
THE HIVE
Will Du@Toronto

What Have We Learn So Far?

- What is Hive Partition?
- What is Hive Bucket Table?
- What is bucket table used for?
- What is different between internal and external table?
- How to understand “Schema on Read” in Hive?

What do we cover today?

- Hive views
- Hive select,
- Hive join
- Hive union
- Hive load, insert, import, export
- Hive order, sort
- Case study and excersise

Hive Views – Overview

- View is a logical structure to simplify query by hiding sub query, join, functions in a virtual table.
- Hive view does not store data or get materialized
- Once the view is created, its schema is frozen immediately. Subsequent changes to the underlying table schema (such as adding a column) will not be reflected.
- If the underlying table is dropped or changed, querying the view will be failed.

```
CREATE VIEW view_name AS SELECT statement;
ALTER VIEW view_name SET TBLPROPERTIES ('comment' = 'This is a view');
DROP VIEW view_name;
```

Hive SELECT Statement

- SELECT statement is used to project the rows meeting query conditions specified
- Hive SELECT statement is subset of database standard SQL

Examples

- SELECT *, SELECT column_name, SELECT DISTINCT
- SELECT * FROM employee LIMIT 5 //Limit rows returned
- WITH t1 AS (SELECT …) SELECT * FROM t1 //CTE – Common Table Expression
- SELECT * FROM (SELECT * FROM employee) a; //Nested query
- SELECT name, sex_age.sex AS sex FROM employee a WHERE EXISTS

  (SELECT * FROM employee b WHERE a.sex_age.sex = b.sex_age.sex AND b.sex_age.sex = 'male');

  //The subquery support EXIST, NOT EXSIT, IN, NOT IN.

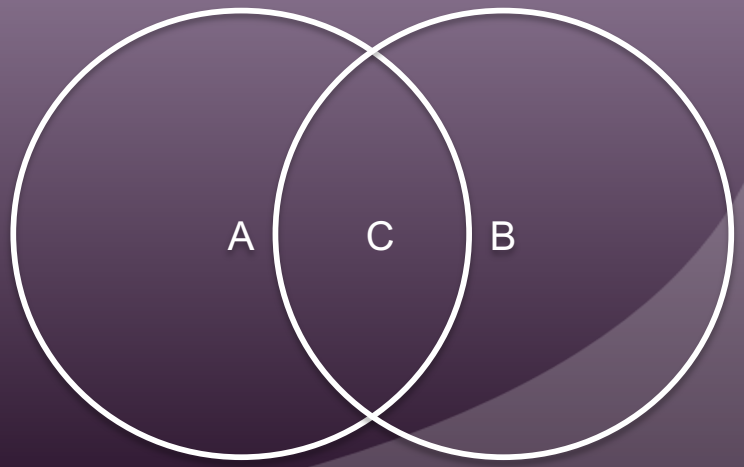
  However, nested subquery is not supported. IN and NOT IN only support one column.

Hive Join - Overview

- JOIN statement is used to combine the rows from two or more tables together
- Hive JOIN statement is similar to database JOIN. But Hive does not support unequal JOIN.
- INNER JOIN, OUTER JOIN (RIGHT OUTER JOIN, LEFT OURTER JOIN, FULL OUTER JOIN), and CROSS JOIN (Cartesian product/JOIN ON 1=1), Implicit JOIN (INNER JOIN without JOIN keywords but use , separate tables)

Example

- C = C1 JOIN C2
- A = C1 LEFT OUTER JOIN C2
- B = C1 RIGHT OUTER JOIN C2
- AUBUC = C1 FULL OUTER JOIN C2



Hive Join – MAPJOIN

- MAPJOIN statement means doing JOIN only by map without reduce job. The MAPJOIN statement reads all the data from the small table to memory and broadcast to all maps.
- When hive.auto.convert.join setting is set to true, Hive automatically converts the JOIN to MAPJOIN at runtime if possible instead of checking the MAPJOIN hint.

Example:

```
SELECT /*+ MAPJOIN(employee) */ emp.name, emph.sin_number
FROM employee emp JOIN employee_hr emph ON emp.name = emph.name;
```

The MAPJOIN operator does not support the following:

- Use MAPJOIN after UNION ALL, LATERAL VIEW, GROUP BY/JOIN/SORT BY/CLUSTER BY/DISTRIBUTE BY
- Use MAPJOIN before by UNION, JOIN and other MAPJOIN

Hive Set Ops – UNION

- Hive support UNION ALL (keep duplications) and support UNION after v1.2.0
- Hive UNION ALL cannot use in the top level query until v0.13.0,

  such as SELECT A UNION ALL SELECT B
- Other set operator can be implemented using JOIN/OUTER JOIN to implement

  Example:

```
//MINUS
jdbc:hive2://> SELECT a.name
. . . . . .> FROM employee a
. . . . . .> LEFT JOIN employee_hr b
. . . . . .> ON a.name = b.name
. . . . . .> WHERE b.name IS NULL;
```

```
//INTERCEPT
jdbc:hive2://> SELECT a.name
. . . . . .> FROM employee a
. . . . . .> JOIN employee_hr b
. . . . . .> ON a.name = b.name;
```

Exercise 01 Table Projection and Join

Time: 15 min

Exercise 01 Task

1. Get distinct employee name from employee table
2. Fetch first few rows from the table
3. Find out the preferred workplace for Michale
4. Practice CTE and subquery
5. Get emp.name, emph.sin_number by join employee and employee_hr
6. Get emp.name, empi.employee_id, emph.sin_number by employee and employee_hr and employee_id
7. Try Implicit and self join
8. Try cross join, outer join, left and right outer join
9. Union/Intercept/Minus names from employee and employee_hr

Hive Data Movement – LOAD

- To move data in Hive, it uses the LOAD keyword. Move here means the original data is moved to the target table/partition and does not exist in the original place anymore.

```
1
2   LOAD DATA LOCAL INPATH '/home/dayongd/Downloads/employee_hr.txt'
3   OVERWRITE INTO TABLE employee_hr;
4
5   LOAD DATA LOCAL INPATH '/home/dayongd/Downloads/employee.txt'
6   OVERWRITE INTO TABLE employee_partitioned
7   PARTITION (year=2014, month=12);
8
9   LOAD DATA INPATH '/user/dayongd/employee/employee.txt'
10  OVERWRITE INTO TABLE employee;
11
12  LOAD DATA INPATH 'hdfs://[dfs_host]:8020/user/dayongd/employee/employee.txt'
13  OVERWRITE INTO TABLE employee;
```

LOCAL keywords specifies where the files are located in the host
OVERWRITE is used to decide whether to append or replace the existing data

Hive Data Exchange – INSERT

- To insert data into table/partition, Hive uses INSERT statement.
- Generally speaking, Hive INSERT is weak than what's in the DBMS.
- Hive INSERT supports OVERWRITE and INTO syntax
- Hive only support INSERT … SELECT (INSERT VALUES start from v0.14.0 on special ACID tables only)
- Hive use INSERT OVERWRITE LOCAL DIRECTORY local_directory SELECT * FROM employee to insert data to the local files (opposite to LOAD statement). Only "OVERWRITE" keywords are supported. There is no support of appending data.
- Hive supports multiple INSERT from the same table

Hive Data Exchange – INSERT Example

```
1   //Insert from select
2   INSERT INTO TABLE employee SELECT * FROM ctas_employee;
3
4   //Multiple insert
5   FROM ctas_employee
6   INSERT OVERWRITE TABLE employee
7   SELECT *
8   INSERT OVERWRITE TABLE employee_internal
9   SELECT * ;
10
11  //Insert to file in the local directory
12  INSERT OVERWRITE LOCAL DIRECTORY '/tmp/output1'
13  SELECT * FROM employee;
14
15  //Insert to local file with specified row separators
16  INSERT OVERWRITE LOCAL DIRECTORY '/tmp/output2'
17  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
18  SELECT * FROM employee;
```

Better performance

Ways to export data

Hive Data Exchange – [EX|IM]PORT

- Hive uses IMPORT and EXPORT statement to deal with data migration
- All data and metadata are exported or imported
- The EXPORT statement export data in a subdirectory called data and metadata in a file called _metadata. For example,
  - EXPORT TABLE employee TO '/user/dayongd/output3';
  - EXPORT TABLE employee_partitioned partition (year=2014, month=11) TO '/user/dayongd/output5';
- After EXPORT, we can manually copy the exported files to the other HDFS. Then, import them with IMPORT statement.
  - IMPORT TABLE empolyee_imported FROM '/user/dayongd/output3';
  - IMPORT TABLE employee_partitioned_imported FROM '/user/dayongd/output5';

Exercise 02 Data Movement

Time: 15 min

Exercise 02 Task

1. Load a file from local and hdfs folder to internal employee table
2. Create table ctas_employee by select data from employee
3. Practice insert from CTE
4. Practice multi-insert
5. Practice Dynamic partition insert to employee table partition
6. Insert data from employee to local csv file
7. Try export and import tables for data migration
8. Flat workplace, skills_score in employee table

Hive Sorting Data – ORDER and SORT

- ORDER BY (ASC|DESC) is similar to standard SQL
- ORDER BY in Hive performs a global sort of data by using only one reducer. For example,

  SELECT name FROM employee ORDER BY NAME DESC;

- SORT BY (ASC|DESC) decides how to sort the data in each reducer.
- When number of reducer is set to 1,

  it is equal to ORDER BY. For example,

```
1  SET mapred.reduce.tasks = 2;
2  SELECT name FROM employee SORT BY NAME DESC;
3  +----------+
4  | name     |
5  +----------+
6  | Shelley  |
7  | Michael  |
8  | Lucy     |
9  | Will     |
10 +----------+
```

When there is more than 1 reducer, the data is not sort properly.

Hive Sorting Data – DISTRIBUTE

- DISTRIBUTE BY is similar to the GROUP BY statement in standard SQL
- It makes sure the rows with matching column value will be partitioned to the same reducer
- When to use with SORT BY, put DISTRIBUTE BY before SORT BY (since partition is ahead of reducer sort)
- The distributed column must appear in the SELECT column list.

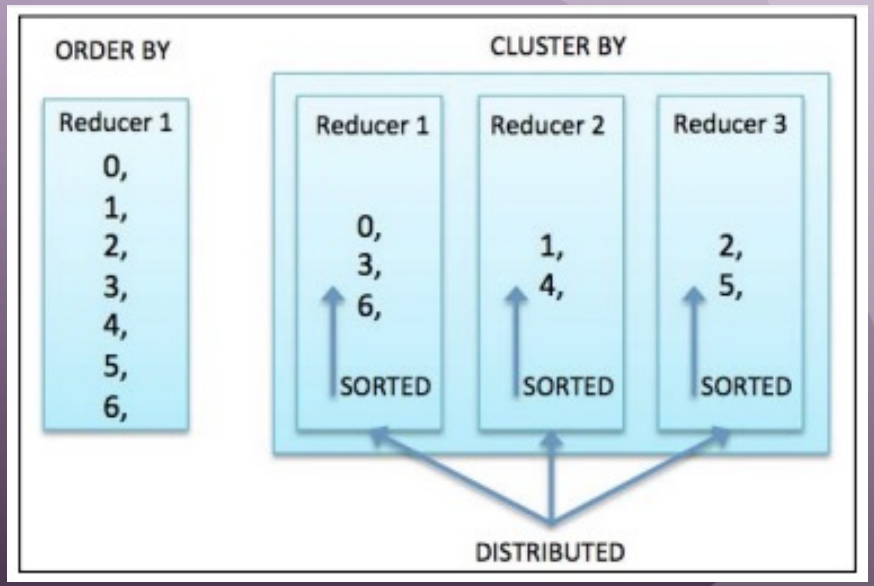
```
SELECT department_id ,name, employee_id, evaluation_score
FROM employee_hr
DISTRIBUTE BY department_id SORT BY evaluation_score;
```

Hive Sorting Data – CLUSTER

- CLUSTER BY = DISTRIBUTE BY + SORT BY on the same column
- CLUSTER BY does not support DESC yet
- To fully utilize all reducer to perform global sort, we can use CLUSTER BY first, then ORDER BY after.
- An example,

  SELECT name, employee_id

  FROM employee_hr CLUSTER BY name;



Exercise 03 Hive Overall Practice

1. A shopping data analysis – home work

2. File header and trailer generation – on class
- Header row: HEADER|2017-03-26 11:34:62|employee_20170326.flat
- Trailer row: TRAILER|2017-03-26|DETAIL ROW COUNT: 65

Time: 30 min

Hive Other Topics

- Hive operators
- Hive UDFs | PLSQL
- Hive aggregations
- UDFs extension
- Streaming
- Hive securities

Q&A

Thank You

Contact me at willddy@gmail.com

iT 21 LEARNING