UNIT - 5

PIG ,HIVE ,HBASE

Topics to be covered...

Hadoop Eco System Framework

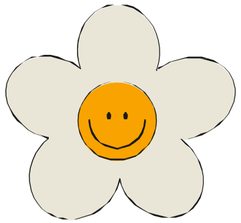
Pig
Hive
HBase

Pig

Introduction to PIG
Execution Modes of Pig
Comparison of Pig with Databases
Grunt
Pig Latin
User Defined Functions
Data Processing operators

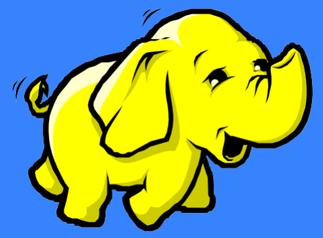
Hive

Hive concepts
Hive architecture and installation
Comparison with traditional databases
HiveQL
Querying data and user defined functions
Sorting and aggregating
Map Reduce scripts
Joins
Subqueries



HBase

HBase concepts
HBase vs RDBMS
Schema design
Zookeeper
IBM Big Data strategy
Introduction to Infosphere
Big Insights and Big Sheets
Introduction to Big SQL.



Hadoop Eco System Framework

Pig Application
Hive Application
HBase Application

Pig Application

Pig Application

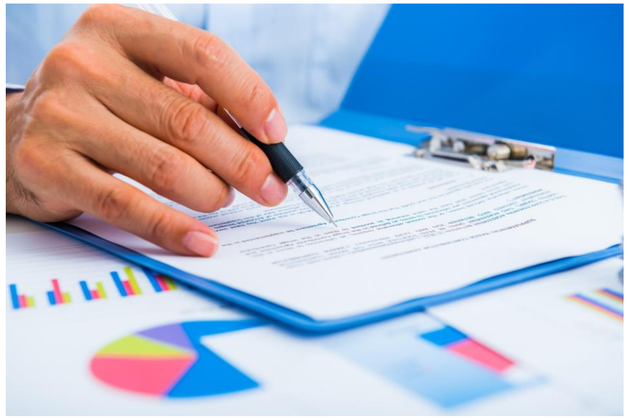
- Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce.
- It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes.
- First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language.
- Internally Pig Engine(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction.
- Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

Need of Pig

- One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task.
- Apache Pig reduces the time of development using the multi-query approach.
- Pig is beneficial for the programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

Features of Apache Pig

- **Rich set of operators:** Apache Pig provides rich sets of operators like the filters, join, sort, etc.
- **Ease of programming:** Easy to learn, read and write. Especially for SQL-programmer.
- **Line of Code:** Fewer lines of code.
- **Extensibility:** Using the existing operators, users can develop their own functions to read, process, and write data.
- **Handles all kinds of data:** Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

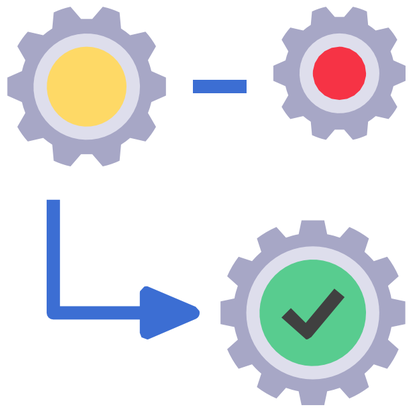


Execution Modes of Pig

Execution Modes of Pig

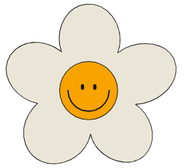
- **Local Mode:** In this mode, all the files are installed and run from your local host and local file system. There is no need of Hadoop or HDFS. This mode is generally used for testing purpose.

- **MapReduce Mode:** MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig Latin statements to process the data, a MapReduce job is invoked in the back-end to perform a particular operation on the data that exists in the HDFS.



Comparison of Pig with Databases

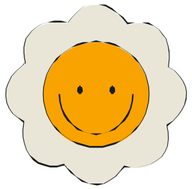
Comparison of Pig with Databases



Pig Vs MapReduce:

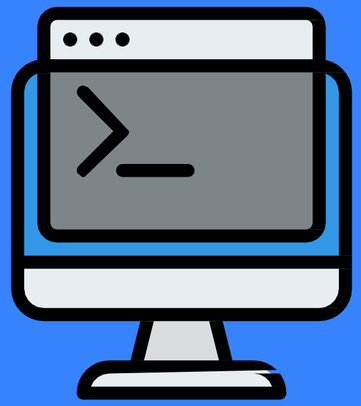
Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

Comparison of Pig with Databases



Pig Vs SQL:

Pig	SQL
Pig Latin is a **procedural** language.	SQL is a **declarative** language.
In Apache Pig, **schema** is optional. We can store data without designing a schema (values are stored as $01, $02 etc.)	Schema is mandatory in SQL.
The data model in Apache Pig is **nested relational**.	The data model used in SQL **is flat relational**.
Apache Pig provides limited opportunity for **Query optimization**.	There is more opportunity for query optimization in SQL.



Grunt

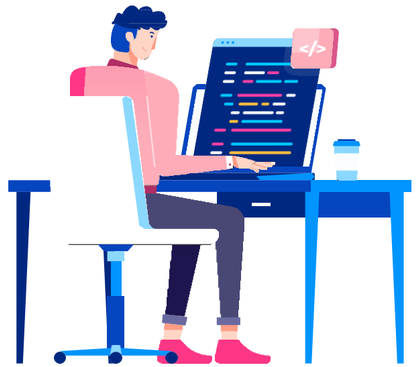
Grunt

- Grunt shell is a shell command. The Grunts shell of Apace pig is mainly used to write pig Latin scripts.
- Pig script can be executed with grunt shell which is native shell provided by Apache pig to execute pig queries.
- It does not provide a number of commands found in standard Unix shells, such as pipes, redirection, and background execution.
- You can get a list of commands using the help commands.
- To exit Grunt type quit or Ctrl-D.

We can invoke shell commands using sh and fs.

Syntax of sh command:

[grunt> sh ls]



Pig Latin

Pig Latin

The Pig Latin is a data flow language used by Apache Pig to analyze the data in Hadoop. It is a textual language that abstracts the programming from the Java MapReduce idiom into a notation.

Pig Latin Statements:

The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

- It can span multiple lines.
- Each statement must end with a semi-colon.
- It may include expression and schemas.
- By default, these statements are processed using multi-query execution

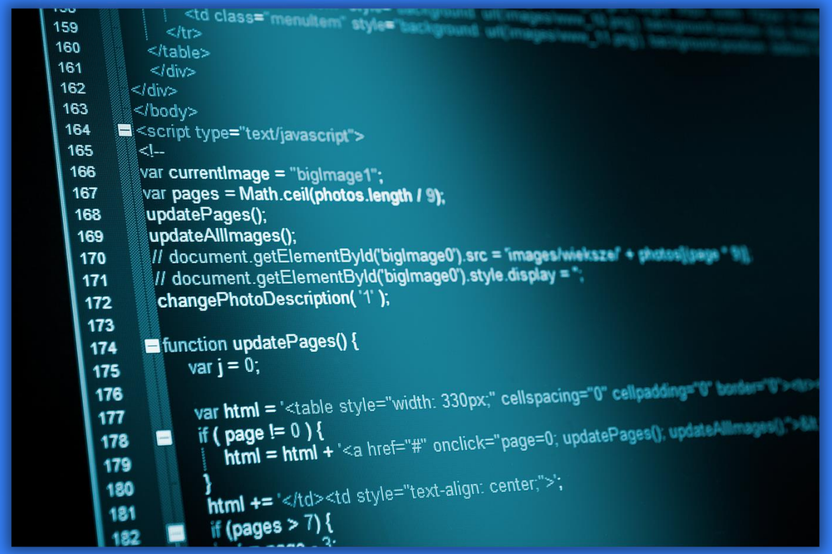
Pig Latin

Pig Example:

- Using Pig find the most occurred start letter.

Case 1: Load the data into bag named "lines". The entire line is stuck to element line of type character array.

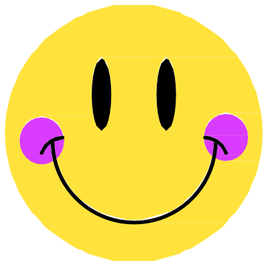
```
grunt> lines  = LOAD "/user/Desktop/data.txt" AS (line: chararray);
```



User Defined Functions(UDF’s)

User Defined Functions(UDF’s)

- Apache Pig provides extensive support for User Defined Functions (UDF’s). Using these UDF’s, we can define our own functions and use them.
- The UDF support is provided in six programming languages, namely, Java, Python, JavaScript, Ruby and Groovy.
- Complete support is provided in Java and limited support is provided in all the remaining languages.
- Using Java, you can write UDF’s involving all parts of the processing like data load/store, column transformation, and aggregation.
- Apache Pig has been written in Java, the UDF’s written using Java language work efficiently compared to other languages.



Types of UDF's in Java

1) Filter Functions:

- The filter functions are used as conditions in filter statements.
- These functions accept a Pig value as input and return a Boolean value.

2) Eval Functions:

- Writing an eval function is a small step up from writing a filter function.
- The Eval functions are used in FOREACH-GENERATE statements.
- These functions accept a Pig value as input and return a Pig result.

3) Algebraic Functions:

- The Algebraic functions act on inner bags in a FOREACHGENERATE statement.
- These functions are used to perform full MapReduce operations on an inner bag.

Data Processing operators

Data Processing operators

The Apache Pig Operators is a high-level procedural language for querying large data sets using Hadoop and the Map Reduce Platform. A Pig Latin statement is an operator that takes a relation as input and produces another relation as output.

Relational Operators:
Relational operators are the main tools Pig Latin provides to operate on the data. It allows you to transform the data by sorting, grouping, joining, projecting and filtering. This section covers the basic relational operators.

LOAD:
LOAD operator is used to load data from the file system or HDFS storage into a Pig relation.In this example, the Load operator loads data from file ‘first’ to form relation ‘loading1’. The field names are user, url, id.

FOREACH:
This operator generates data transformations based on columns of data. It is used to add or remove fields from a relation. Use FOREACH-GENERATE operation to work with columns of data.

Data Processing operators

FILTER:
This operator selects tuples from a relation based on a condition. In this example, we are filtering the record from ‘loading1’ when the condition ‘id’ is greater than 8.

JOIN:
JOIN operator is used to perform an inner, equijoin join of two or more relations based on common field values. The JOIN operator always performs an inner join. Inner joins ignore null keys, so it makes sense to filter them out before the join. I

STORE:
Store is used to save results to the file system. Here we are saving loading3 data into a file named storing on HDFS.

SPLIT:
SPLIT operator is used to partition the contents of a relation into two or more relations based on some expression. Depending on the conditions stated in the expression.

Engineering in One Video (EIOV)

Watch video on YouTube

Hive

EIOV

Engineering

IN ONE VIDEO

CONNECT, LEARN AND GROW

Engineering in One Video (EIOV)

Watch video on YouTube

Hive Concepts

Hive

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

- Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive:

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

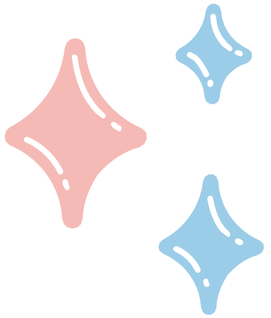
Hive

Limitations of Hive:

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Differences between Hive and Pig

- Hive is commonly used by Data Analysts.
- Pig is commonly used by programmers.



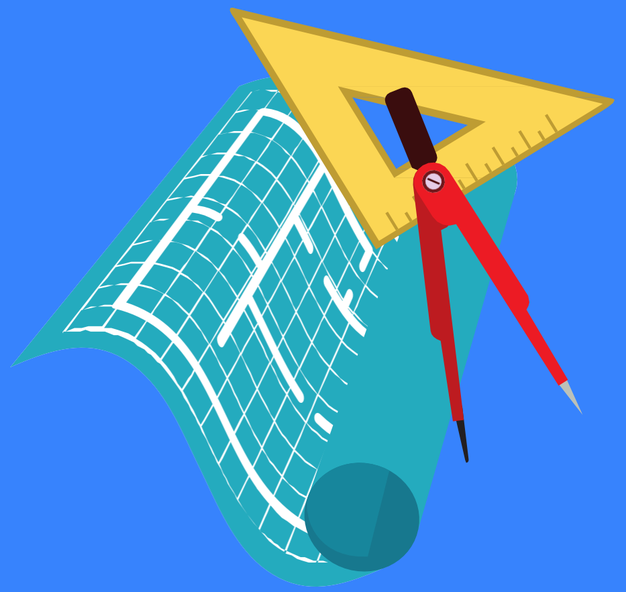
- Hive can handle structured data.
- Pig can handle semi-structured data.

- Hive works on server-side of HDFS cluster.
- Pig works on client-side of HDFS cluster.

- Hive is slower than Pig.
- Pig is comparatively faster than Hive.

Engineering in One Video (EIOV)

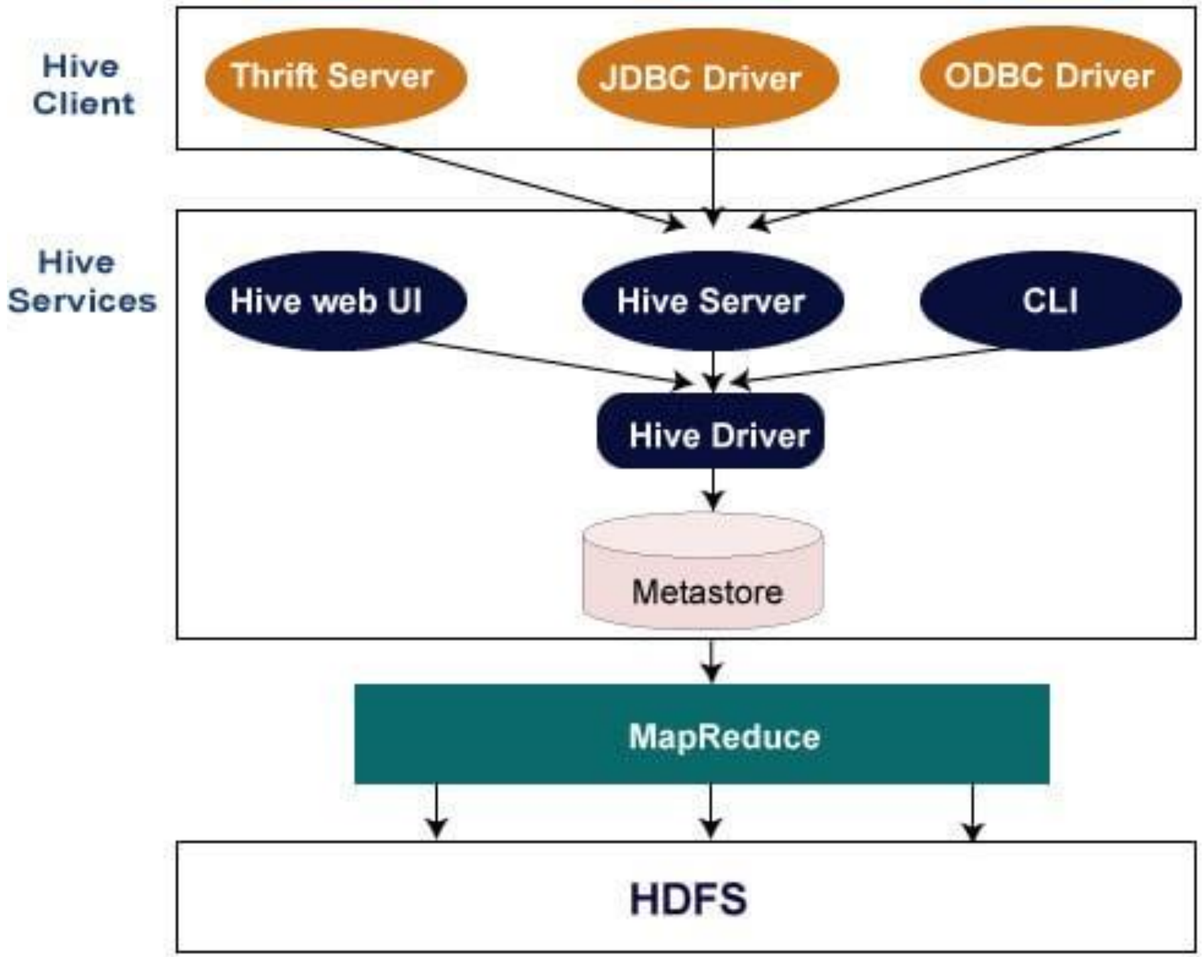
Watch video on YouTube



Hive architecture and installation

Engineering in One Video (EIOV) Watch video on YouTube

Hive architecture and installation



Hive Services

- **Hive CLI -** The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.
- **Hive Web User Interface -** The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands.
- **Hive MetaStore -** It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type information.
- **Hive Driver -** It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler.
- **Hive Compiler -** The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs.

Hive installation

Pre-requisite

- Java Installation - Check whether the Java is installed or not using the following command.

  $ java -version

- Hadoop Installation - Check whether the Hadoop is installed or not using the following command.

  $hadoop version

- Download the Apache Hive tar file.
- Unzip the downloaded tar file. (tar -xvf apache-hive-1.2.2-bin.tar.gz)
- Open the bashrc file. ($ sudo nano ~/.bashrc)
- Now, provide the following HIVE_HOME path.

  export HIVE_HOME=/home/codegyani/apache-hive-1.2.2-bin
  export PATH=$PATH:/home/codegyani/apache-hive-1.2.2-bin/bin

- Update the environment variable. ($ source ~/.bashrc)
- Let's start the hive by providing the following command.($ hive)

Engineering in One Video (EIOV)

Watch video on YouTube

Comparison with traditional databases

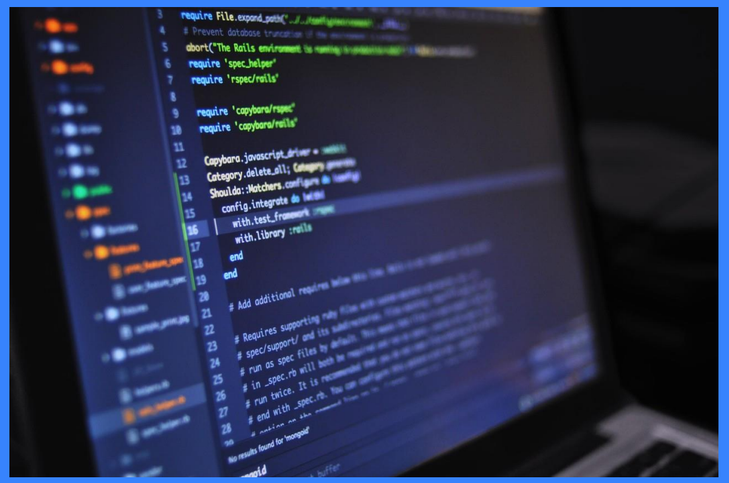
Comparison with traditional databases

	RDBMS	HIVE
Language	SQL-92 standard (maybe)	Subset of SQL-92 plus Hive-specific extension
Update Capabilities	INSERT, UPDATE and DELETE	INSERT but not UPDATE or DELETE
Transactions	Yes	No
Latency	Sub-Second	Minutes or more
Indexes	Any number of indexes, very important for performance	No indexes, data is always scanned (in parallel)
Data size	TBs	PBs
Data per query	GBs	PBs

Augment MySQL Deployments, Sarah [illegible], Cloudera, 2010

Engineering in One Video (EIOV)

Watch video on YouTube



HiveQL

Watch video on YouTube

HiveQL

- The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore.
- Hive supports 4 file formats which are: Text file, Sequence file, ORC and RC file.
- Hive supports both primitive and complex data types.
- Primitive includes numeric, boolean and string.
- Complex data types includes arrays, maps.

Engineering in One Video (EIOV)

Watch video on YouTube

Querying data and user defined functions(UDFs)

Querying data

- **SELECT** statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result.

Example:

- Let us take an example for SELECT…WHERE clause. Assume we have the employee table as given below, with fields named Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

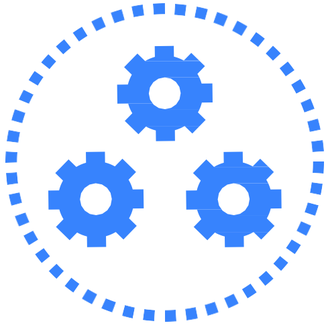
ID	Name	Salary	Designation	Dept
1201	Gopal	45000	Technical manager	TP
1202	Manisha	45000	Proofreader	PR
1203	Manas	40000	Technical writer	TP
1204	Kiran	40000	Hr Admin	HR
1205	Karan	30000	Op Admin	Admin

```
hive> SELECT * FROM employee WHERE salary>30000;
```

ID	Name	Salary	Designation	Dept
1201	Gopal	45000	Technical manager	TP
1202	Manisha	45000	Proofreader	PR
1203	Manas	40000	Technical writer	TP
1204	Kiran	40000	Hr Admin	HR

UDFs (User Defined Functions):

- In Hive, the users can define own functions to meet certain client requirements. These are known as UDFs in Hive. User Defined Functions written in Java for specific modules.
- Some of UDFs are specifically designed for the reusability of code in application frameworks.
- During the Query execution, the developer can directly use the code, and UDFs will return outputs according to the user defined tasks. It will provide high performance in terms of coding and execution.
- For example, for string stemming we don't have any predefined function in Hive, for this we can write stem UDF in Java. Wherever we require Stem functionality, we can directly call this Stem UDF in Hive.



Engineering in One Video (EIOV)

Watch video on YouTube

Sorting and aggregating

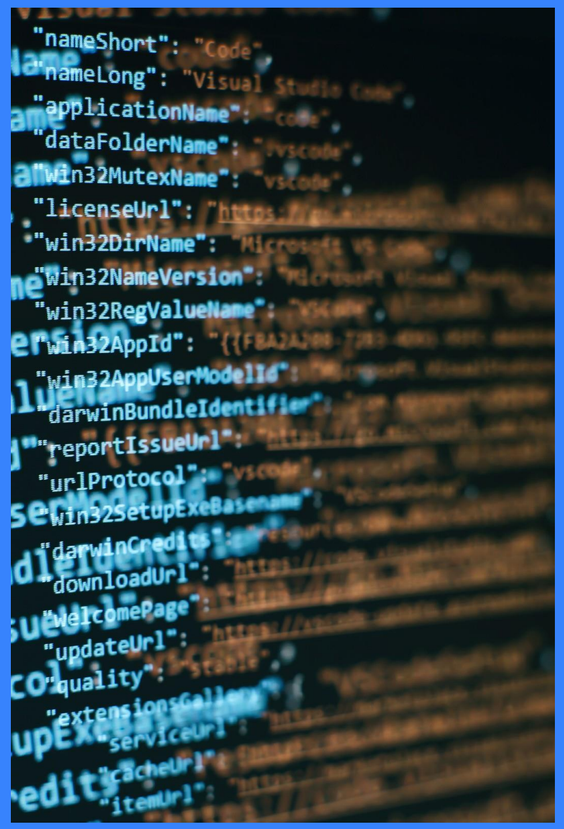
Sorting and aggregating

- Sorting data in Hive can be achieved by use of a standard ORDER BY clause, but there is a catch. ORDER BY produces a result that is totally sorted, as expected, but to do so it sets the number of reducers to one, making it very inefficient for large datasets.
- In some cases, you want to control which reducer a particular row goes to, typically so you can perform some subsequent aggregation. This is what Hive's DISTRIBUTE BY clause does. Here's an example to sort the weather dataset by year and temperature

```
hive> FROM records2
    > SELECT year, temperature
    > DISTRIBUTE BY year
    > SORT BY year ASC, temperature DESC;
1949    111
1949    78
1950    22
1950    0
1950    -11
```

Engineering in One Video (EIOV)

Watch video on YouTube

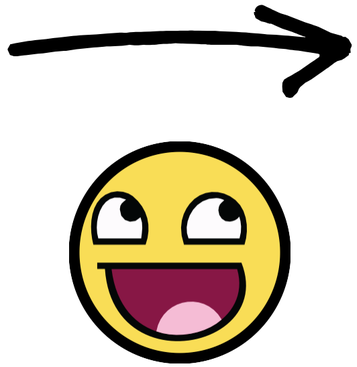


Map Reduce scripts

Map Reduce scripts

- Using an approach like Hadoop Streaming, the TRANSFORM, MAP, and REDUCE clauses make it possible to invoke an external script or program from Hive.

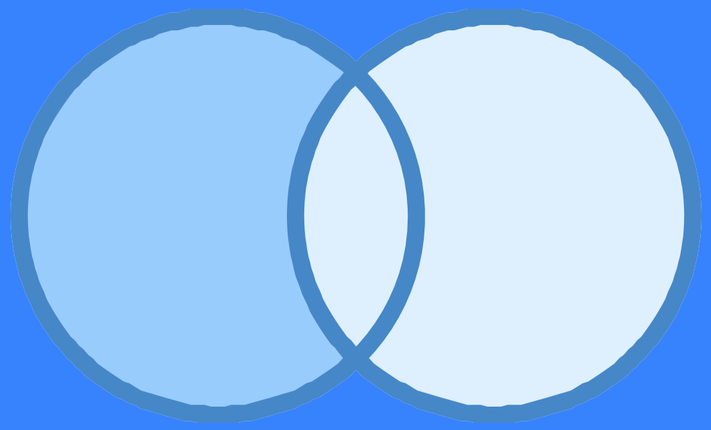
```
FROM (
  FROM records2
  MAP year, temperature, quality
  USING 'is_good_quality.py'
  AS year, temperature) map_output
REDUCE year, temperature
USING 'max_temperature_reduce.py'
AS year, temperature;
```



1949	111
1949	78
1950	22
1950	0
1950	-11

Engineering in One Video (EIOV)

Watch video on YouTube



Joins

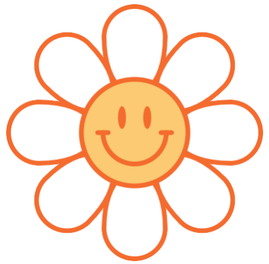
Joins

Inner joins

```
hive> SELECT * FROM sales;
Joe   2
Hank  4
Ali   0
Eve   3
Hank  2
hive> SELECT * FROM things;
2   Tie
4   Coat
3   Hat
1   Scarf
```

We can perform an inner join on the two tables as follows:

```
hive> SELECT sales.*, things.*
    > FROM sales JOIN things ON (sales.id = things.id);
Joe    2   2   Tie
Hank   2   2   Tie
Eve    3   3   Hat
Hank   4   4   Coat
```



Engineering in One Video (EIOV)

Watch video on YouTube

Joins

Outer joins

LEFT OUTER JOIN

```
hive> SELECT sales.*, things.*
    > FROM sales LEFT OUTER JOIN things
ON (sales.id = things.id);

Ali     0    NULL NULL
Joe     2    2    Tie
Hank    2    2    Tie
Eve     3    3    Hat
Hank    4    4    Coat
```

RIGHT OUTER JOIN

```
hive> SELECT sales.*, things.*
    > FROM sales RIGHT OUTER JOIN things ON
(sales.id = things.id);

NULL    NULL 1    Scarf
Joe     2    2    Tie
Hank    2    2    Tie
Eve     3    3    Hat
Hank    4    4    Coat
```

Joins

Outer joins

FULL OUTER JOIN

```
hive> SELECT sales.*, things.*
    > FROM sales FULL OUTER JOIN things ON (sales.id = things.id);
Ali     0    NULL NULL
NULL    NULL 1    Scarf
Joe     2    2    Tie
Hank    2    2    Tie
Eve     3    3    Hat
Hank    4    4    Coat
```

Engineering in One Video (EIOV)

Watch video on YouTube

Subqueries

Subqueries

- A subquery is a SELECT statement that is embedded in another SQL statement. Hive has limited support for subqueries, only permitting a subquery in the FROM clause of a SELECT statement.

```
SELECT station, year, AVG(max_temperature)

FROM (

 SELECT station, year, MAX(temperature) AS
max_temperature

 FROM records2

 WHERE temperature != 9999

  AND (quality = 0 OR quality = 1 OR quality = 4
OR quality = 5 OR quality = 9)

 GROUP BY station, year

) mt

GROUP BY station, year;
```

- A subquery is a SELECT statement that is embedded in another SQL statement. Hive has limited support for subqueries, only permitting a subquery in the FROM clause of a SELECT statement.

Engineering in One Video (EIOV)

Watch video on YouTube

HBase

Engineering in One Video (EIOV)

Watch video on YouTube

HBase concepts

EIOV

Engineering

IN ONE VIDEO

CONNECT, LEARN AND GROW

HBase concepts

- 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.

Storage Mechanism in HBase

- HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns.
- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Storage Mechanism in HBase

Given below is an example schema of table in HBase.

Rowid	Column Family			Column Family			Column Family			Column Family		
	Col1	Col2	Col3	Col1	Col2	Col3	Col1	Col2	Col3	Col1	Col2	Col3
1												
2												
3												

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

Engineering in One Video (EIOV)

Watch video on YouTube

Hbase vs RDBMS

Hbase vs RDBMS

RDBMS	HBase
It requires SQL (structured query language)	NO SQL
It has a fixed schema	No fixed schema
It is row oriented	It is column oriented
It is not scalable	It is scalable
It is static in nature	Dynamic in nature
Slower retrieval of data	Faster retrieval of data
It follows the ACID (Atomicity, Consistency, Isolation and Durability) property.	It follows CAP (Consistency, Availability, and Partition-tolerance) theorem.
It can handle structured data	It can handle structured, unstructured as well as semi-structured data
It cannot handle sparse data	It can handle sparse data

Engineering in One Video (EIOV)

Watch video on YouTube

Schema design

Schema design

- HBase table can scale to billions of rows. This table allows you to store terabytes of data in it.
- The HBase table supports the high read and write throughput at low latency.
- The HBase schema design is very different compared to the relation database schema design.
- HBase Schema Row key, Column family, Column qualifier, individual and Row value Size Limit. Consider below is the size limit when designing schema in Hbase:
- **Row keys:** 4 KB per key
- **Column families:** not more than 10 column families per table
- **Column qualifiers:** 16 KB per qualifier
- **Individual values:** less than 10 MB per cell
- **All values in a single row:** max 10 MB

Engineering in One Video (EIOV)

Watch video on YouTube

APACHE ZooKeeper™

Zookeeper

Zookeeper

- Apache Zookeeper is an open-source server that reliably coordinates distributed processes and applications.
- It allows distributed processes to coordinate with each other through a shared hierarchal namespace which is organized similarly to a standard file system.
- Apache Zookeeper provides a hierarchical file system (with ZNodes as the system files) that helps with the discovery, registration, configuration, locking,
- ZooKeeper server maintains configuration information, naming, providing distributed synchronization, and providing group services, used by distributed applications.

Resource Utilization Details:

- Zookeeper Clusters, monitor memory (heap and non-heap) on the Znode get alerts of changes in resource consumption.
- Automatically collect, graph and get alerts on garbage collection iterations, heap size and usage, threads.
- Make sure the total node count inside the ZooKeeper tree is consistent.

Engineering in One Video (EIOV)

Watch video on YouTube

IBM Big Data strategy

IBM Big Data strategy

- IBM, a US-based computer hardware and software manufacturer, had implemented a Big Data strategy.
- Where the company offered solutions to store, manage, and analyze the huge amounts of data generated daily and equipped large and small companies to make informed business decisions.
- The company believed that its Big Data and analytics products and services would help its clients become more competitive and drive growth.

Issues :

- Understand the concept of Big Data and its importance to large, medium, and small companies in the current industry scenario.
- Understand the need for implementing a Big Data strategy and the various issues and challenges associated with this.
- Analyze the Big Data strategy of IBM.
- Explore ways in which IBM’s Big Data strategy could be improved further.

Engineering in One Video (EIOV)

Watch video on YouTube

Introduction to Infosphere

Introduction to Infosphere

- InfoSphere Information Server provides a single platform for data integration and governance.
- The components in the suite combine to create a unified foundation for enterprise information architectures, capable of scaling to meet any information volume requirements.
- You can use the suite to deliver business results faster while maintaining data quality and integrity throughout your information landscape.
- InfoSphere Information Server helps your business and IT personnel collaborate to understand the meaning, structure, and content of information across a wide variety of sources.
- By using InfoSphere Information Server, your business can access and use information in new ways to drive innovation, increase operational efficiency, and lower risk.

Engineering in One Video (EIOV)

Watch video on YouTube

Big Insights and Big Sheets

Engineering in One Video (EIOV) Watch video on YouTube

Big Insights and Big Sheets

Big Insights:

- Big Insights is a software platform for discovering, analyzing, and visualizing data from disparate sources.
- The flexible platform is built on an Apache Hadoop open-source framework that runs in parallel on commonly available, low-cost hardware.

Big Sheets:

- Big Sheets is a browser-based analytic tool included in the InfoSphere BigInsights Console that you use to break large amounts of unstructured data into consumable, situation-specific business contexts.
- These deep insights help you to filter and manipulate data from sheets even further.

Engineering in One Video (EIOV)

Watch video on YouTube

Introduction to Big SQL

Engineering in One Video (EIOV) Watch video on YouTube

Introduction to Big SQL 

- IBM Big SQL is a high performance massively parallel processing (MPP) SQL engine for Hadoop that makes querying enterprise data from across the organization an easy and secure experience.
- A Big SQL query can quickly access a variety of data sources including HDFS, RDBMS, NoSQL databases, object stores, and Web HDFS by using a single database connection or single query for best-in-class analytic capabilities.

How Big SQL works:

- Big SQL's robust engine executes complex queries for relational data and Hadoop data. Big SQL provides an advanced SQL compiler and a cost-based optimizer for efficient query execution. Combining these with a massive parallel processing (MPP) engine helps distribute query execution across nodes in a cluster.

Happy Ending!

Congratulations!