# **HBase**

***Introduction***-

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google’s big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

***Architecture –***

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into “Stores”. Stores are saved as files in HDFS.

HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

The master server -

* Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
* Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
* Maintains the state of the cluster by negotiating the load balancing.
* Is responsible for schema changes and other metadata operations such as creation of tables and column families.

The region servers have regions that -

* Communicate with the client and handle data-related operations.
* Handle read and write requests for all the regions under it.
* Decide the size of the region by following the region size thresholds.

***Applications-***

* It is used whenever there is a need to write heavy applications.
* HBase is used whenever we need to provide fast random access to available data.
* Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

***Advantages-***

* HBase is linearly scalable.
* It has automatic failure support.
* It provides consistent read and writes.
* It integrates with Hadoop, both as a source and a destination.
* It has easy java API for client.
* It provides data replication across clusters.

# **Neo4J**

***Introduction-***

Neo4j is one of the popular Graph Databases and Cypher Query Language (CQL). Neo4j is written in Java Language. This tutorial explains the basics of Neo4j, Java with Neo4j, and Spring DATA with Neo4j. Neo4j is the world's leading open source Graph Database which is developed using Java technology. It is highly scalable and schema free (NoSQL).

***Architecture-***

Neo4j Graph Database follows the Property Graph Model to store and manage its data.

Following are the key features of Property Graph Model −

* The model represents data in Nodes, Relationships and Properties
* Properties are key-value pairs
* Nodes are represented using circle and Relationships are represented using arrow keys
* Relationships have directions: Unidirectional and Bidirectional
* Each Relationship contains "Start Node" or "From Node" and "To Node" or "End Node"
* Both Nodes and Relationships contain properties
* Relationships connects nodes

In Property Graph Data Model, relationships should be directional. If we try to create relationships without direction, then it will throw an error message.

In Neo4j too, relationships should be directional. If we try to create relationships without direction, then Neo4j will throw an error message saying that "Relationships should be directional".

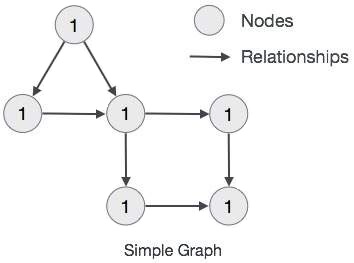
Neo4j Graph Database stores all of its data in Nodes and Relationships. We neither need any additional RRBMS Database nor any SQL database to store Neo4j database data. It stores its data in terms of Graphs in its native format.

Neo4j uses Native GPE (Graph Processing Engine) to work with its Native graph storage format.

The main building blocks of Graph DB Data Model are −

* Nodes
* Relationships
* Properties

Following is a simple example of a Property Graph.



***Applications-***

## [**Managing Network Operations with Graphs**](https://neo4j.com/business-edge/managing-network-operations-with-graphs/)

Using Neo4j graph architecture to map and manage your network, you capture the complex connections of the systems, devices, services, applications and users across your enterprise. As you do, you maintain extreme visibility and control of your network, with the ability to handle problems rapidly as they arise now and in the future.

## [**Relationships Drive Real-Time Recommendations**](https://neo4j.com/business-edge/relationships-drive-real-time-recommendations/)

Real-time recommendations have taken their place as a crucial part of today’s online experience. After starting in the online dating and e-commerce sectors, recommendations are now being used in all industries as merchants and service providers rush to provide personalized solutions.

## [**Graph Databases for Identity and Access Management**](https://neo4j.com/business-edge/identity-and-access-management/)

The growth of enterprise networks and the explosion of connected devices has turned Identity and Access Management (IAM) into one of the top concerns of IT organizations across the globe.

## [**Graph-Based Master Data Management**](https://neo4j.com/business-edge/graph-based-master-data-management/)

To use consistent versions of crucial operating data in systems across your organization, a central source of master data is needed—which includes information about your customers, products, business units, locations, suppliers, partners, processes and policies.

***Advantages-***

* **Flexible data model** − Neo4j provides a flexible simple and yet powerful data model, which can be easily changed according to the applications and industries.
* **Real-time insights** − Neo4j provides results based on real-time data.
* **High availability** − Neo4j is highly available for large enterprise real-time applications with transactional guarantees.
* **Connected and semi structures data** − Using Neo4j, you can easily represent connected and semi-structured data.
* **Easy retrieval** − Using Neo4j, you can not only represent but also easily retrieve (traverse/navigate) connected data faster when compared to other databases.
* **Cypher query language** − Neo4j provides a declarative query language to represent the graph visually, using an ascii-art syntax. The commands of this language are in human readable format and very easy to learn.
* **No joins** − Using Neo4j, it does NOT require complex joins to retrieve connected/related data as it is very easy to retrieve its adjacent node or relationship details without joins or indexes.



# ***Comparision***

|  |  |  |
| --- | --- | --- |
| Name | **HBase**[**X**](https://db-engines.com/en/system/Neo4j) | **Neo4j**[**X**](https://db-engines.com/en/system/HBase) |
| Description | Wide-column store based on Apache Hadoop and on concepts of BigTable | Open source graph database |
| Primary database model | [Wide column store](https://db-engines.com/en/article/Wide+Column+Stores) | [Graph DBMS](https://db-engines.com/en/article/Graph+DBMS) |
| |  |  | | --- | --- | | [DB-Engines Ranking](https://db-engines.com/en/ranking) | [ranking trend](https://db-engines.com/en/ranking_trend/system/HBase;Neo4j) | | [Trend Chart](https://db-engines.com/en/ranking_trend/system/HBase%3BNeo4j) | | |  |  |  | | --- | --- | --- | | Score | 60.77 | | | Rank | #17 | [Overall](https://db-engines.com/en/ranking) | |  | #2 | [Wide column stores](https://db-engines.com/en/ranking/wide+column+store) | | |  |  |  | | --- | --- | --- | | Score | 41.88 | | | Rank | #22 | [Overall](https://db-engines.com/en/ranking) | |  | #1 | [Graph DBMS](https://db-engines.com/en/ranking/graph+dbms) | |
| Website | [hbase.apache.org](http://hbase.apache.org/) | [neo4j.com](https://neo4j.com/) |
| Technical documentation | [hbase.apache.org](http://hbase.apache.org/) | [neo4j.com/­docs](https://neo4j.com/docs/) |
| Developer | Apache Software Foundation | Neo4j, Inc. |
| Initial release | 2008 | 2007 |
| Current release | 1.4.3, April 2018 | 3.3.5, April 2018 |
| License | Open Source | Open Source |
| Cloud-based | no | no |
| Implementation language | Java | Java, Scala |
| Server operating systems | Linux Unix Windows | Linux  OS X Solaris Windows |
| Data scheme | schema-free | schema-free and schema-optional |
| Typing | no | yes |
| XML support | no |  |
| Secondary indexes | no | yes |
| SQL | no | no |
| APIs and other access methods | Java API RESTful HTTP API Thrift | Cypher query language Java API Neo4j-OGM  RESTful HTTP API Spring Data Neo4j TinkerPop 3 |
| Supported programming languages | C C# C++ Groovy Java PHP Python Scala | .Net Clojure Elixir Go Groovy Haskell Java JavaScript Perl PHP Python Ruby Scala |
| Server-side scripts | yes | yes |
| Triggers | yes | yes |
| Partitioning methods | Sharding | none |
| Replication methods | selectable replication factor | Causal Clustering using Raft protocol |
| MapReduce | yes | no |
| Consistency concepts | Immediate Consistency | Causal and Eventual Consistency configurable in Causal Cluster setup Immediate Consistency in stand-alone mode |
| Foreign keys | no | yes |
| Transaction concepts | no | ACID |
| Concurrency | yes | yes |
| Durability | yes | yes |
| In-memory capabilities | no |  |
| User concepts | Access Control Lists (ACL) | Users, roles and permissions. Pluggable authentication with supported standards (LDAP, Active Directory, Kerberos) |