

Hive

Apache Hive is a data WH system that is built on top of hadoop.

spark sql

- Data analysts
- SQL developers

Code working on data
no Java or MR code

Facebook developed Hive ≈ 2010

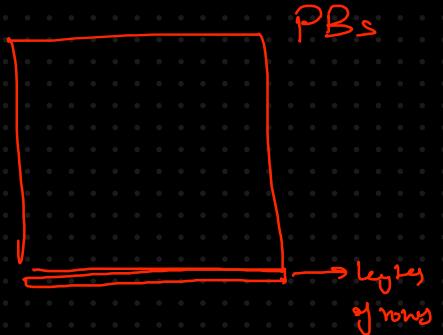
SQL like interface
HQL

library =

librarian with a
ipad / computer

PBs of data

How Hive makes data processing easier:



without hive

with Hive

Java based MR code
to process data

SQL like
queries

Programmers

SQL users
Analysts

Few Common Ques/misconceptions about Hive

Misconception

1. Hive is a database like SQL / PostgreSQL SQL ?

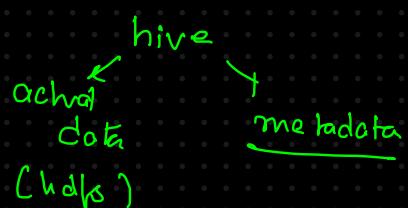
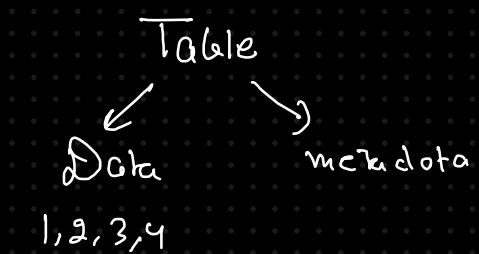
Hive is not a database but a data warehouse tool built on Hadoop.

An interface to Query large datasets stored in hdfs

Feature	Traditional RDBMS (MySQL, PostgreSQL)	Apache Hive
Storage	Stores structured data in tables on disk	Uses HDFS for distributed storage
Query Execution	Uses in-memory processing	Uses MapReduce, Tez, or Spark
Transaction Type	OLTP (Row-level operations)	OLAP (Batch processing)

Feature	Traditional Databases (MySQL, PostgreSQL)	Apache Hive
Storage	Stores data in structured tables on disks	Stores data in HDFS (distributed storage)
Query Processing	Executes queries in-memory	Uses MapReduce, Tez, or Spark for distributed processing
Schema	Schema-on-Write (Data must fit schema)	Schema-on-Read (Can query raw files)
Transaction Type	OLTP (Transactional systems)	OLAP (Batch processing, analytics)
Speed	Fast for small queries, real-time updates	Optimized for large-scale batch processing

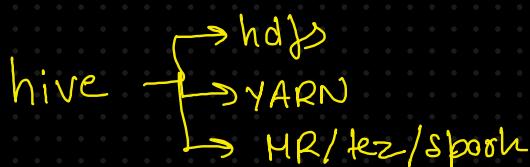
Hive is a metadatabase



2. Hive is a replacement for Hadoop

Built on top of hadoop & depends on hdfs, YARN

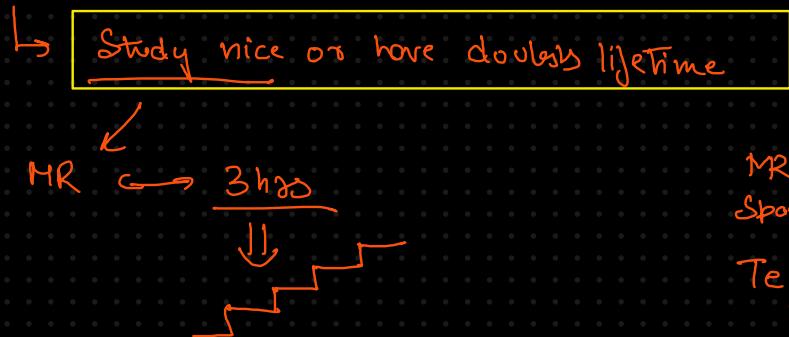
Query engine



```
Hive Session ID = 61b9b5ae-7609-498c-a06d-4fa12313a968
> ;
hive> set hive.execution.engine
> ;
hive> set hive.execution.engine=tez
hive> set hive.execution.engine=mr
> ;
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

Why I teach things in depth?

end-to-end



3. Hive Queries run like normal SQL Queries DB

1st ⇒ abstraction of MR, Tez, Spark

~AI
select * from table → MR code

2nd distributed

4. Hive can perform Row level transactions like MySQL

Hive as a technology
is designed for batch processing

Bigrush to kill on out

→ it will not modify individual records, but create new partitions

5. Hive is always slower than Spark

Hive on MR ≈ slower

Hive on Spark/Tez ⇒ faster than MR

Hive is better for structured storage and complex queries

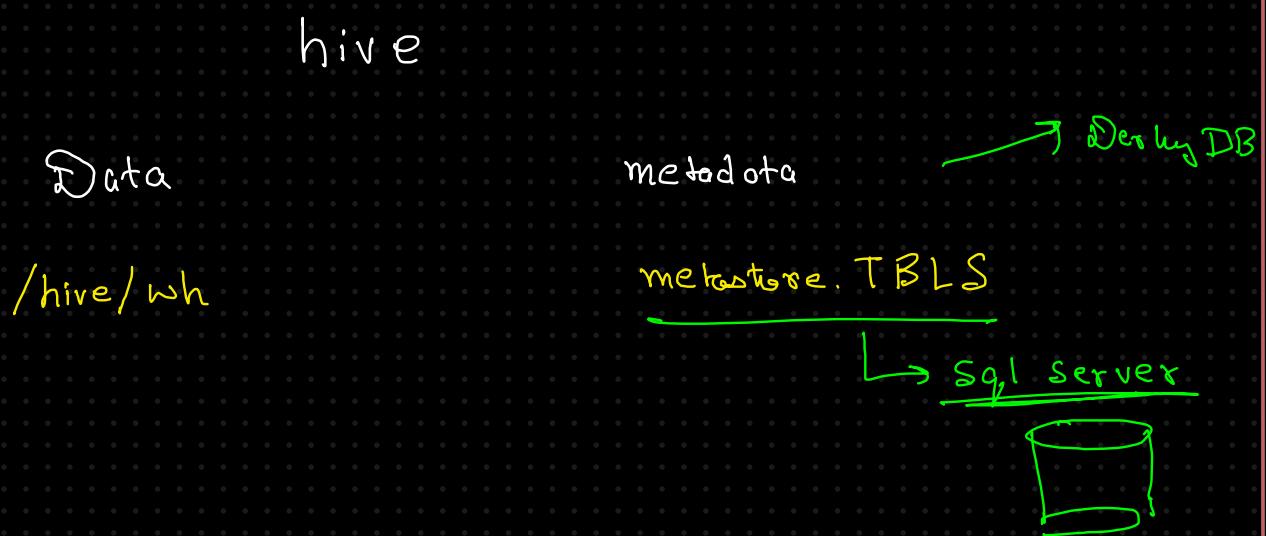
Use Case	Use Hive	Use Spark
Large-scale batch processing	✓ Yes	✗ No
Real-time analytics	✗ No	✓ Yes
ETL jobs on structured data	✓ Yes	✗ No
Streaming data processing	✗ No	✓ Yes

Hive only works with HDFS

S3, ADLS, GCS, hdfs

Feature	Hive Advantage
SQL-Like Interface	Allows non-programmers to work with big data.
Scalability	Handles petabytes of data.
Optimized Execution	Uses MapReduce, Tez, or Spark for distributed processing.
Storage Flexibility	Works with HDFS, S3, Azure Blob, GCS.
Schema Flexibility	Schema-on-read enables analyzing raw files.

Feature	Hive CLI	Beeline (Recommended)
Connection Type	Directly connects to Hive	Uses JDBC for remote execution
Security	No authentication	Supports authentication (LDAP, Kerberos)
Multiple Connections	Single session only	Supports multiple concurrent sessions
Performance	More resource-heavy	Optimized for query performance
Recommended?	✗ Deprecated	✓ Yes, for production use

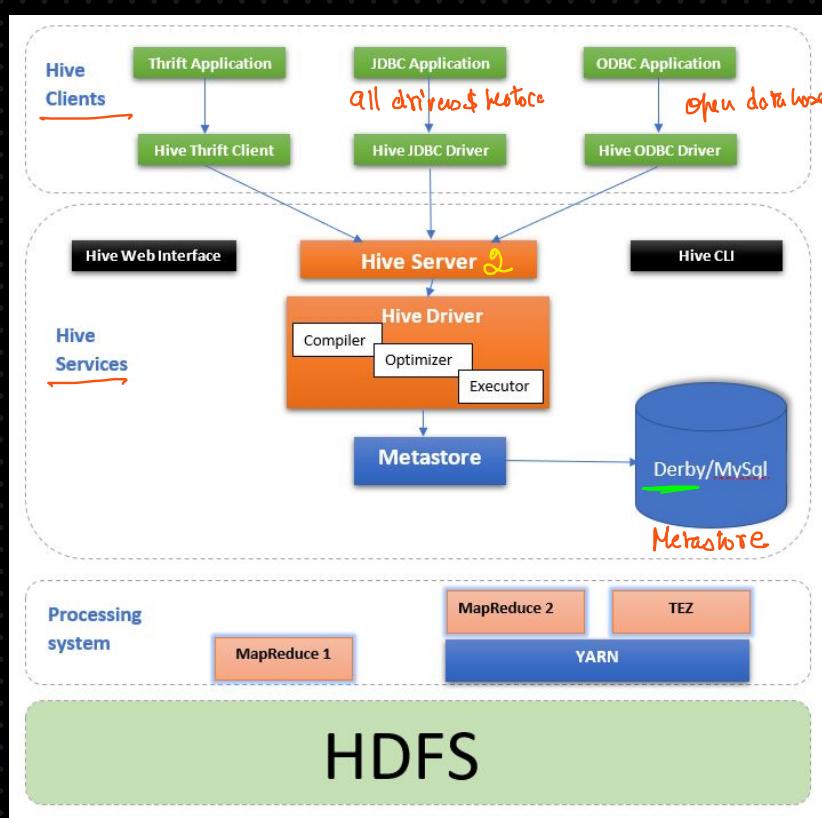


Hive Architecture

Hive is built on top of Hadoop & provides a SQL like interface for querying and managing the large datasets.

Be on time

instead of waiting us this
comes
 Please focus on yourself.



Beeline

! jdbc : 16000

Client

Server

Front

Backend

Results

Request

Hive Clients : Allows user to connect/interact with hive via commands or application

JDBC, ODBC, Thrift
 Java application
 Apache Thrift

Hive Server:- Acts as a middle layer between client & execution engine

manages client request,
 sends them to execution & returns the result

Hive Server 2 successor of 1

Drivers : Control query execution from start to end

Parsing & query
Communicate with
Compiler, optimizers
& execution engine

Compiler parses and Translate HiveQL → execution plan

Query → tasks

Optimizers Improves execution efficiency

Applies technique like
predicate pushdown,
join reordering

Execution engine run the execution plan

Using Hadoop's framework

MR, Tez, Spark

↓
default

Metastore Store the metadata about databases, columns, tables, partitions, schema.

Essential for Hive to keep track of data

Select id, name from customer

you haven't showed us on how to connect to the metastore this time as well to see the content inside Metastore

← Revise Revise Revise

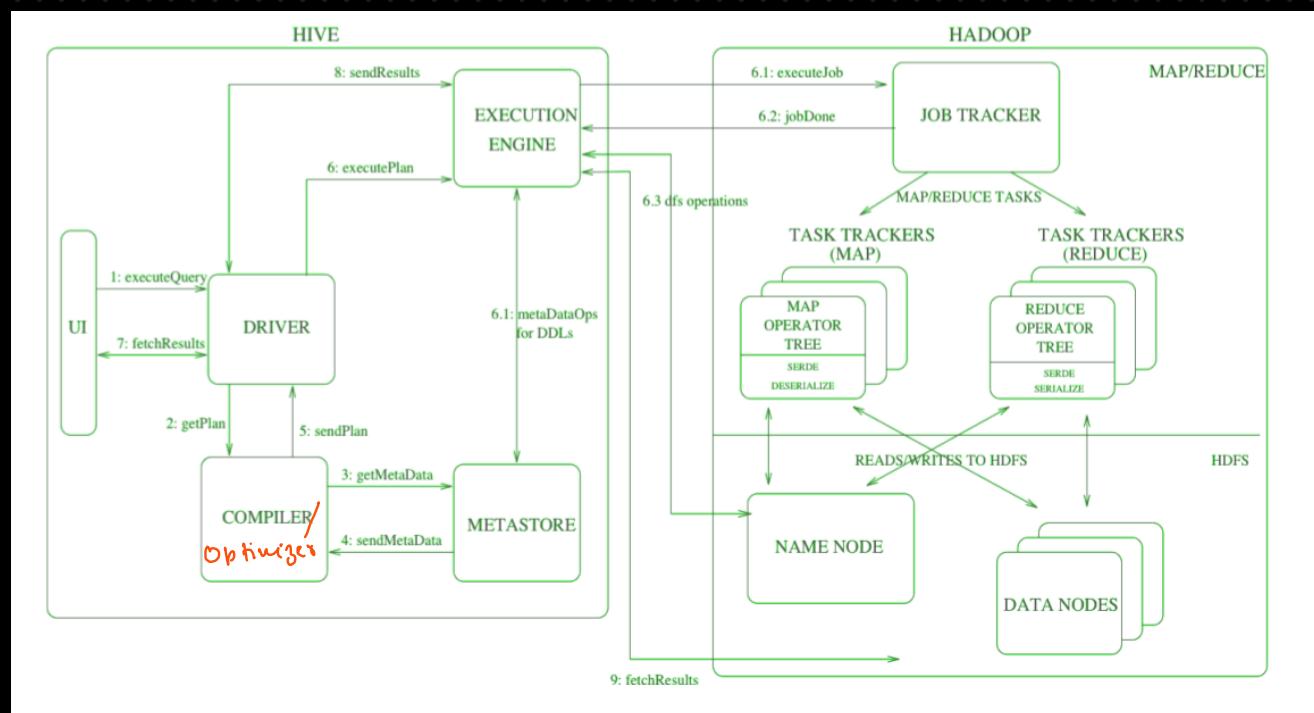
Storage layers (HDFS)
Actual location where data resides

External managed
↓
hdfs
hive not on hdfs

Metastore

→ Remote → metastore bubble
and useful for
non-Java application

↪ Embedded



1. User submits a Query via CLI, JDBC, ODBC, Thrift
2. Driver receives the Query & forwards it to the Compiler
3. Compiler checks for the Schemas & retrieve metadata from metastore
4. Optimizer optimizes the Query by restructuring, predicate pushdown etc.
5. The execution engine determine the processing framework
6. Tasks are assigned to YARN or ST
7. HDFS data is read and processed via MR, Tez, Spark
8. Results are written back to hdfs & returned to user

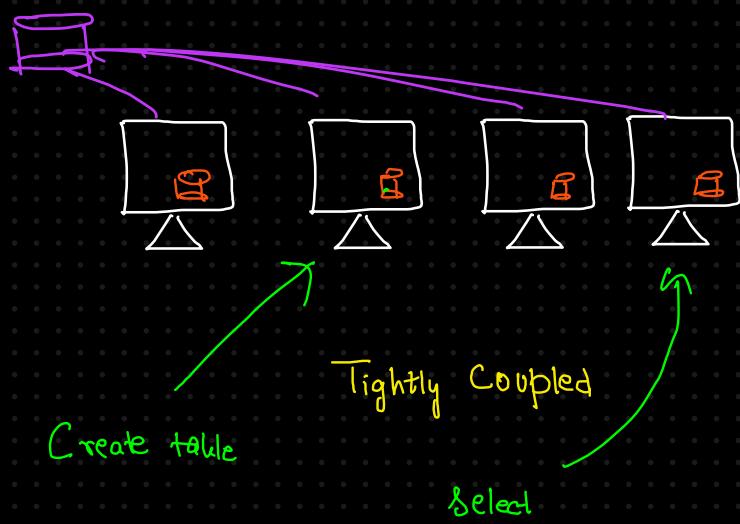
Derby DB

default dB in hive

tightly coupled

does not support concurrent operations

Limited Scalability



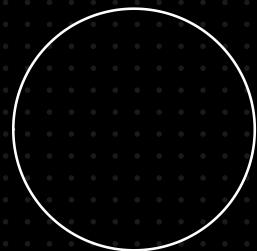
Aspect	Derby DB (Local Mode)	MySQL (Remote Mode)
What is it?	An embedded relational database (RPD) that comes bundled with Hive.	A standalone relational database server used for Hive metadata.
Mode of Operation	Local mode (Embedded): Stores metadata on the same machine as Hive.	Remote mode: Stores metadata on a dedicated MySQL server.
Multi-User Access	✗ No (Single-user only, since it locks the database for a single process).	✓ Yes (Multiple users can access Hive concurrently).
Metadata Sharing	✗ Cannot share metadata across nodes (limited to local machine).	✓ Shared and accessible across the entire Hadoop cluster.
Use Case	Suitable for local testing or single-user development environments.	Suitable for production environments with multiple users and nodes.
Scalability	✗ Poor (Not suitable for large-scale deployments).	✓ High (Handles large-scale metadata operations).
Performance	Slower for large-scale queries due to local storage.	Faster due to optimized querying and indexing.
Persistence	Metadata is stored locally and may be lost if the local system fails.	Metadata is stored centrally and is more durable.

Hive External vs Managed Table

Aspect	Internal (Managed) Table	External Table
Storage Location	Data is stored in Hive's default warehouse directory (/user/hive/warehouse/).	Data is stored at the user-specified HDFS path.
Data Management	Hive manages both the table schema and data.	Hive manages only the schema, not the data.
Data Deletion	Dropping the table deletes the data permanently from HDFS.	Dropping the table deletes only the schema, but the data remains in HDFS.
Use Case	Suitable for temporary or intermediate tables.	Preferred when sharing datasets across multiple systems.
LOCATION Clause	Not required (Hive defaults to the warehouse directory).	Required to specify the external storage path.
Table Property	<code>EXTERNAL=FALSE</code> (Default)	<code>EXTERNAL=TRUE</code> (Set explicitly or using <code>EXTERNAL</code> keyword during creation).

Hive Partitioning

+ 91 - ↗

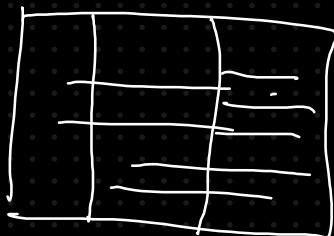


where `customer.State =`

'Mumbai'

and

`id > 1000`



dynamic

hive automatically
creates the partitions
based on column values.

Static

manually specifying
the partition
values

