**Strategies for Hive Incremental Loads**

**Approach 1: Four Step Incremental Strategy**

Step 1: Ingest – Create the table with the raw data.

Step 2: Incremental Data – Create the external table with the incremental data.

Step 3: Combine the two tables with a unique key

Step 4: Purge the Incremental table data. And whenever a new file is received, it is moved to the incremental location.

To prepare for the next series of incremental records from the source, replace the Base table with only the most up-to-date records. Also, delete the previously imported Change record content by deleting the files located in the external table location.

Sample Example:



Benefits:

* By maintaining an External Table for updates only, the table contents can be refreshed by simply adding or deleting files to that folder.

**Approach 2: Using Rank**

Step 1: Assign a rank to each row based on a unique key.

Step 2: Perform union operation on source table and incremental table

Step 3: Pick the row which has the least rank.

Sample Example:



Drawbacks: Window functions like ROW\_NUMBER and RANK have proven to be extremely useful in solving a wide variety of common T-SQL problems but a limitation in the SQL Server query optimizer often means that views containing window functions do not perform as well as expected

**Approach 3: Using Hive Merge**

Step 1: You need to configure your system to allow Hive transactions.

Step 2: Second, your table must be a transactional table. That means the table must be clustered, be stored as ORCFile data, and have a table property that says "transactional" = "true".

Enable all these properties for Hive Merge:

* set hive.support.concurrency=true;
* set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
* set hive.enforce.bucketing=true;
* set hive.exec.dynamic.partition.mode=nostrict;
* set hive.compactor.initiator.on=true;
* set hive.compactor.worker.threads=1;

Hive Merge Example:



Benefits:

* MERGE was standardized in SQL 2008 and is a powerful SQL statement that allows inserting, updating, and deleting data in a single statement. MERGE is powerful because you can specify as many WHEN MATCHED/WHEN NOT MATCHED clauses as you want.
* Transactional Tables: Hive supports single-table transactions. Tables must be marked as transactional to support UPDATE and DELETE operations. Only transactional tables can support updates and deletes.
* Run Compactor – The “Run Compactor” setting should always be set to true.
* Number of threads used by Compactor – This controls the maximum number of background MapReduce jobs that may run at any given time to compact tables. It is best to have this be a ratio of the number of transactional tables that are actively updated. In any event the value should always be greater than 0. Typically, 5 to 10 would be appropriate in production settings. 1 thread is only applicable for sandbox environments.
* Compactions: Data must be periodically compacted to save space and optimize data access. It is best to let the system handle these automatically, but these can also be scheduled in an external scheduler.
* Hive support ACID operations only for ORC format and transactional tables

Drawbacks:

* Inserts are fast, updates and deletes are relatively slower. Updates and deletes perform full partition scans. Plan for this by batching data appropriately. If your workload includes many updates and deletes, compact regularly. Your total data size will grow until you compact, and analytical queries will slowly degrade until compaction is done. Hive ACID’s architecture, updates must be done in bulk. Doing row-at-a-time updates will not work at any practical scale.