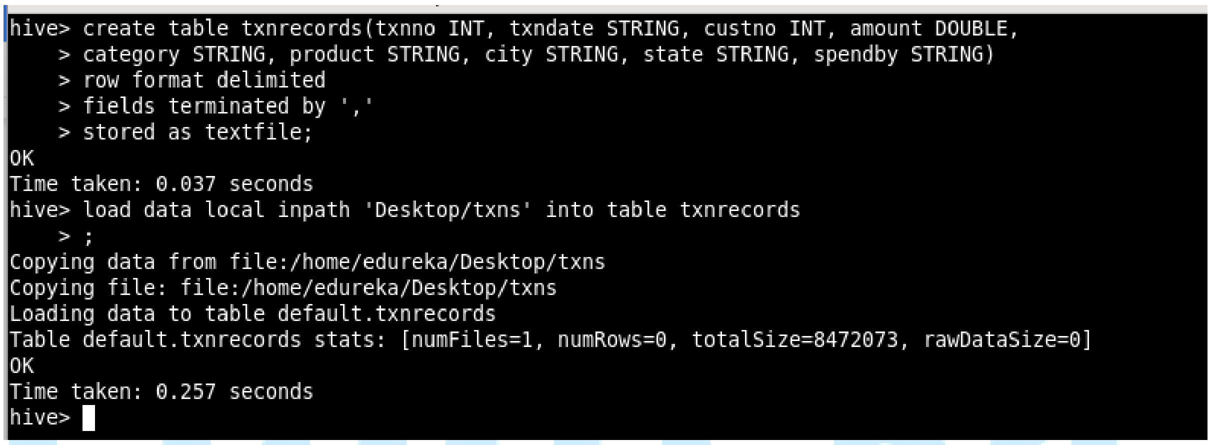
Module 6: HIVE

Hive Views and Indexes

Hive Index

One of the Hive query optimization method is Hive Index. Hive index are used to speed up the access of column or set of columns in Hive database. Without an index, the database system has to read all rows in the table to find the data you have selected.

→ Hive Index are available from Hive version 0.7.
→ Maintaining an index requires extra disk space and building an index has a processing cost
→ Hive Index works for Both Internal and external table and Partition table.



```
hive> create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE,
    > category STRING, product STRING, city STRING, state STRING, spendby STRING)
    > row format delimited
    > fields terminated by ','
    > stored as textfile;
OK
Time taken: 0.037 seconds
hive> load data local inpath 'Desktop/txns' into table txnrecords
    > ;
Copying data from file:/home/edureka/Desktop/txns
Copying file: file:/home/edureka/Desktop/txns
Loading data to table default.txnrecords
Table default.txnrecords stats: [numFiles=1, numRows=0, totalSize=8472073, rawDataSize=0]
OK
Time taken: 0.257 seconds
hive>
```

Syntax for creating Index:

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
WITH DEFERRED REBUILD
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
  [ ROW FORMAT ...] STORED AS ...
  | STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
[COMMENT "index comment"]
```

NOTE : Square brackets represent the optional commands.

index.handler.class.name : Index handlers are used to stored the base table index in a particular data structure. We can implement the custom index handler using Java classes.
First version Index handler is
org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler.
Hive also offers a bitmap index handler as of the 0.8 release, which is intended for creating indexes on columns with a few unique values.

Index handler responsibilities:

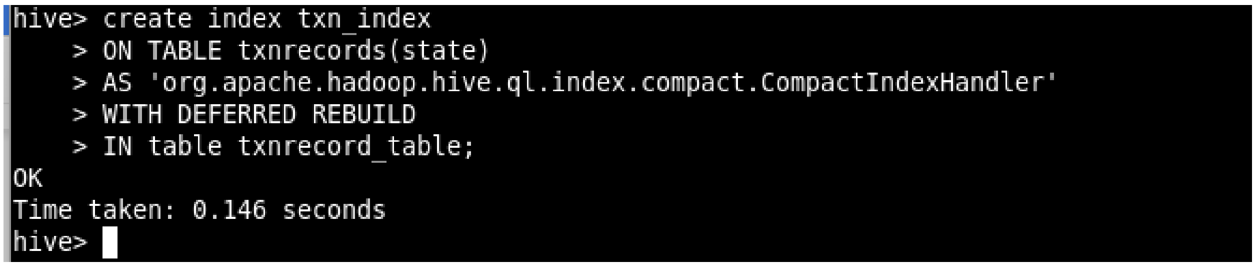
- ➔ During CREATE INDEX, validating the format of the base table and then generating the structure of the index table (if any) and filling any additional information into the index's storage descriptor.
- ➔ During REBUILD, producing a plan for reading the base table's data and writing to the index storage and/or index table.
- ➔ During DROP, deleting any index-specific storage (index tables are dropped automatically by Hive).
- ➔ During queries, participating in optimization in order to convert operators such as filters into index access plans (this part is out of scope for the moment).

WITH DEFERRED REBUILD: instructs Hive to first create an empty index table. Then we have to use the ALTER INDEX ….. REBUILD command used to build the index structure.

The index data for a table is stored in another table. This is why you need to first create the index table and then build it to populate the table using **WITH DEFERRED REBUILD.**

If you not specify the [IN TABLE index_table_name] The indexes are stored in default index table. The Index table name is **default__<table name>_<index name>__**

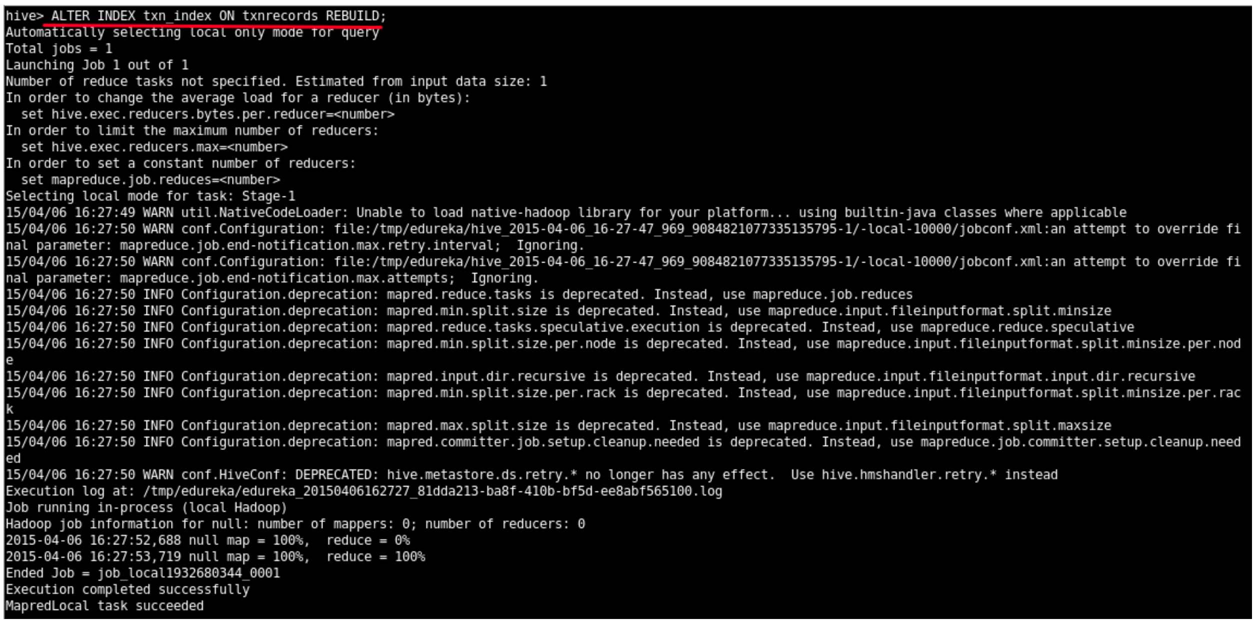
The index table contain Three type of columns.
Base table column which used to for indexing.
_bucketname column which is the location of the data in the Hive warehouse.
_offsets column type array, which contain the index of the base table.



```
hive> create index txn_index
    > ON TABLE txnrecords(state)
    > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
    > WITH DEFERRED REBUILD
    > IN table txnrecord_table;
OK
Time taken: 0.146 seconds
hive> 
```

Once empty index table is created you have alter the empty index table that created first time.If the data in the base table is changed then ALTER … REBUILD command must be used to bring the index up to date.

Command : ALTER INDEX index_name ON table_name REBUILD;

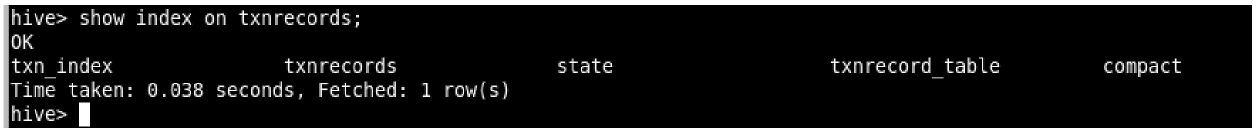


```
hive> ALTER INDEX txn_index ON txnrecords REBUILD;
Automatically selecting local only mode for query
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-1
15/04/06 16:27:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
15/04/06 16:27:50 WARN conf.Configuration: file:/tmp/edureka/hive_2015-04-06_16-27-47_969_9084821077335135795-1/-local-10000/jobconf.xml:an attempt to override fi
nal parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
15/04/06 16:27:50 WARN conf.Configuration: file:/tmp/edureka/hive_2015-04-06_16-27-47_969_9084821077335135795-1/-local-10000/jobconf.xml:an attempt to override fi
nal parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.min.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.split.minsize
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.reduce.speculative
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.min.split.size.per.node is deprecated. Instead, use mapreduce.input.fileinputformat.split.minsize.per.nod
e
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat.input.dir.recursive
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.min.split.size.per.rack is deprecated. Instead, use mapreduce.input.fileinputformat.split.minsize.per.rac
k
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.max.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.split.maxsize
15/04/06 16:27:50 INFO Configuration.deprecation: mapred.committer.job.setup.cleanup.needed is deprecated. Instead, use mapreduce.job.committer.setup.cleanup.need
ed
15/04/06 16:27:50 WARN conf.HiveConf: DEPRECATED: hive.metastore.ds.retry.* no longer has any effect.  Use hive.hmshandler.retry.* instead
Execution log at: /tmp/edureka/edureka_20150406162727_81dda213-ba8f-410b-bf5d-ee8abf565100.log
Job running in-process (local Hadoop)
Hadoop job information for null: number of mappers: 0; number of reducers: 0
2015-04-06 16:27:52,688 null map = 100%,  reduce = 0%
2015-04-06 16:27:53,719 null map = 100%,  reduce = 100%
Ended Job = job_local1932680344_0001
Execution completed successfully
MapredLocal task succeeded
```

Showing an Index :

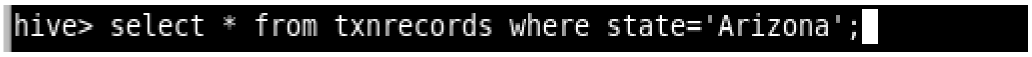
Command : SHOW INDEX ON base_table_name;

This command display the index name, base table, index column, Index table, Storage handler.



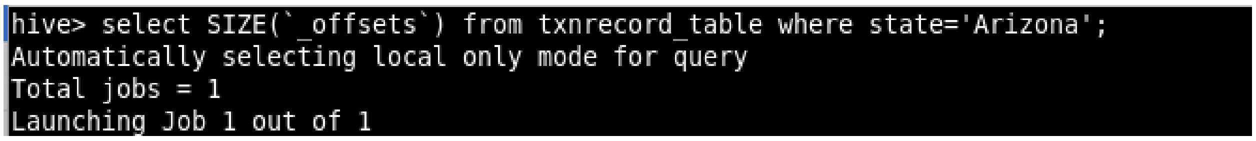
```
hive> show index on txnrecords;
OK
txn_index               txnrecords              state                   txnrecord_table         compact
Time taken: 0.038 seconds, Fetched: 1 row(s)
hive> 
```

Querying on Index table :



```
hive> select * from txnrecords where state='Arizona';
```

You can write the query on index base table using index column or you can write query on index table. In this example index table is txnrecord_table.

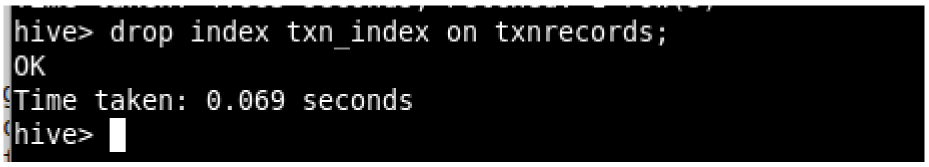


```
hive> select SIZE(`_offsets`) from txnrecord_table where state='Arizona';
Automatically selecting local only mode for query
Total jobs = 1
Launching Job 1 out of 1
```

DROPPING index :

Command **:** DROP INDEX index_name ON base_table_name;

You can drop the index table without dropping the base table.



```
hive> drop index txn_index on txnrecords;
OK
Time taken: 0.069 seconds
hive> 
```

Views in Hive

A view allows a query to be saved and treated like a table. It is a logical construct, as it
does not store data like a table.Views are support from hive 0.6 version and later.

Views are kind of virtual tables, You can apply all the DML operation.

Views are used to :

1.Views Reduce Query Complexity. Views encapsulate the complexity so it is very easy to understand for the end user.

For example If nested query contain JOINING statements and query data from JOINING statement, Instead of writing nested query divided the query to small parts.

FROM (
SELECT * FROM txnrecords JOIN customers
ON (txnrecords.txnno=customers.txnno) WHERE state='Arizona') a SELECT a.firstname a.lastname WHERE a.txnno=4002310;

This complex query we can convert into smaller queries using views.

CREATE VIEW records AS
SELECT * FROM txnrecords JOIN customers
ON (txnrecords.txnno=customers.txnno) WHERE state='Arizona'

Now qeury the data using view.

SELECT firstname lastname from records WHERE txnno=4002310;

2. Views restrict data based to used by different users but currently this feature is not fully supported hive.

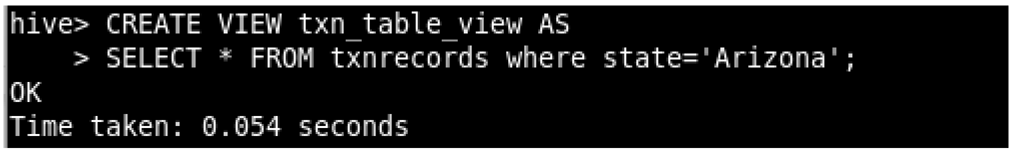
3. Summarize data from various tables which can be used to generate reports.

Syntax for view :

CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)]
[COMMENT view_comment]
[TBLPROPERTIES (property_name = property_value, ...)]
AS SELECT ...;

CREATE VIEW creates a view with the given name. An error is thrown if a table or view with the same name already exists.

If no column names are supplied, the names of the view's columns will be derived automatically from the defining SELECT expression.


```
hive> CREATE VIEW txn_table_view AS
    > SELECT * FROM txnrecords where state='Arizona';
OK
Time taken: 0.054 seconds
```

A view is a purely logical object with no associated storage. When a query references a view, the view's definition is evaluated in order to produce a set of rows for further processing by the query.

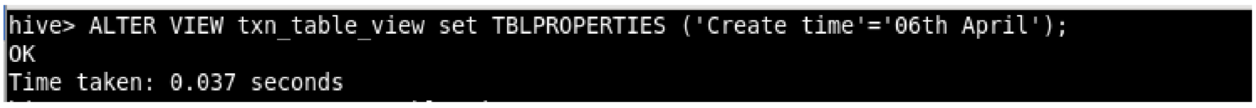
Views are read-only and may not be used as the target of LOAD/INSERT/ALTER. For changing metadata.

Syntax for ALTER view TBLPROPERTIES :

We cannot ALTER view's data, we can only ALTER the view TBLPROPERTIES.

ALTER VIEW view_name SET TBLPROPERTIES table_properties;

table_properties:
  : (property_name = property_value, property_name = property_value, ...)



```
hive> ALTER VIEW txn_table_view set TBLPROPERTIES ('Create time'='06th April');
OK
Time taken: 0.037 seconds
```

Dropping view:

DROP VIEW [IF EXISTS] view_name;

DROP VIEW removes metadata for the specified view.