**1. When to use internal and when to use external tables in hive?**

When there is data already in HDFS, an external Hive table can be created to describe the data. It is called EXTERNAL because the data in the external table is specified in the LOCATION properties instead of the default warehouse directory.

When keeping data in the internal tables, Hive fully manages the life cycle of the table and data. This means the data is removed once the internal table is dropped. If the external table is dropped, the table metadata is deleted but the data is kept. Most of the time, an external table is preferred to avoid deleting data along with tables by mistake

INTERNAL: Table is created First and Data is loaded later, supports Truncate and ACID properties.

EXTERNAL: Data is present and Table is created on top of it. Truncate and ACID properties not supported.

Internal tables are used when the data needs to be updated or some rows need to be deleted because ACID properties can be supported on the Internal tables but ACID properties cannot be supported on the external tables.

**2. Joins in hive, Map Side join in hive?**

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins>

<https://acadgild.com/blog/join-in-hive-optimization>

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database.

JOIN - Like SQL Inner Join

LEFT OUTER JOIN - All rec from left table and matching rec from right table

RIGHT OUTER JOIN - All rec from right table and matching rec from left table

FULL OUTER JOIN - Matching and Non-matching records from both tables.

Complex expressions in ON clause are supported, starting with Hive 2.2.0 (see [HIVE-15211](https://issues.apache.org/jira/browse/HIVE-15211), [HIVE-15251](https://issues.apache.org/jira/browse/HIVE-15251)). Prior to that, Hive did not support join conditions that are not equality conditions.

Hive converts joins over multiple tables into a single map/reduce job if for every table the same column is used in the join clauses

* In every map/reduce stage of the join, the last table in the sequence is streamed through the reducers where as the others are buffered. Therefore, it helps to reduce the memory needed in the reducer for buffering the rows for a particular value of the join key by organizing the tables such that the largest tables appear last in the sequence. e.g. in

|  |
| --- |
| SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key1) |

* all the three tables are joined in a single map/reduce job and the values for a particular value of the key for tables a and b are buffered in the memory in the reducers. Then for each row retrieved from c, the join is computed with the buffered rows. Similarly for

|  |
| --- |
| SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2) |

* there are two map/reduce jobs involved in computing the join. The first of these joins a with b and buffers the values of a while streaming the values of b in the reducers. The second of one of these jobs buffers the results of the first join while streaming the values of c through the reducers.
* In every map/reduce stage of the join, the table to be streamed can be specified via a hint. e.g. in

|  |
| --- |
| SELECT /\*+ STREAMTABLE(a) \*/ a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key1) |

* all the three tables are joined in a single map/reduce job and the values for a particular value of the key for tables b and c are buffered in the memory in the reducers. Then for each row retrieved from a, the join is computed with the buffered rows. If the STREAMTABLE hint is omitted, Hive streams the rightmost table in the join.

**3. Difference between parquet, ORC and AVRO. How ORC file organizes data?**

<https://www.datanami.com/2018/05/16/big-data-file-formats-demystified/#:~:text=The%20biggest%20difference%20between%20ORC,in%20a%20row%2Dbased%20format.&text=While%20column%2Doriented%20stores%20like,might%20be%20the%20better%20choice.>

1) AVRO:-

* It is row major format.
* Its primary design goal was schema evolution.
* In the avro format, we store schema separately from data. Generally avro schema file (.avsc) is maintained.

2) ORC

* Column oriented storage format.
* Originally it is Hive's Row Columnar file. Now improved as Optimized RC (ORC)
* Schema is with the data, but as a part of footer.
* Data is stored as row groups and stripes.
* Each stripe maintains indexes and stats about data it stores.

3) Parquet

* Similar to ORC. Based on google dremel
* Schema stored in footer
* Column oriented storage format
* Has integrated compression and indexes

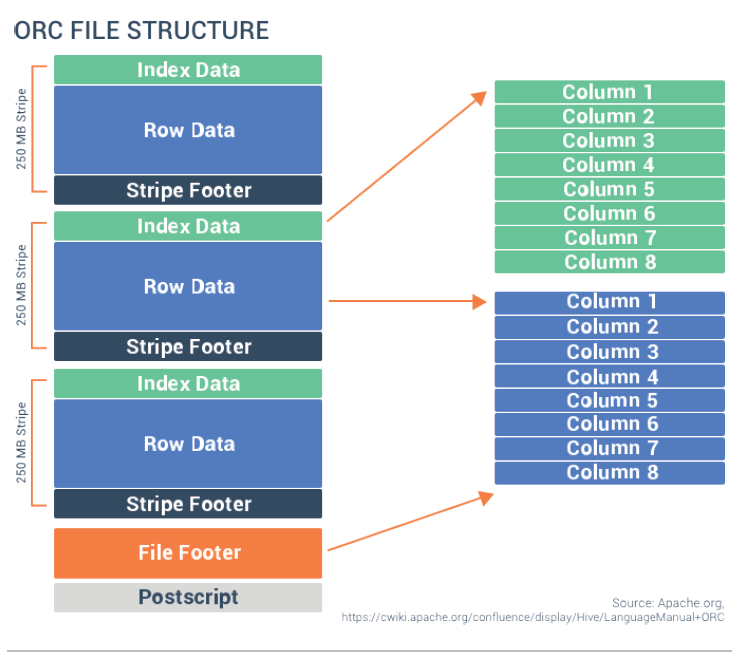
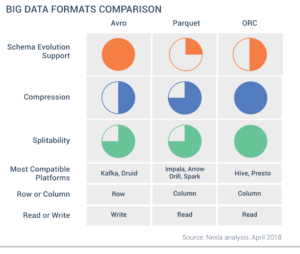
Space or compression wise I found them pretty close to each other

Around 10 GB of CSV data compressed to 1.1 GB of ORC with ZLIB compression and same data to 1.2 GB of Parquet GZIP. Both file formats with SNAPPY compression, used around 1.6 GB of space. Conversion speed wise ORC was little better it took 9 min whereas parquet took 10 plus min.

Snappy Compression + Parquet + Spark works best - as Spark has a vectorized reader for Parquet format.

Zlib Compression + ORC + Hive works best - as hive has a vectorized reader for ORC.

SNAPPY for time based performance, ZLIB for resource performance (Drive Space)

 [](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2018/05/Nexla-File-Format.png)

**4. Best file format to store data if we have a simple select query?**

<https://dwgeek.com/best-practices-to-optimize-hive-query-performance.html/>

**5. Optimization techniques in hive?**

<https://www.qubole.com/blog/5-tips-for-efficient-hive-queries/>

**6. What does post script holds in hive?**

**Postscript** - holds compression parameters and the size of the compressed footer.

<https://stackoverflow.com/questions/48860476/can-someone-explain-me-the-output-of-orcfiledump#:~:text=Postscript%20%2D%20holds%20compression%20parameters%20and,min%2C%20max%2C%20and%20sum.>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC>

**7. Tez Architecture.**

<https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez#:~:text=Tez%20is%20a%20new%20application,of%20the%20map%2Dreduce%20framework.>

<https://stackoverflow.com/questions/25521363/apache-tez-architecture-explanation>

**8. What is Tez execution engine, which engine you should use for hive?**

Chooses execution engine. Options are: mr (Map Reduce, default), tez ([Tez](https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez) execution, for Hadoop 2 only), or spark ([Spark](https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark) execution, for Hive 1.1.0 onward).

While mr remains the default engine for historical reasons, it is itself a historical engine and is deprecated in the Hive 2 line ([HIVE-12300](https://issues.apache.org/jira/browse/HIVE-12300)).

<https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#:~:text=Chooses%20execution%20engine.,line%20(HIVE%2D12300).>

<https://sites.google.com/site/hadoopbigdataoverview/hive/hive-optimization-techniques-1/hive-optimization-techniques>

<http://hadooptutorial.info/hive-on-tez/>

**9. What is the difference between Tez and MR execution engine?**

<https://community.cloudera.com/t5/Support-Questions/Difference-between-mr-and-Tez/td-p/103672>

Tez simplifies processing for both small scale (low-latency) and large-scale (high throughput) workloads. The more complex query is the more benefit from TEZ. For simple queries consisting of single map step it will be most probably no difference at all because there is nothing to optimize. TEZ represents query as a DAG (directed acyclic graph) for a single job and eliminates unnecessary steps like read/write to durable storage, sort of the output from each Map, also enables containers reuse. Tez is always the best choice, for simple queries it will work not worse than MR and much better for complex queries. And consider this: For MR and for TEZ you have to tune different sets of configuration parameters, there are a lot of TEZ-specific and a lot of MR-specific. Choose TEZ and you will simplify your life even in cases when there is nothing to optimize.

**10. Yarn Architecture?**

<https://www.edureka.co/blog/hadoop-yarn-tutorial/>

<https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>

**11. Input split vs Block.**

<https://www.edureka.co/community/5629/inputsplit-vs-hdfs-block>

<https://techvidvan.com/tutorials/hadoop-inputsplit-vs-blocks/>

<https://data-flair.training/blogs/mapreduce-inputsplit-vs-block-hadoop/>

**12. What is skew data and how we handle it in hive?**

<https://www.quora.com/What-is-a-skewed-table-in-Hive#:~:text=Basically%2C%20we%20use%20Hive%20skew,table%20compared%20to%20other%20data.>

<https://stackoverflow.com/questions/36147699/how-to-process-skewed-data-using-hive>

<https://data-flair.training/blogs/skew-join-in-hive/>

<https://medium.com/expedia-group-tech/skew-join-optimization-in-hive-b66a1f4cc6ba>

<https://cwiki.apache.org/confluence/display/Hive/Skewed+Join+Optimization>