**What is HIVE**

* + Data warehouse system for hadoop
  + Run SQL-like queries that get compiled and run as Map Reduce jobs.
  + Displays the result back to the user
  + Data in hadoop even though generally unstructured has some vague structure associated with it

**DatawareHouse:**

A database designed to handle transactions isn’t designed to handle analytics. It isn’t structured to do analytics well. A data warehouse, on the other hand, is structured to make analytics fast and easy.

Database is perform an OLTP (online transaction processing)model where as Datawarehouse on OLAP(online analytical processing).

To effectively perform analytics, you need a data warehouse. A data warehouse is a database of a different kind: an OLAP (online analytical processing) database. A data warehouse exists as a layer on top of another database or databases (usually OLTP databases). The data warehouse takes the data from all these databases and creates a layer optimized for and dedicated to analytics.

Data warehouses are potentially much bigger than the databases from where the data is derived. Databases usually store only the data that is currently in active use; older records can be purged and moved to backups, mainly for performance reasons.

**Components:**

Metastore :

* + Hive stores the schema of the Hive tables in a Hive Metastore. Metastore is used to hold all the information about the tables and partitions that are in the warehouse. By default, the metastore is run in the same process as the Hive service and the default Metastore is DerBy Database.

SerDe :

* Serializer, Deserializer gives instructions to hive on how to process a record.

**Why HIVE**

* + To ease the use of hadoop file system and Map Reduce for non-developers. Users like scientist, analysts etc just need to know SQL syntax
  + Writing SQL is faster than writing code

**HIVE DataType**

Hive supports all the common primitive data formats such as BIGINT, BINARY, BOOLEAN, CHAR, DECIMAL, DOUBLE, FLOAT, INT, SMALLINT, STRING, TIMESTAMP, and TINYINT.

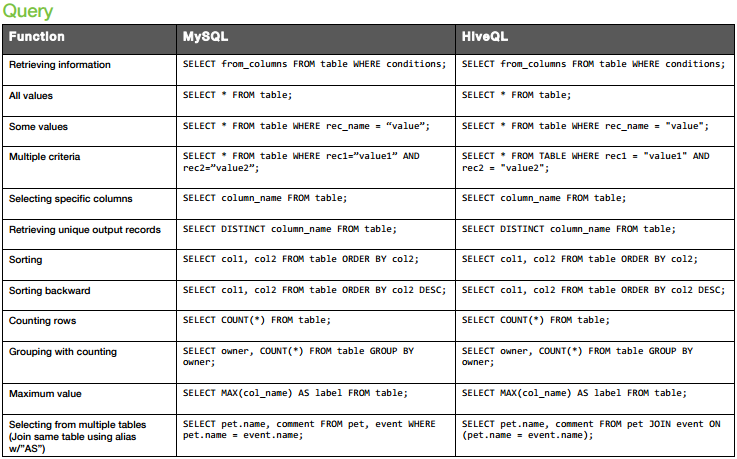
**RDBMS VS HIVE**

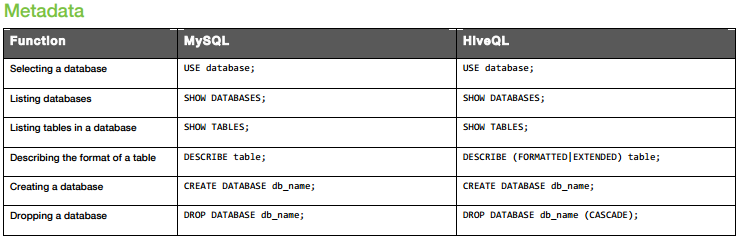
* In RDBMS, a table's schema is enforced at data load time, if the data being loaded does not conform to the schema then it is rejected. This design is called schema on write. But in Hive does not verify the data when it is loaded, but rather when it's retrieved. This is called schema on read. So while loading, HIVE can be very fast.
* In RDBMS, record level updates, inserts and deletion happens. But in HIVE it's not possible till version 0.14.
* In RDBMS, maximum data size allowed will be in 1-7 TB data whereas HIVE can accommodate 100's petabytes very easily.
* RDBMS, supports OLTP model (Online Transaction Processing) but HIVE is NOT and HIVE Suits for OLAP model (Online Analytical process) and best suits for BATCH processing.
* To overcome the limitations of HIVE, HBASE is being integrated with HIVE to support record level operations.
* Hive is very easily scalable at low cost but RDBMS is not that much scalable that too. It is very costly scale up.

**What HIVE is NOT?**

* + It is not RDBMS
  + OLTP workloads- low latency
  + Correlated subqueries
  + Even with small amount of data time to return the response can’t be compared to RDBMs

**Hive for SQL Users (Cheat Sheet)**





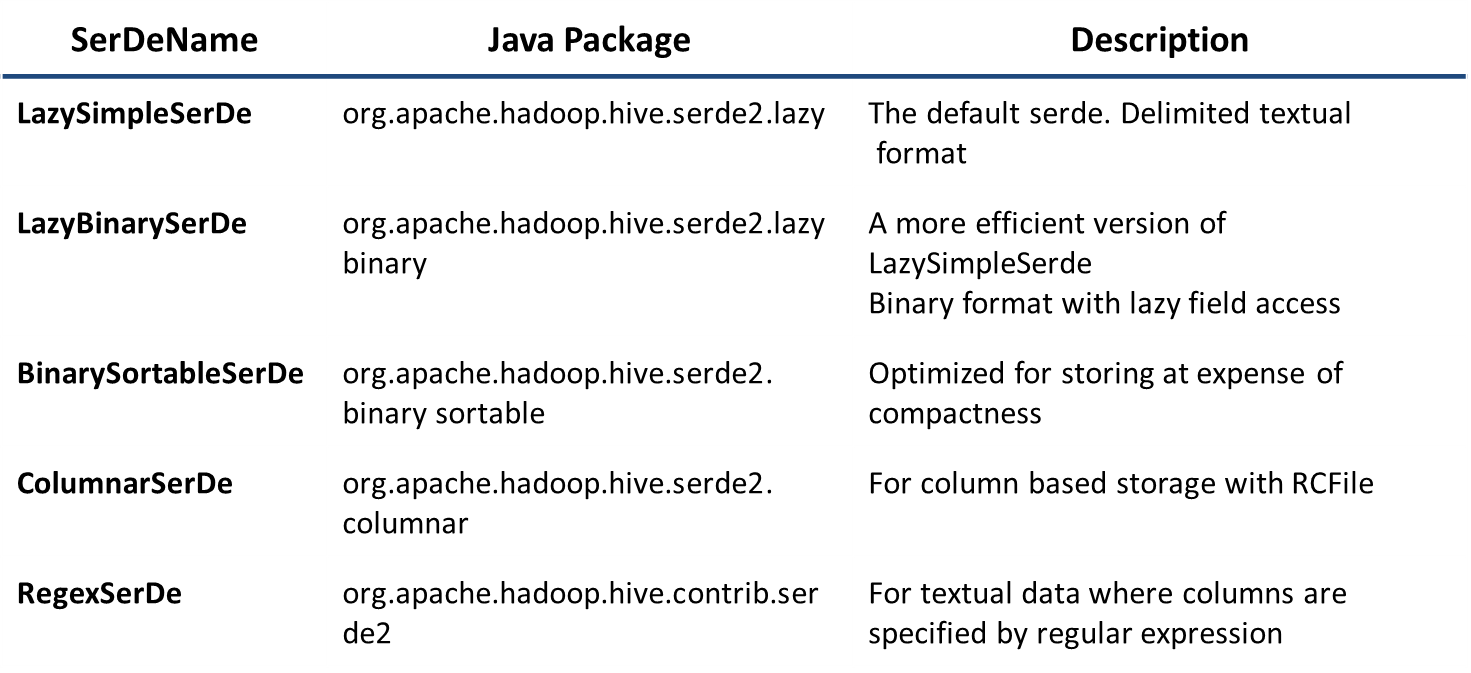
**Limitations of HIVE:**

* You cannot update data after it is inserted
* There is no "insert into table values ... " statement
* You can only load data using bulk load
* There is not "delete from " command
* You can only do bulk delete
* Correlated sub-queries are not allowed.

**SerDe**

* + Serde (Serializer/Deserializer) is used by Hive to control how lines are read and written to from files
  + When used as serializer (i.e insert) table’s serde will serialize hive’s internal representation of row of data into bytes written to output file
  + When used as deserializer (i.e querying the data), serde will deserialize a row of data from bytes in file to objects

**HIVE SerDe**



**HIVE Default Storage Format**

* + Default format is delimited text, with a row per line.
  + Rows in a table are delimited by newline character.
  + Hive uses LazySimpleSerde for this format

**HIVE Tables**

**Managed table (Internal Table)**

* By default hive create internal tables and store the data in warehouse.
* On delete both metadata and data will be deleted.

**External table**

* If the user wants to change the default directory to some specific existing directory in HDFS, following command can be used. Here only schema will be store and data reside outside.
* On delete only metadata will be deleted.

**HIVE partitions:**

* + Partitioning a data set means dividing and splitting the data into smaller partitions using values of columns.
  + Hive partitions are stored in subdirectories of table directory
  + Allows hive to filter data at input path
  + Example every day’s files can be stored for each day in a directory based   
    on dates.

**Usecase:**

Partition divides large amount of data into multiple slices based on value of a table column(s).

Assume that you are storing information of people in entire world spread across 196+ countries spanning around 500 crores of entries. If you want to query people from a particular country (raj city), in absence of partitioning, you have to scan all 500 crores of entries even to fetch thousand entries of a country. If you partition the table based on country, you can fine tune querying process by just checking the data for only one country partition. Hive partition creates a separate directory for a column(s) value.

1. Distribute execution load horizontally
2. Faster execution of queries in case of partition with low volume of data. e.g. Get the population from "raj city" returns very fast instead of searching entire population of world.

**Types of Partition:**

* Static Partition
* Dynamic Partition

**Bucketing:**

Assuming the above partition scenario when partitioning our tables based geographic location like country the some bigger countries will have large partitions (70-80 % of data) whereas smaller countries data will create small partitions (20-30 % data). So, in these cases Partitioning will not be ideal.

To overcome the problem of over partitioning, HIVE provides Bucketing concept, another technique for decomposing table data sets into more manageable parts.