors to consider before finalizing which NoSQL database to use.