1.differ between database and dataware house

**Database**

1. Used for Online Transactional Processing ([OLTP](https://en.wikipedia.org/wiki/Online_transaction_processing)) but can be used for other purposes such as Data Warehousing. This records the data from the user for history.
2. The tables and joins are complex since they are normalized (for [RDMS](https://en.wikipedia.org/wiki/Relational_database_management_system)). This is done to reduce redundant data and to save storage space.
3. Entity – Relational modeling techniques are used for RDMS database design.
4. Optimized for write operation.
5. Performance is low for analysis queries.

**Data Warehouse**

1. Used for Online Analytical Processing ([OLAP](https://en.wikipedia.org/wiki/Online_analytical_processing)). This reads the historical data for the Users for business decisions.
2. The Tables and joins are simple since they are de-normalized. This is done to reduce the response time for analytical queries.
3. Data – Modeling techniques are used for the Data Warehouse design.
4. Optimized for read operations.
5. High performance for analytical queries.
6. Is *usually* a Database.

2 what is split size and block size , its deference

block size =split size

split size Block : Hdfs talks in terms of blocks for eg : if you have file of 256 mb and you have configured your block size is 128 mb so now 2 blocks gets created for 256 mb.

Block size is configurable across the cluster and even file basis also.

Split : It has something related with map reduce , you do have an option that you can change the split size , means you can modify your split size greater than your block size or your split size less than your block size . By default if you don't do any configuration then your split size is approximately equal to block size .

In map reduce processing, number of mapper spawned would be equal to your number of splits : for a file if 10 splits are there then 10 mappers would be spawned.

|  |  |
| --- | --- |
|  | .**Difference between block size and split size.**  Split is logical split of the data, basically used during data processing using Map/Reduce program or other data processing techniques on Hadoop Ecosystem. Split size is user defined value and you can choose your own split size based on your volume of data(How much data you are processing).  Split is basically used to control number of Mapper in Map/Reduce program. If you have not defined any input split size in Map/Reduce program then default HDFS block split will be considered as input split.  **Example:**  Suppose you have a file of 100MB and HDFS default block configuration is 64MB, then it will chopped in 2 split and occupy 2 blocks. Now you have a Map/Reduce program to process this data but you have not specified any input split then based on the number of blocks(2 block) input split will be considered for the Map/Reduce processing and 2 mapper will get assigned for this job.  But suppose, you have specified the split size(say 100MB) in your Map/Reduce program then both blocks(2 block) will be considered as a single split for the Map/Reduce processing and 1 Mapper will get assigned for this job.  Suppose, you have specified the split size(say 25MB) in your Map/Reduce program then there will be 4 input split for the Map/Reduce program and 4 Mapper will get assigned for the job.  **Conclusion:**   1. Split is a logical division of the input data while block is a physical division of data. 2. HDFS default block size is default split size if input split is not specified. 3. Split is user defined and user can control split size in his Map/Reduce program. 4. One split can be mapping to multiple blocks and there can be multiple split of one block. 5. The number of map tasks (Mapper) are equal to the number of splits. |

3. how record reader read the data

To understand record reader in Hadoop, we need to understand the Hadoop data flow. So, Let’s see how the data flow in Hadoop?

**MapReduce** has a simple model of data processing. Inputs and Outputs for the map and reduce functions are [**key-value pairs.**](http://data-flair.training/blogs/key-value-pairs-hadoop-mapreduce/) The map and reduce functions in Hadoop MapReduce have the following general form:

* **map:** (K1, V1) → list(K2, V2)
* **reduce:** (K2, list(V2)) → list(K3, V3)

Now before processing, it needs to know on which data to process, this is achieved with the [**InputFormat**](http://data-flair.training/blogs/hadoop-inputformat-types/) class. InputFormat is the class which selects the file from[**HDFS**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/) that should be input to the map function. An InputFormat is also responsible for creating the [**InputSplits**](http://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/) and dividing them into records. The data is divided into the number of splits(typically 64/128mb) in HDFS. This is called as inputsplit which is the input that is processed by a single map.

InputFormat class calls the **getSplits()** function and computes splits for each file and then sends them to the **JobTracker**, which uses their storage locations to schedule map tasks to process them on the **TaskTrackers**. Map task then passes the split to the **createRecordReader()** method on InputFormat in task tracker to obtain a RecordReader for that split. The RecordReader load’s data from its source and converts into key-value pairs suitable for reading by the[**mapper.**](http://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)

Hadoop RecordReader uses the data within the boundaries that are being created by the inputsplit and creates Key-value pairs for the mapper. The “start” is the byte position in the file where the RecordReader should start generating key/value pairs and the “end” is where it should stop reading records. In Hadoop RecordReader, the data is loaded from its source and then the data is converted into key-value pairs suitable for reading by the Mapper. It communicates with the inputsplit until the file reading is not completed.

4.diffencebetween managed table and external table

Hive has a relational database on the master node it uses to keep track of state. For instance, when you CREATE TABLE FOO(foo string) LOCATION 'hdfs://tmp/';, this table schema is stored in the database.

If you have a partitioned table, the partitions are stored in the database(this allows hive to use lists of partitions without going to the file-system and finding them, etc). These sorts of things are the 'metadata'.

When you drop an internal table, it drops the data, and it also drops the metadata.

When you drop an external table, it only drops the meta data. That means hive is ignorant of that data now. It does not touch the data itself.

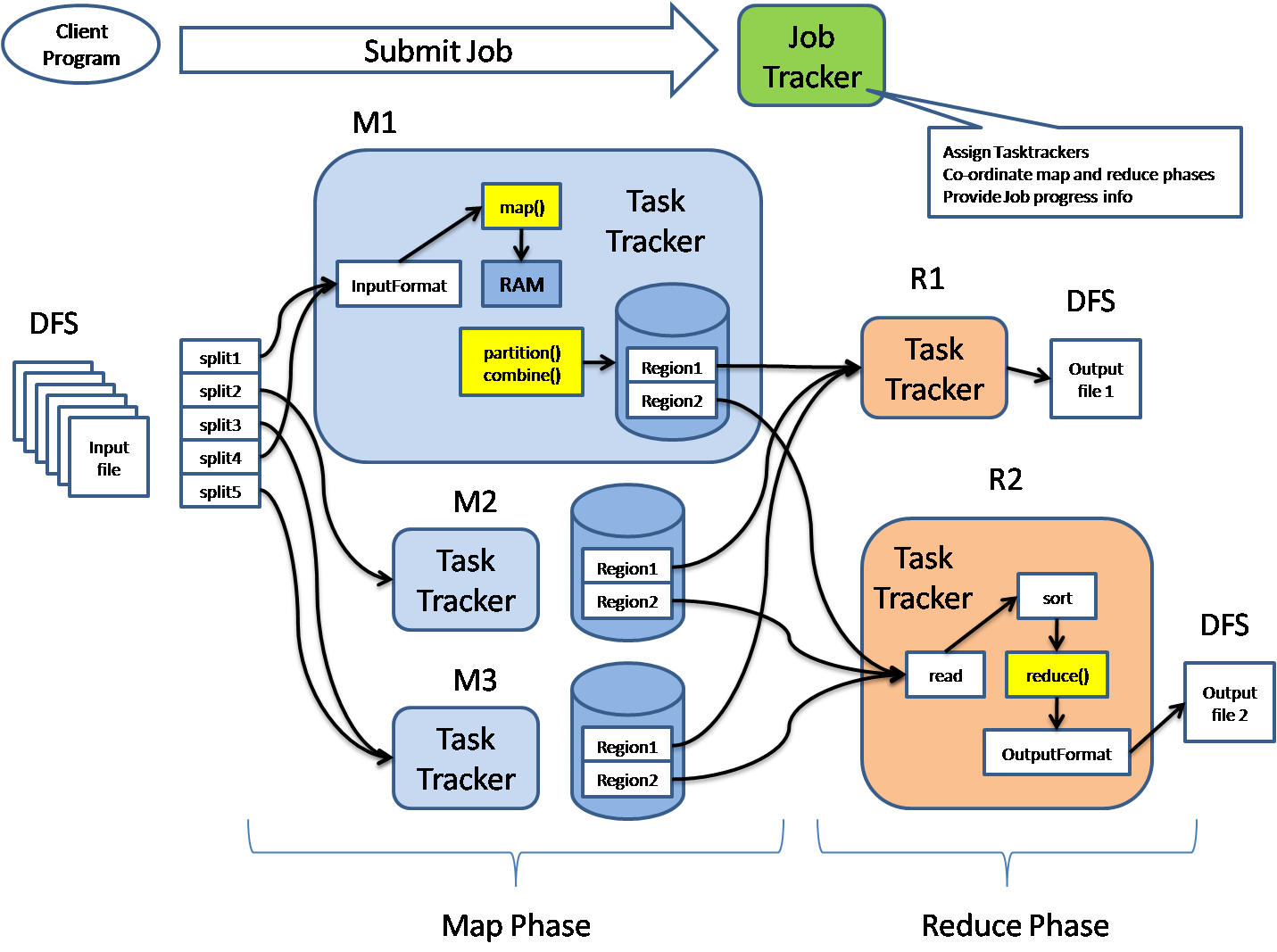
4.How to store data at particular location in Hive

with LOCATION key word we can store any place in HDFS

but we cannot store locally

5. what is the partitions Hive Partitions is a way to organizes tables into partitions by dividing tables into different parts based on partition keys.

Partition is helpful when the table has one or more Partition keys. Partition keys are basic elements for determining how the data is stored in the table. two types static and dynamic

6.mapreduce architecture 

8.sqoop statements :import and export

9.how to read the last update record from the tables;

SELECT MAX (document\_number) FROM <table>

sort itab by field1 desending.

1st record is ur latest record.

SELECT TOP 1 \* FROM Table ORDER BY ID DESC

**MySql**

SELECT \* FROM Table ORDER BY ID DESC LIMIT 1

**Differences between Pig and Hive-**Depending on the purpose and type of data you can either choose to use Hive Hadoop component or Pig Hadoop Component based on the below differences :

1) Hive Hadoop Component is used mainly by data analysts whereas Pig Hadoop Component is generally used by Researchers and Programmers.

2) Hive Hadoop Component is used for completely structured Data whereas Pig Hadoop Component is used for semi structured data.

3) Hive Hadoop Component has a declarative SQL language (HiveQL) whereas Pig Hadoop Component has a procedural data flow language (Pig Latin)

4) Hive Hadoop Component is mainly used for creating reports whereas Pig Hadoop Component is mainly used for programming.

5) Hive Hadoop Component operates on the server side of any cluster whereas Pig Hadoop Component operates on the client side of any cluster.

6) Hive Hadoop Component is helpful for ETL whereas Pig Hadoop is a great [ETL tool for big data](http://www.dezyre.com/article/5-reasons-why-etl-professionals-should-learn-hadoop/74) because of its powerful transformation and processing capabilities.

7) Hive can start an optional thrift based server that can send queries from any nook and corner directly to the Hive server which will execute them whereas this feature is not available with Pig.

8) Hive directly leverages SQL expertise and thus can be learnt easily whereas Pig is also SQL-like but varies to a great extent and thus it will take some time efforts to master Pig.

9) Hive makes use of exact variation of the SQL DLL language by defining the tables beforehand and storing the schema details in any local database whereas in case of Pig there is no dedicated metadata database and the schemas or data types will be defined in the script itself.

10) The Hive Hadoop component has a provision for partitions so that you can process the subset of data by date or in an alphabetical order whereas Pig Hadoop component does not have any notion for partitions though might be one can achieve this through filters.

11) Pig supports Avro whereas Hive does not.

12) Pig can be installed easily over Hive as it is completely based on shell interaction

13) Pig Hadoop Component renders users with sample data for each scenario and each step through its “Illustrate” function whereas this feature is not incorporated with the Hive Hadoop Component.

14) Hive has smart inbuilt features on accessing raw data but in case of Pig Latin Scripts we are not pretty sure that accessing raw data is as fast as with HiveQL.

15) You can join, order and sort data dynamically in an aggregated manner with Hive and Pig however Pig also provides you an additional COGROUP feature for performing outer joins.

There is no battle between HIVE and PIG in the real world. They don’t have the same advantages and disadvantages while processing enormous amounts of data. It’s just the initial ambiguity on deciding the tool which suits the need. HIVE Query language (HiveQL) suits the specific demands of analytics meanwhile PIG supports huge data operation. PIG was developed as an abstraction to avoid the complicated syntax of Java programming for MapReduce. On the other hand HIVE QL is based around SQL, which makes it easier to learn for those who know SQL. AVRO is supported by PIG making serialization faster. When it really boils down on taking decision between Pig and Hive, the suitability of the each component for the given business logic must be considered and then the decision must be taken.

**Conclusion:**To conclude with after having understood the difference between Pig and Hive, both Hive Hadoop and Pig Hadoop Component will help to achieve the same goals, we can say that Pig is a script kiddy and Hive comes in, innate for all the natural database developers. When it comes to access choices, Hive is said to have more features over Pig. Both the Hive and Pig components are reportedly having near about the same number of committers in every project and likely in the near future we are going to see great advancements in both on the development front.