Cube Abstraction In Hive

Introduction

Cube abstraction in Hive enables users to define schema of the data at warehousing level and also query the same, without knowing the physical storages and rollups.

Terminology

Cube:

- A cube is a set of dimensions and measures in a particular subject.

Measure:

- A measure is a quantity that you are interested in measuring.
- A measure will have name, type, default aggregator and a format string.

Dimension:

- A dimension is an attribute, or set of attributes, by which you can divide measures into sub-categories.
- A dimension can be as simple as having name and type. // BaseDimension
- A dimension can also refer to a dimension table // ReferencedDimension
- A dimension can be a hierarchy of dimensions. // HierachicalDimension
- A dimension can define all the possible values it can take. // InlineDimension

Fact Tables:

- Cube will have fact tables associated with it.
- A fact table would have subset of measures and dimensions.
- Fact tables can be rolled at any dimension and time.

Dimension tables:

- A table with list of columns.
- The table can have references to other dimension tables.
- The dimension tables can be shared across cubes.

Storage Model

We would like to share the schema definition of cube, facts and dimensions across clusters. So, defining a storage model such that Fact or dimension table can have storages associated with it. Also, Storage can be anything like storage supported by FileSystem such as HDFS or S3; or supported by StorageHandler such as HBase.

Update Period:

- Fact or dimension tables can be updated at regular intervals.

- Supports SECONDLY, MINUTELY, HOURLY, DAILY, WEEKLY, MONTHLY, QUARTERLY, YEARLY update periods.

Fact storage table:

- Fact storage table is the physical fact table for the associated storage, for the specified update period.
- Will have the same schema as fact table definition.
- A fact table can have multiple storages associated with it; and multiple update periods associated with each storage.

Dimension storage table:

- Dimension storage table is the physical dimension table for the associated storage.
- Will have the same schema as dimension table definition.
- Dimension storage table can have snapshot dumps at specified regular intervals or a table with no dumps.
- Dimension table can have multiple storages associated it; and zero or one update periods associated with each storage.

Naming convention:

- For fact table name: FACT1, for storage:S1, with HOURLY update period, the storage table name is S1_FACT1_HOURLY
- For dimension table name: DIM1, for storage: S1, the storage table name is S1_DIM1

Metastore:

User can do all the following operations on cube

- Define cube.
- Add fact tables to cube.
- Add dimension tables.
- Add storage tables to fact tables and dimension tables, at different update periods.
- Add more storages to fact tables
- Add more update periods to a storage

  We will store Cube, Fact table and Dimension table definitions in Hive metastore as tables without any data being added. Will leverage table properties to store the details such as references.

API

- void createCube(String name, Set<Measure> measures, Set<CubeDimension> dimensions)
- void createCubeFactTable(String cubeName, String factName,
  List<FieldSchema> columns,
  Map<Storage, List<UpdatePeriod>> storageAggregatePeriods)
- void createCubeDimensionTable(String dimName,
  List<FieldSchema> columns,
  Map<String, TableReference> dimensionReferences, Map<Storage, UpdatePeriod> dumpPeriods)
- void addStorage(CubeFactTable, Storage, List<UpdatePeriod>)

- void addStorageUpdatePeriod(CubeFactTable table, Storage storage, UpdatePeriod updatePeriod)
- void addPartition(CubeFactTable table, Storage storage, UpdatePeriod updatePeriod, Date partitionTimestamp)
- void addPartition(CubeDimensionTable table, Storage storage, Date partitionTimestamp)
- boolean partitionExists(CubeFactTable fact, Storage storage, UpdatePeriod updatePeriod, Date partitionTimestamp)
- CubeFactTable getFactTable(String tableName)
- CubeDimensionTable getDimensionTable(String tableName)

Cube QL:

User can query the cube through Cube QL, which is subset of Hive QL. Here is the grammar:

```
CUBE SELECT [DISTINCT] select_expr, select_expr, ...
FROM cube_table_reference
WHERE [where_condition AND] TIME_RANGE_IN(from, to)
[GROUP BY col_list]
[HAVING having_expr]
[ORDER BY colList]
[LIMIT number]

cube_table_reference:
    cube_table_factor
  | join_table
join_table:
    cube_table_reference JOIN cube_table_factor [join_condition]
  | cube_table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN cube_table_reference [join_condition]
cube_table_factor:
    cube_name [alias]
  | ( cube_table_reference )
join_condition:
    ON equality_expression ( AND equality_expression )*
equality_expression:
    expression = expression
colOrder: ( ASC | DESC )
colList : colName colOrder? (',' colName colOrder?)*
```

TIME_RANGE_IN(from, to) : The time range inclusive of '**from'** and exclusive of '**to'**.

Cube QL supports all the functions that hive supports as documented in Hive Functions

Note

- If No Join condition is passed for Joins, Join condition will be inferred from the cube definition.
- Projected fields will be added to group by col list automatically, if not present.
- If the group by keys specified are not projected, they will be projected

CubeQueryRewriter

Cube query rewriter rewrites the cube query into HQL with storage tables resolved.

Query translation: Example

Query:

Select SUM(measure1), SUM(measure2) from MYCUBE join DIMTABLE where time_range_in ("2012-10-31-23", "2012-11-08-00") group by DIMTABLE.dim1;

Say the engine has three storages: Storage S1, Storage S2, and Storage S3. S1 has both Daily and an hourly dump, S2 has only daily dumps, and S3 has only hourly dumps.

Also, Say we three cases D1, D2, D3.

D1: Driver works on all three storages - S1, S2 and S3;
D2: Driver works only on S2;
D3: Driver works only on S3.

The query gets translated as the following for each case:

D1:

```
Select DIMTABLE.dim1, SUM(MYCUBE.measure1), SUM(MYCUBE.measure2) from
S1_MYCUBE_FACT2_DAILY MYCUBE join S1_DIMTABLE DIMTABLE on
MYCUBE.dimkey = DIMTABLE.dimcolumn where MYCUBE.dt >= "2012-11-01" and
MYCUBE.dt <= "2012-11-07" group by DIMTABLE.dim1
UNION
Select DIMTABLE.dim1, SUM(MYCUBE.measure1), SUM(MYCUBE.measure2) from
S1_MYCUBE_FACT2_DAILY MYCUBE join S1_DIMTABLE DIMTABLE on
MYCUBE.dimkey = DIMTABLE.dimcolumn where MYCUBE.dt = "2012-11-08-00" or
MYCUBE.dt = "2012-10-31-23" group by DIMTABLE.dim1;
```

D2: Cannot answer the query

D3:

```
Select DIMTABLE.dim1, SUM(MYCUBE.measure1), SUM(MYCUBE.measure2) from
S3_MYCUBE_FACT3_HOURLY MYCUBE join S3_DIMTABLE DIMTABLE on
MYCUBE.dimkey = DIMTABLE.dimcolumn where MYCUBE.dt >= "2012-10-31-23" and
MYCUBE.dt <= "2012-11-08-00" group by DIMTABLE.dim1;
```