# HIVE

<https://data-flair.training/blogs/apache-hive-tutorial/>

<https://www.tutorialspoint.com/hive/hive_introduction.htm>

# What is Hive?

****Apache Hive**** is an **open source data warehouse system** built on top of Hadoop Haused for querying and analyzing large datasets stored in Hadoop files.

Initially, you have to write complex **[Map-Reduce](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)** jobs, but now with the help of the Hive, you just need to submit merely****SQL**** queries.

Hive is mainly targeted towards users who are comfortable with SQL.

Hive use language called ****HiveQL**** ****(Hive Query Language)****(HQL), which is similar to SQL.

HiveQL (compiler) automatically translates SQL-like queries into MapReduce jobs.

Hive abstracts the complexity of Hadoop. The main thing to notice is that there is no need to learn java for Hive.

The Hive generally runs on your workstation and converts your SQL query into a series of jobs for execution on a **[Hadoop cluster](http://data-flair.training/blogs/install-hadoop-2-x-ubuntu-hadoop-multi-node-cluster/)**. Apache Hive organizes data into tables. This provides a means for attaching the structure to data stored in ****HDFS****.

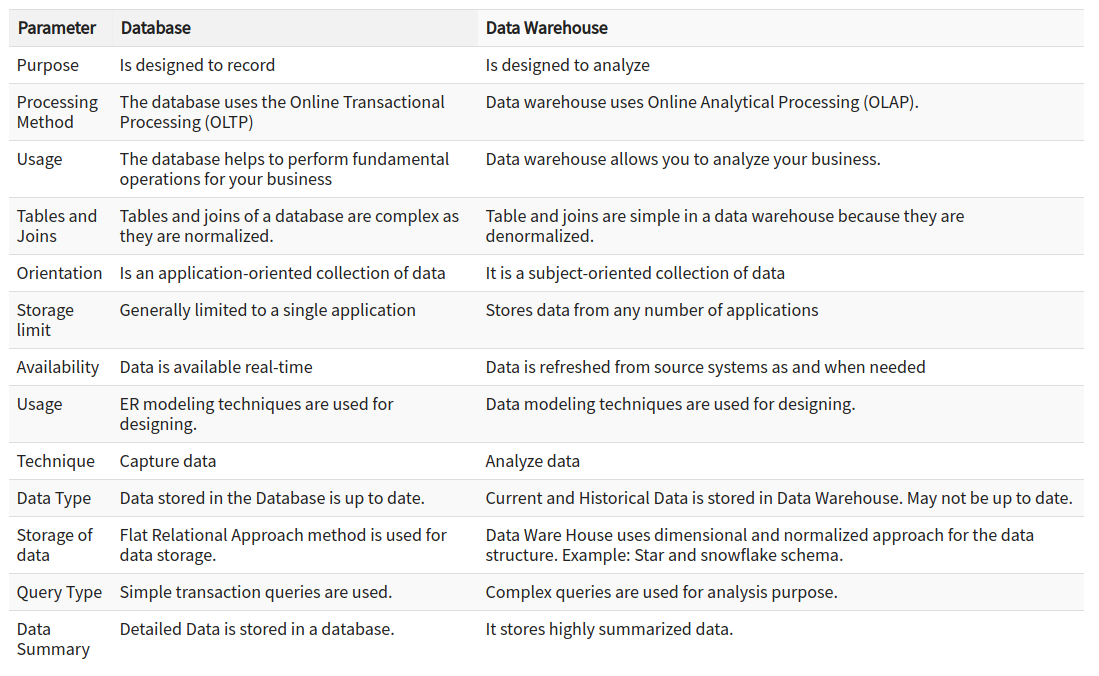
few misconceptions occur about Hive. So, let’s clarify that:

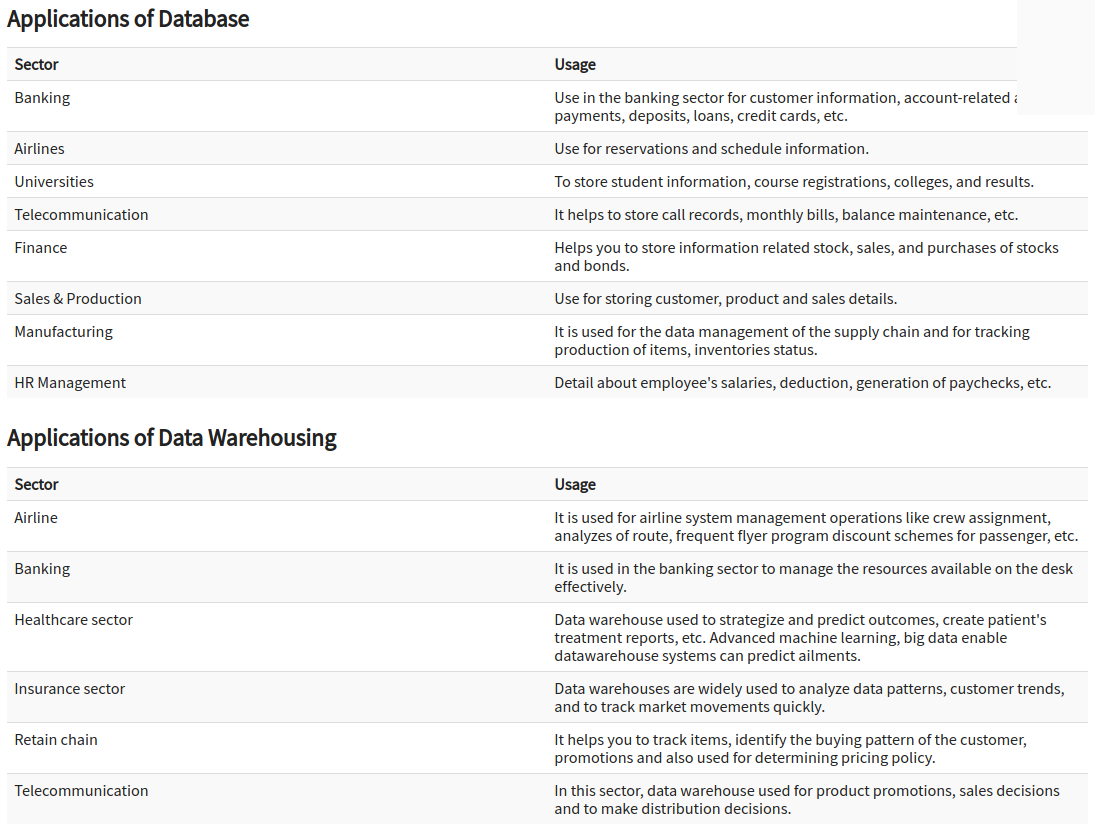
* We can say it is not a relational database
* Also, not a design for OnLine Transaction Processing (OLTP)
* Even not a language for real-time queries and row-level updates

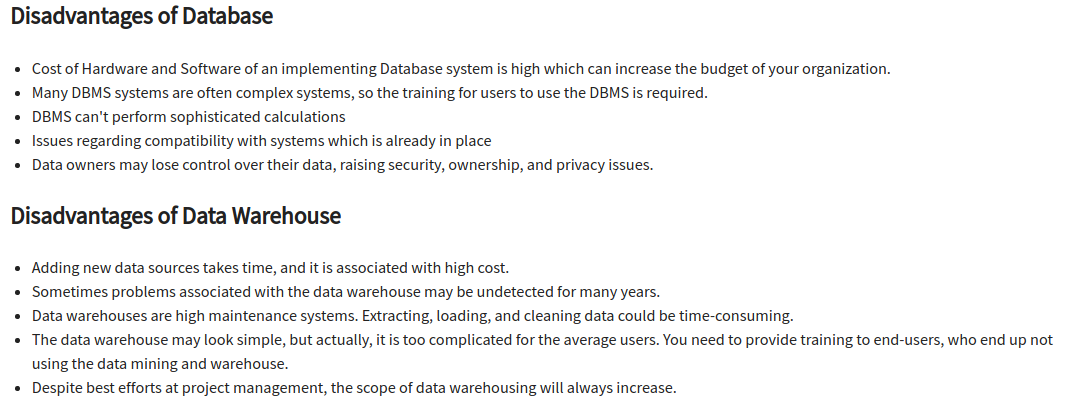
# **Difference between Database and Data Warehouse**

A database is a collection of related data which represents some elements of the real world. It is designed to be built and populated with data for a specific task. It is also a building block of your data solution.

A data warehouse is an information system which stores historical and commutative data from single or multiple sources. It is designed to analyze, report, integrate transaction data from different sources.







# **Apache History Hive**

Data Infrastructure Team at Facebook developed Hive. Apache Hive is also one of the technologies that are being used to address the requirements at Facebook. It is very popular with all the users internally at Facebook. It is being used to run thousands of jobs on the cluster with hundreds of users, for a wide variety of applications.  
Apache Hive-Hadoop cluster at Facebook stores more than 2PB of raw data. It regularly loads 15 TB of data on a daily basis.  
Now it is being used and developed by a number of companies like Amazon, IBM, Yahoo, Netflix, Financial Industry Regulatory Authority (FINRA) and many others.

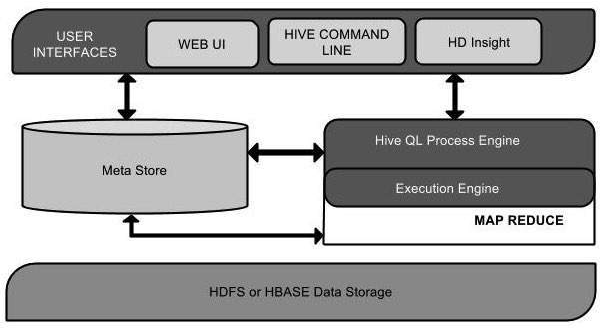
# **Why Apache Hive?**

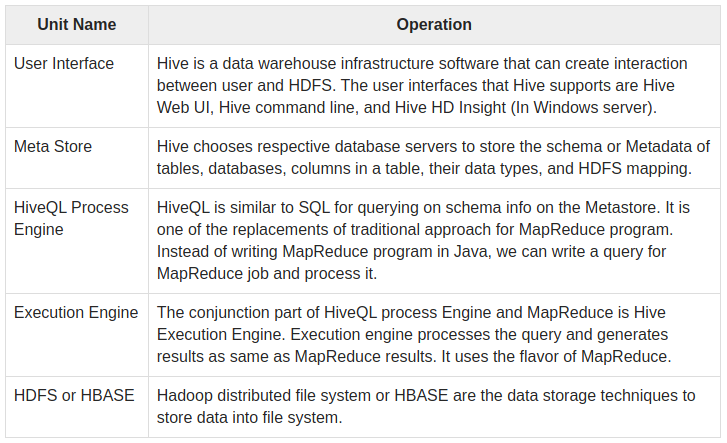
Apache Hive saves developers from writing complex Hadoop MapReduce jobs for ad-hoc requirements. Hence, hive provides summarization, analysis, and query of data. Hive is very fast and scalable. It is highly extensible. Since Apache Hive is similar to SQL, hence it becomes very easy for the SQL developers to learn and implement Hive Queries.

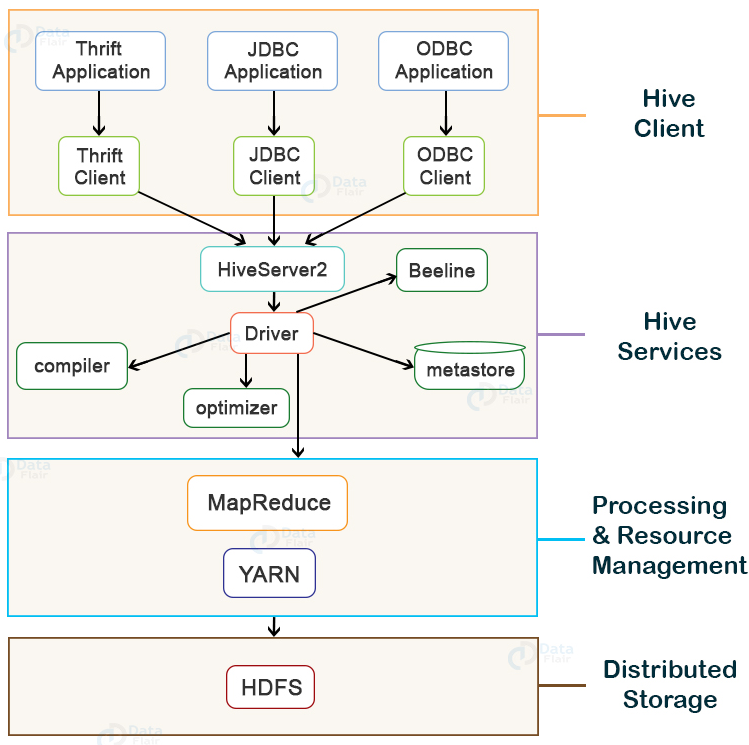
**The most important feature of Apache Hive is that to learn Hive we don’t have to learn Java.**

As we know it is mainly used for data querying, analysis, and summarization. Moreover, it helps to improve the developer productivity. **However, that comes at the cost of increasing latency and decreasing efficiency**

# **Hive Architecture**







The above figure shows the architecture of Apache Hive and its major components. The major components of Apache Hive are:

1. **[Hive Client](https://data-flair.training/blogs/apache-hive-architecture/" \l "Hive-Client)**
2. **[Hive Services](https://data-flair.training/blogs/apache-hive-architecture/" \l "Hive-Services)**
3. **[Processing and Resource Management](https://data-flair.training/blogs/apache-hive-architecture/" \l "Processing-and-Resource-Management)**
4. **[Distributed Storage](https://data-flair.training/blogs/apache-hive-architecture/" \l "Distributed-Storage)**

### Hive Client

Hive supports applications written in any language like Python, Java, C++, Ruby, etc. using JDBC, ODBC, and Thrift drivers, for performing queries on the Hive. Hence, one can easily write a hive client application in any language of its own choice.

Hive clients are categorized into three types:

#### 1. Thrift Clients

The Hive server is based on Apache Thrift so that it can serve the request from a thrift client.

#### 2. JDBC client

Hive allows for the Java applications to connect to it using the JDBC driver. JDBC driver uses Thrift to communicate with the Hive Server.

#### 3. ODBC client

Hive ODBC driver allows applications based on the ODBC protocol to connect to Hive. Similar to the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive Server.

### Hive Service

To perform all queries, Hive provides various services like the Hive server2, Beeline, etc. The various services offered by Hive are:

#### 1. Beeline

The Beeline is a command shell supported by HiveServer2, where the user can submit its queries and command to the system. It is a ****JDBC**** client that is based on ****SQLLINE CLI**** (pure Java-console based utility for connecting with relational database and executing SQL queries).

#### 2. Hive Server 2

HiveServer2 is the successor of HiveServer1. HiveServer2 enables clients to execute queries against the Hive. It allows multiple clients to submit requests to Hive and retrieve the final results. It is basically designed to provide the best support for open API clients like JDBC and ODBC.

****Note:**** **Hive server1, also called a Thrift server**, is built on Apache Thrift protocol to handle the cross-platform communication with Hive. It allows different client applications to submit requests to Hive and retrieve the final results. **It does not handle concurrent requests from more than one client due to which it was replaced by HiveServer2.**

#### 3. Hive Driver

The [Hive](https://hive.apache.org/) driver receives the ****HiveQL**** statements submitted by the user through the command shell. It creates the session handles for the query and sends the query to the compiler.

#### 4. Hive Compiler

Hive compiler parses the query. It performs semantic analysis and type-checking on the different query blocks and query expressions by using the metadata stored in metastore and generates an execution plan.

The execution plan created by the compiler is the ****DAG(Directed Acyclic Graph)****, where each stage is a map/reduce job, operation on HDFS, a metadata operation.

#### 5. Optimizer

Optimizer performs the transformation operations on the execution plan and splits the task to improve efficiency and scalability.

#### 6. Execution Engine

Execution engine, after the compilation and optimization steps, executes the execution plan created by the compiler in order of their dependencies using Hadoop.

#### 7. Metastore

Metastore is a central repository that stores the metadata information about the structure of tables and partitions, including column and column type information.

It also stores information of serializer and deserializer, required for the read/write operation, and HDFS files where data is stored. This metastore is generally a relational database(**embedded derby database**).

Metastore provides a Thrift interface for querying and manipulating Hive metadata.

We can configure metastore in any of the two modes:

* ****Remote:**** In remote mode, metastore is a Thrift service and is useful for non-Java applications.
* ****Embedded:**** In embedded mode, the client can directly interact with the metastore using JDBC.

#### 8. HCatalog

HCatalog is the table and storage management layer for Hadoop. It enables users with different data processing tools such as Pig, MapReduce, etc. to easily read and write data on the grid.

It is built on the top of Hive metastore and exposes the tabular data of Hive metastore to other data processing tools.

#### 9. WebHCat

WebHCat is the REST API for HCatalog. It is an HTTP interface to perform Hive metadata operations. It provides a service to the user for running Hadoop MapReduce (or YARN), Pig, Hive jobs.

### Processing Framework and Resource Management

Hive internally uses a ****MapReduce**** framework as a defacto engine for executing the queries.

MapReduce is a software framework for writing those applications that process a massive amount of data in parallel on the large clusters of commodity hardware. MapReduce job works by splitting data into chunks, which are processed by map-reduce tasks.

Read **[MapReduce](https://data-flair.training/blogs/hadoop-mapreduce-tutorial/)** article to learn the MapReduce framework in detail.

<https://data-flair.training/blogs/hadoop-mapreduce-tutorial/>

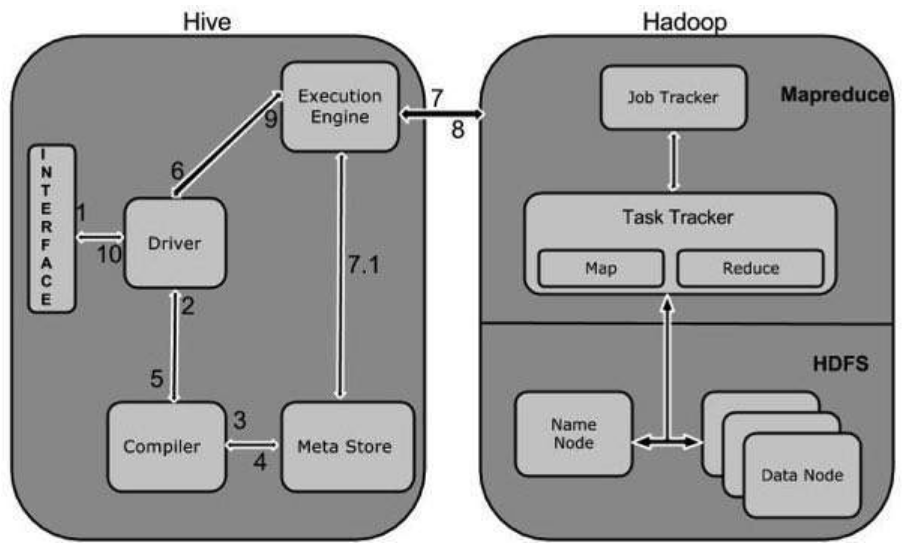
### Distributed Storage

Hive is built on top of Hadoop, so it uses the underlying Hadoop Distributed File System for the distributed storage.

Go through the **[HDFS Introduction](https://data-flair.training/blogs/hadoop-hdfs-tutorial/)** article to learn HDFS.

<https://data-flair.training/blogs/hadoop-hdfs-tutorial/>

# **Working of HIVE**



|  |  |
| --- | --- |
| **Step No.** | **Operation** |
| 1 | **Execute Query**  The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute. |
| 2 | **Get Plan**  The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query. |
| 3 | **Get Metadata**  The compiler sends metadata request to Metastore (any database). |
| 4 | **Send Metadata**  Metastore sends metadata as a response to the compiler. |
| 5 | **Send Plan**  The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete. |
| 6 | **Execute Plan**  The driver sends the execute plan to the execution engine. |
| 7 | **Execute Job**  Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job. |
| 7.1 | **Metadata Ops**  Meanwhile in execution, the execution engine can execute metadata operations with Metastore. |
| 8 | **Fetch Result**  The execution engine receives the results from Data nodes. |
| 9 | **Send Results**  The execution engine sends those resultant values to the driver. |
| 10 | **Send Results**  The driver sends the results to Hive Interfaces. |

# **Features of Hive**

In this section of Hive Tutorial, we study Apache Hive features. So, let’s discuss all-

* The best feature is it offers data summarization, query, and analysis in much easier manner.
* However, to process data without actually storing in**[HDFS](https://data-flair.training/blogs/features-of-hadoop-hdfs/)**, Hive supports external tables.
* Moreover, it fits the low-level interface requirement of Hadoop perfectly.
* Also, to improve performance it supports partitioning of data at the level of tables.
* While it comes to optimizing logical plans, Hive has a rule-based optimizer available.
* Hive is scalable, familiar, and extensible in nature.
* For working with HiveQL Knowledge of basic SQL query is enough. We don’t need any knowledge of programming language.
* By using Hive, it is possible to process structured data in Hadoop.
* Hive makes Querying very simple, as same as SQL.
* By using Hive, it is possible to run Ad-hoc queries for the data analysis

# **Limitation of Hive**

Apache Hive Tutorial discuss this following limitation of Hive. Let’s discuss all –

* We can not perform real-time queries with Hive. Also, it does not offer row-level updates.
* Moreover,  for interactive data browsing Hive offers acceptable latency.
* Also, we can say Hive is not the right choice for online transaction processing.
* While it comes to latency, for Hive queries latency is generally very high.

# **Pig vs Hive vs Hadoop MapReduce**

|  |  |  |  |
| --- | --- | --- | --- |
| **Topics** | **Hive** | **MapReduce** | **Pig** |
| **Language** | It has SQL like Query language. | Also, has compiled language | It has the scripting language |
| **Abstraction** | It has a Low level of Abstraction | has the High level of Abstraction | It has the High level of Abstraction |
| **Line of codes** | Comparatively less no. of the line of codes from both MapReduce and Pig | It has More line of codes | Comparatively less no. of the line of codes from MapReduce |
| **Development Efforts** | Comparatively fewer development efforts from both MapReduce and Pig | More development effort is involved | Comparatively less development effort |
| **Code Efficiency** | Code efficiency is relatively less | It has high Code efficiency | Code efficiency is relatively less. |