

A **NoSQL** (originally referring to "non SQL" or "non relational")database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but did not obtain the "NoSQL" moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com. 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.

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability, partition tolerance, and speed. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages (instead of SQL, for instance the lack of ability to perform ad-hoc joins across tables), lack of standardized interfaces, and huge previous investments in existing relational databases.Most NoSQL stores lack true ACID transactions, although a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.

Instead, most NoSQL databases offer a concept of "eventual consistency" in which database changes are propagated to all nodes "eventually" (typically within milliseconds) so queries for data might not return updated data immediately or might result in reading data that is not accurate, a problem known as stale reads. Additionally, 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 that 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 ACID transactions and X/Open XA standards for distributed transaction processing.



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.

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.

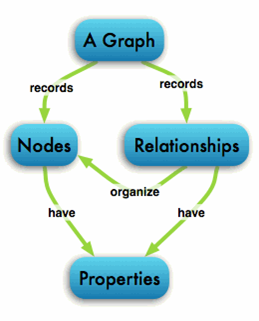
BigTable, for instance is a high performance, compressed and proprietary data storage system owned by Google

.

* 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.

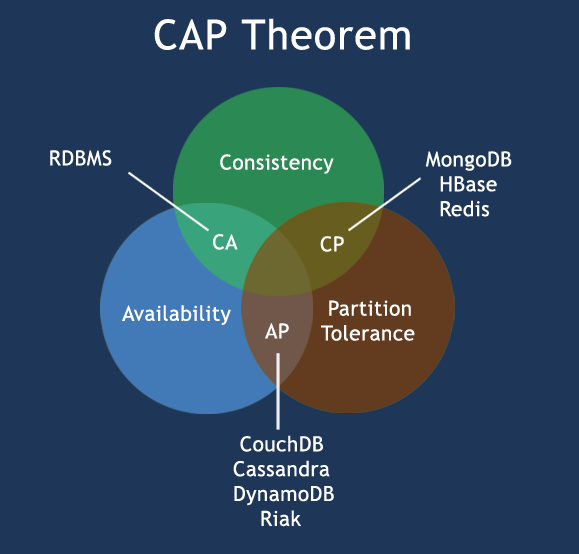
InfoGrid and Infinite Graph are the most popular graph based databases.  InfoGrid allows the connection of as many edges (Relationships) and nodes (MeshObjects), making it easier to represent hyperlinked and complex set of information.

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

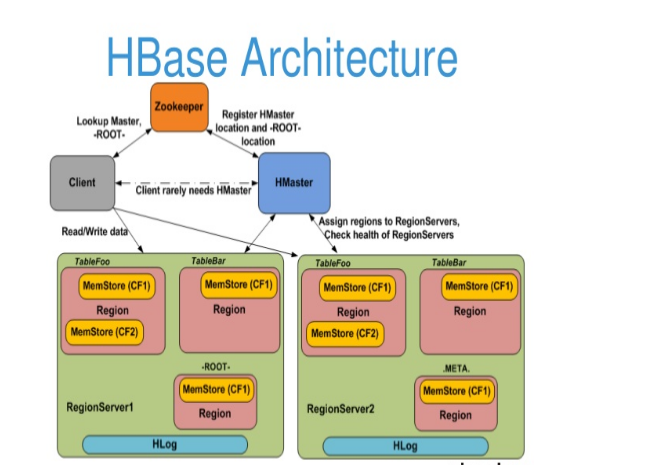
MeshBase–   It is a perfect option where standalone deployment is required.

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



* **Consistency** - A read is guaranteed to return the most recent write for a given client.
* **Availability** - A non-failing node will return a reasonable response within a reasonable amount of time (no error or timeout).
* **Partition Tolerance** - The system will continue to function when network partitions occur.
* 
* **CP** - Consistency/Partition Tolerance - Wait for a response from the partitioned node which could result in a timeout error. The system can also choose to return an error, depending on the scenario you desire. Choose Consistency over Availability when your business requirements dictate atomic reads and writes.
* **AP** - Availability/Partition Tolerance - Return the most recent version of the data you have, which could be stale. This system state will also accept writes that can be processed later when the partition is resolved. Choose Availability over Consistency when your business requirements allow for some flexibility around when the data in the system synchronizes. Availability is also a compelling option when the system needs to continue to function in spite of external errors (shopping carts, etc.)





1. **HMaster**

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster –

* Manages and Monitors the Hadoop Cluster
* Performs Administration (Interface for creating, updating and deleting tables.)
* Controlling the failover
* DDL operations are handled by the HMaster
* Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.

1. **Region Server**

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components –

* Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
* MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
* Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
* HFile is the actual storage file that stores the rows as sorted key values on a disk.

1. **Zookeeper**

HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.

ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers. Various services that Zookeeper provides include –

* Establishing client communication with region servers.
* Tracking server failure and network partitions.
* Maintain Configuration Information
* Provides ephemeral nodes, which represent different region servers.



