**Implemenation Of SCD Using Hive**

**Implementing SCD in HIVE**

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. There are 3 types of SCDs

1. SCD1 – In SCD1 all the old records are overwritten with new record and the old records are lost
2. SCD2 – Creating a new record and also retaining the existing record where in the new record will be the effective active record and old record(s) will be considered as inactive
3. SCD3 – SCD3 stores two versions of values for certain selected level attributes. Each record stores the previous value and the current value of the selected attribute. When the value of any of the selected attributes changes, the current value is stored as the old value and the new value becomes the current value.

We can implement the all these SCDs using hive as explained below.

**SCD1 implementation using hive**

As mentioned in SCD1 old records are overwritten by the new records and all the old records are lost. Implementation of SCD1 using hive is explained as below

* We will be having a base table(employee\_information) which stores all the employee details partitioned on effective date column
* We will be having a temporary table(temp\_employee\_information) table which has the same schema as that of base table and partitioned on the same effective date column. This table holds the latest records for the employees for whom there are any changes and new employee records if any.
* We will be having a view(difference) which records details of the employees if any for whom there is no change in the data
* Append the employee details from the difference view into temporary table by providing the effective dynamically using a dynamic partitions
* Overwrite the base table details from temporary tables so that the base table gets updated with the latest records for the employees and with new employee records if any. Here we provide the effective date dynamically using dynamic partition.
* Drop the temporary table, difference view

**Base Table(Employee\_Information)**

CREATE TABLE employee\_information (eid int,ename string,unit string)

PARTITIONED BY(effectivedate date)

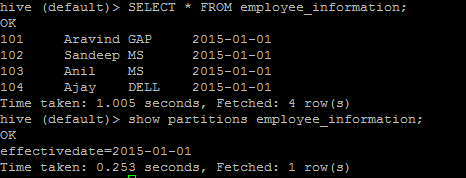
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

;



**Temporary Table(Temp\_Employee\_information)**

CREATE TABLE temp\_employee\_information (eid int,ename string,unit string)

PARTITIONED BY(effectivedate date)

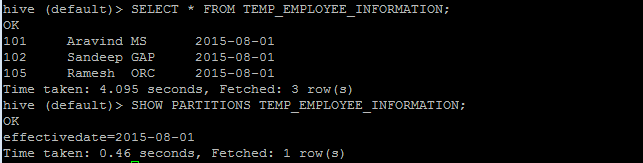
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

;



**View (Difference View)**

CREATE VIEW DIFFERENCE (eid,ename,unit,effectivedate)

AS

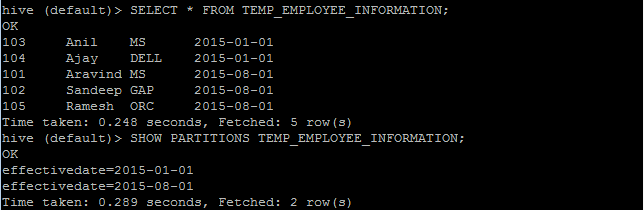
SELECT \* FROM employee\_information where employee\_information.eid NOT IN (SELECT eid FROM temp\_employee\_information);



**Appending the data from Difference View to Temp\_Employee\_Information table**

INSERT INTO TABLE TEMP\_EMPLOYEE\_INFORMATION PARTITION(effectivedate)

SELECT eid,ename,unit,effectivedate FROM DIFFERENCE;

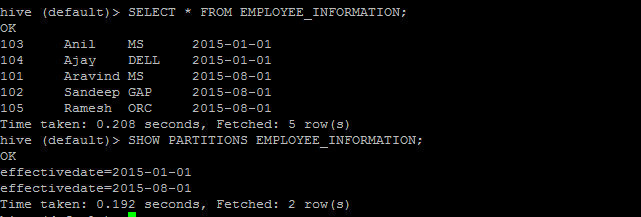


Now the temporary table contains the required SCD1 type of data which we use to overwrite the base table employee\_information;

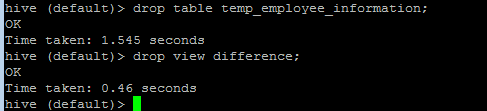
**Overwriting the data from Temp\_Employee\_Information into Employee\_Information table**

INSERT OVERWRITE TABLE EMPLOYEE\_INFORMATION PARTITION(effectivedate)

SELECT eid,ename,unit,effectivedate FROM TEMP\_EMPLOYEE\_INFORMATION;



**Dropping the tables temp\_employee\_informatiln, view difference**



So, whenever we receive the updated data for an employee repeat the step from creating and loading the data into temporary table till we drop the temporary table, view which covers the SCD1 implementation using hive.

**Flow chart for SCD1**

Create a temporary table and load the data

If there any latest records or new records available available

If There are any records for which there is no change

No

Yes No

Yes

yes

Create the difference view to track those records.

Append the records from difference view to Temporary table

Overwrite the records in base table with the records from temporary table

Drop the temPorary table and difference view

**SCD2 Implementation using Hive**

As mentioned above as part of SCD2 we need to preserve the old record and need to create a new record wherein the old record will be the inactive record(historical record) and new record will be the effective active record. We assume here that the start date, end date specifies the old and new records. Implementation part is explained below

* We will be having a base table(employee\_information) which stores all the employee details partitioned on effective date column
* We will be having a temporary table(temp\_employee\_information) table which has the same schema as that of base table and partitioned on the same effective date column. This table holds the latest records for the employees for whom there are any changes and new employee records if any.
* We will be having a view(difference) which records details of the employees if any for whom there is no change in the data.
* We will be having a view(emp\_merge\_new) which is required for updating the enddate for the employee records that are being updated. These updated records will become the historical records in the future. This can be achived by using INNER JOIN on employee\_information table, temp\_employee\_information table where we update the endate to (temp.effective\_date-1) using date\_sub function.
* We will be having a view(employee\_information\_old) which is required for tracking the existing historical records for the employee is any. This is achived using an INNER JOIN on employee\_information, temp\_employee\_information which tracks the historical records if any for the employees for whom latest details are present in temp\_employee\_information.(For historical records enddate will not be NULL)
* Append the records from difference view, emp\_merge\_view, employee\_information\_old into temp\_employee\_information table. Here we provide the effective date dynamically by making use of dynamic partitions.
* Overwrite the data from temp\_employee\_information table into employee\_information table wherein we provide the effective data dynamically using dynamic partition
* Then drop the temp\_employee\_information table, difference view, emp\_merge\_new, employee\_information\_old views.

**Base Table (Employee\_Information)**

CREATE TABLE employee\_information (eid int,ename string,unit string,endate date)

PARTITIONED BY(effectivedate date)

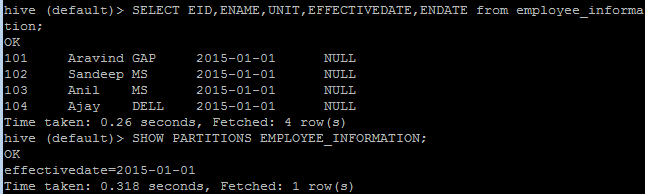
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

;



**Temporary Table(Temp\_Employee\_Information)**

CREATE TABLE temp\_employee\_information (eid int,ename string,unit string,endate date)

PARTITIONED BY(effectivedate date)

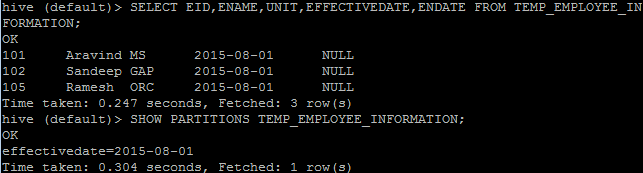
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

**;**



**View(Difference)**

CREATE VIEW DIFFERENCE (eid,ename,unit,effectivedate,enddate)

AS

SELECT employee\_information.eid,employee\_information.ename,employee\_information.unit,employee\_information.effectivedate,employee\_information.endate FROM employee\_information where employee\_information.eid NOT IN (SELECT eid FROM temp\_employee\_information);



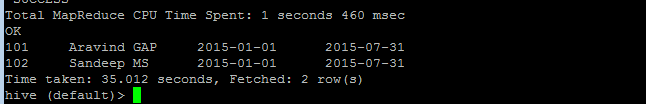
**View(emp\_merge\_view)**

CREATE VIEW emp\_merge\_new(eid,ename,unit,effectivedate,enddate)

AS

SELECT employee\_information.eid,employee\_information.ename,employee\_information.unit,employee\_information.effectivedate,date\_sub(temp\_employee\_information.effectivedate,1) as enddate FROM employee\_information INNER JOIN temp\_employee\_information ON employee\_information.eid=temp\_employee\_information.eid

WHERE employee\_information.endate IS NULL;



Updating the end date using date\_sub(temp\_employee\_information.effectivedate,1)

**View(employee\_information\_old)**

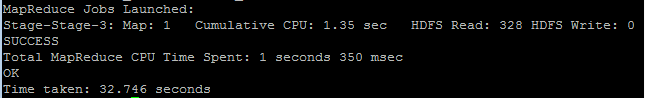
CREATE VIEW employee\_information\_old(eid,ename,unit,effectivedate,enddate)

AS

SELECT employee\_information.eid,employee\_information.ename,employee\_information.unit,employee\_information.effectivedate,date\_sub(temp\_employee\_information.effectivedate,1) as enddate FROM employee\_information INNER JOIN temp\_employee\_information ON employee\_information.eid=temp\_employee\_information.eid

WHERE employee\_information.endate IS NOT NULL;

**Note –** As part of this doc this view doesn’t contains any record as currently for all the employees in employee\_information table employee don’t have any historical records.



As mentioned for historical records end date will not be NULL.

**Appending the records from Difference view, Emp\_merge\_new, Employee\_information\_old view into temp\_employee\_information**

INSERT INTO TABLE temp\_employee\_information PARTITION(effectivedate)

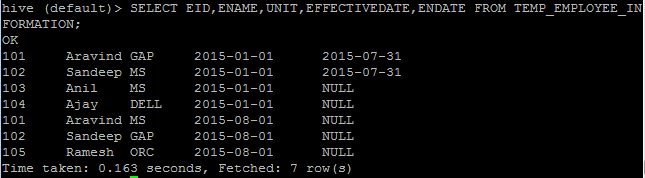
SELECT eid,ename,unit,enddate,effectivedate FROM DIFFERENCE;

INSERT INTO TABLE temp\_employee\_information PARTITION(effectivedate)

SELECT eid,ename,unit,enddate,effectivedate FROM emp\_merge\_new;

INSERT INTO TABLE temp\_employee\_information PARTITION(effectivedate)

SELECT eid,ename,unit,enddate,effectivedate FROM employee\_information\_old;

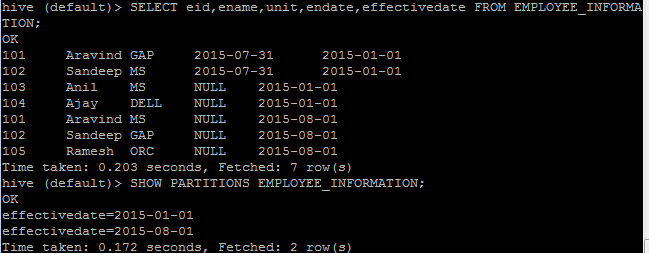


Now the temporary table contains the required SCD1 type of data which we use to overwrite the base table employee\_information;

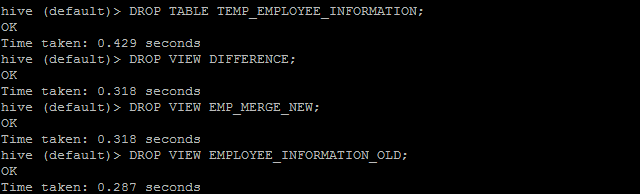
**Overwriting the temp\_employee\_information data with employee\_information data**

INSERT OVERWRITE TABLE EMPLOYEE\_INFORMATION PARTITION(effectivedate)

SELECT eid,ename,unit,endate,effectivedate FROM TEMP\_EMPLOYEE\_INFORMATION;



**Drop the table temp\_employee\_information, view difference, view emp\_merge\_new, view employee\_information\_old.**



So, whenever we receive the updated data for an employee repeat the step from creating and loading the data into temporary table till we drop the temporary table, views which covers the SCD2 implementation using hive.

**Flow chart for SCD2**

Drop the temporary table, difference view, merge\_view, employee\_infomration\_old

Overwrite the records from temporary table into base table

Append the records from difference view, merge view, employee information old view to the temporary table

Create employee\_information\_old view for tracking the historical records

Create a merge view for updating the end date

Create the difference view to track those records

If There are any records for which there is no change

Create a temporary table and load the data

If there any latest records or new records available available

**SCD3 Implementation using hive**

SCD3 is similar to SCD2 but it just preserves only the latest attribute in a record and the immediate previous attribute for that record. For example assume the below details of an employee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eid | Ename | Effective Date | Current Account | Previous Account |
| 101 | Aravind | 1-Jan-15 | GAP | NULL |

If the above employee moved from to another account then the table should be updated as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eid | Ename | Effective Date | Current Account | Previous Account |
| 101 | Aravind | 1-Jan-16 | FIDELITY | GAP |

If the above employee moved again to another account then table should be updated as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eid | Ename | Effective Date | Current Account | Previous Account |
| 101 | Aravind | 1-Jan-17 | BOFA | FIDELITY |

Thus the table contains only the current account and immediate previous account

This can be implemented as follows using hive

* We will be having a base table(Employee\_Informataion) which contains the details of the employees which tracks all the employee details and their account details partitioned on effective date.
* We will be having temporary table(temp\_employee\_information) which contains the details of the employees who are transferred from one account to the other**(Ensure that current unit column names in temp table, emp\_account table must be different),** new employee records if any**.** Alsothe table has same schema as that of employee\_informatoin;
* We will be creating a view(difference) to track the employee details for those whose account details are not getting updated
* We will be creating another view (merge\_view) which takes the effective date, current account details from the temp\_employee\_information table, current account details from employee\_information table
* Create a table merge\_table same as merge\_view using CTAS statement**(This is because we will appending the records from difference view to merge\_table instead of merge\_view as we cannot append/load the records into a view**)
* Append the records from difference view into merge\_table
* Also append all the records from temporary table whose eid is not present in merge\_view so that new records if any will be loaded into merge\_table
* Then we will be overriding the employee\_information table from merge\_table so that the table contains the SCD3 type of data.
* Then drop the view difference, merge\_view, merge\_table
* Repeating the above procedure whenever we get the new records will suffice our SCD3 requirements

All the above mentioned steps are shown below

**Base Table(Employee\_Information)**

CREATE TABLE employee\_information (eid int,ename string,currentunit string,previousunit string)

PARTITIONED BY(effectivedate date)

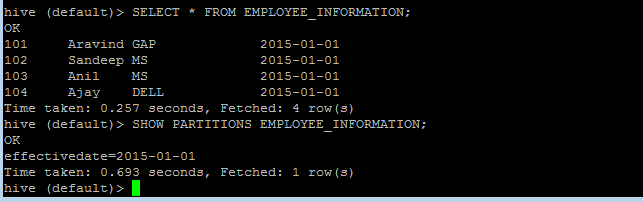
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

;



**Temp Table(Temp\_Employee\_Information)**

CREATE TABLE temp\_employee\_information (eid int,ename string,updatedunit string)

PARTITIONED BY(effectivedate date)

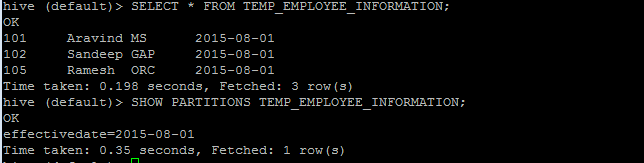
ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

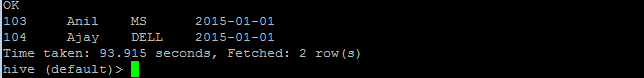
;



**Difference View**

CREATE VIEW DIFFERENCE (eid,ename,unit,effectivedate)

AS

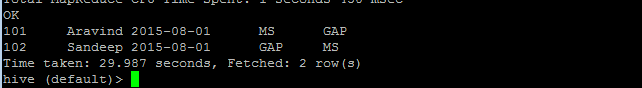
SELECT EMPLOYEE\_INFORMATION.EID,ENAME,CURRENTUNIT,EFFECTIVEDATE FROM employee\_information where employee\_information.eid NOT IN (SELECT eid FROM temp\_employee\_information);

**Merge\_View**

CREATE VIEW MERGE\_VIEW (eid,ename,effectivedate,currentunit,previousunit) AS

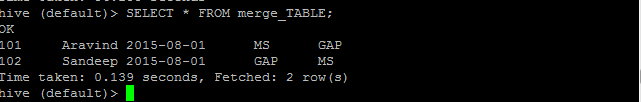
SELECT EMPLOYEE\_INFORMATION.EID,EMPLOYEE\_INFORMATION.ENAME,TEMP\_EMPLOYEE\_INFORMATION.EFFECTIVEDATE,TEMP\_EMPLOYEE\_INFORMATION.UPDATEDUNIT,EMPLOYEE\_INFORMATION.CURRENTUNIT

FROM EMPLOYEE\_INFORMATION JOIN TEMP\_EMPLOYEE\_INFORMATION ON EMPLOYEE\_INFORMATION.EID=TEMP\_EMPLOYEE\_INFORMATION.EID;



**Merge\_Table**

CREATE TABLE MERGE\_TABLE AS SELECT \* FROM MERGE\_VIEW;



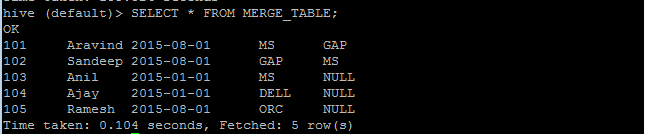
**Note** – Here the purpose of creating a merge\_table is because we cannot load the data into merge\_view as it’s a view. Hence we have created a table and we will be loading the same into the table.

**Appending the record into merge\_table from difference view**

INSERT INTO TABLE MERGE\_TABLE SELECT EID,ENAME,EFFECTIVEDATE,UNIT,NULL FROM DIFFERENCE;

INSERT INTO TABLE MERGE\_TABLE SELECT TEMP\_EMPLOYEE\_INFORMATION.EID,ENAME,EFFECTIVEDATE,UPDATEDUNIT,NULL FROM TEMP\_EMPLOYEE\_INFORMATION

WHERE TEMP\_EMPLOYEE\_INFORMATION.EID NOT IN(SELECT EID FROM MERGE\_VIEW);

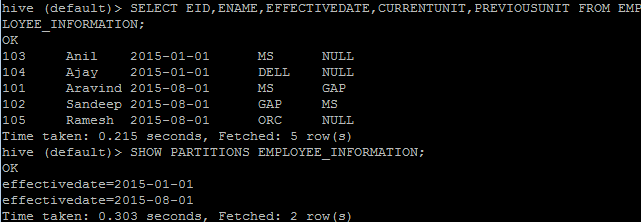


Now the merge\_table contains the actual SCD3 Type of data. We will be using this data to override the employee\_information table.

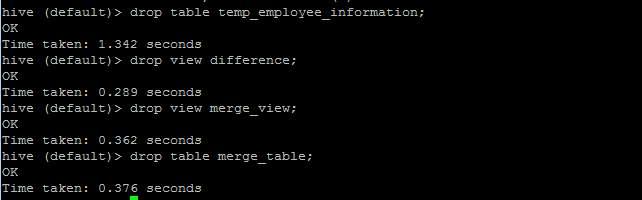
**Overwriting the records from merge\_table into employee\_information table**

INSERT OVERWRITE TABLE EMPLOYEE\_INFORMATION PARTITION(EFFECTIVEDATE)

SELECT eid,ename,currentunit,previousunit,effectivedate FROM MERGE\_TABLE;



**Dropping the temporary table, difference view, merge\_view, merge\_table**



So, whenever we receive the updated data for an employee repeat the step from creating and loading the data into temporary table till we drop the temporary table, views which covers the SCD2 implementation using hive.

**Flow Chart for SCD3** No Yes No Ye

Drop the temporary tables and views

Overwrite the data from merge\_table into base table

Append the records from difference view, only new records from temporary table into merge\_table

Create a merge table same as that of the merge view using CTAS

Create a merge view which contains the current and immediate previous values for the employee records

If There are any records for which there is no change

Create the difference view to track those records

Create a temporary table and load the data

If there any latest records or new records available available

**Points to Remember**

* All the mentioned above example details are done using Hive 0.14.0. Hence some of the features may not be present in previous versions. For example while creating the difference view we have used “**NOT IN”**  with sub query. This feature is not present in the previous versions of hive. In those versions using **LEFT SEMI JOIN** will resolve this issue
* Also in SCD2 implementation we have made use of **DATE\_SUB()** built in functions. In the previous version hive doesn’t support this built in function. Hence in those previous we need to create a user defined functions for DATE\_SUB() functionality.
* Starting from Hive 0.14 hive supports Update capability. But update is possible only on tables that support **ACID**(**A**tomicity, **C**onsistency, **I**solation, **D**urability) transactions and the table must be bucketed without which Update will not work. Also for the table to support ACID properties below hive transaction properties must be set to corresponding
  + [hive.support.concurrency](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.support.concurrency) – true
  + [hive.enforce.bucketing](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.enforce.bucketing) – true
  + [hive.exec.dynamic.partition.mode](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.exec.dynamic.partition.mode) – nonstrict
  + [hive.txn.manager](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.txn.manager) – org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
  + [hive.compactor.initiator.on](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.compactor.initiator.on) – true (for exactly one instance of the Thrift metastore service)
  + [hive.compactor.worker.threads](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.compactor.worker.threads) – a positive number on at least one instance of the Thrift metastore service
* When we are loading the data using dynamic partitions we need to ensure that the dynamic partition column must be specified at last in the **SELECT** without which it would result in a syntactical error.

**References**

[**https://cwiki.apache.org/confluence/display/Hive/Home**](https://cwiki.apache.org/confluence/display/Hive/Home)