**Cube Abstraction In Hive**

**Metastore**

We will store Cube, Fact table and Dimension table definitions in Hive metastore. Will leverage table properties to store the details.

**Classes**

**Cube**

* Name
* Set<CubeMeasure>
* Set<CubeDimension>

**CubeMeasure**

* Name
* Default aggregator
* FormatString
* Type

**ColumnMeasure extends CubeMeasure**

**ExprMeasure extends CubeMeasure**

* Expression

**CubeDimension**

* name
* List<BaseDimension> hierarchy // the order of hierarchy is the order of elements in the list, i.e. first element in the list is the top element in the hierarchy

**BaseDimension**

* name
* type

**ReferencedDimension** extends [BaseDimension](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/BaseDimension?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0)

* [TableReference](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/TableReference?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0)

**InlineDimension** extends [BaseDimension](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/BaseDimension?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0)

* List<String> values

**AbstractCubeTable**

* name
* List<FieldSchema> columns
* Map<String, String> properties

**CubeFactTable** extends **AbstractCubeTable**

* cubename
* Map<String, Set<UpdatePeriod> // from storage to update periods

**CubeDimensionTable** extends **AbstractCubeTable**

* Map<String, UpdatePeriod> snapshotDumpPeriods // from storage to snapshot dump period
* Map<String, [TableReference](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/TableReference?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0)> dimensionReferences

**TableReference**

* Destination table name
* Destination column

enum **UpdatePeriod :**

SECONDLY,  
MINUTELY,  
HOURLY,  
DAILY,  
WEEKLY,  
MONTHLY,  
QUARTERLY,   
YEARLY;

**API**

* void createCube(String name, Set<Measure> measures, Set<CubeDimension> dimensions)
* void createCubeFactTable(String factName,  
  List<FieldSchema> columns,   
  Map<Storage, List<UpdatePeriod>> storageAggregatePeriods)
* void createCubeDimensionTable(String dimName,  
  List<FieldSchema> columns,   
  Map<String, [TableReference](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/TableReference?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0)> dimensionReferences, List<Storage> storages)
* void addStorage([CubeFactTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeFactTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0), Storage, List<UpdatePeriod>)
* void addStorageUpdatePeriod([CubeFactTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeFactTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) table, Storage storage,  
  [UpdatePeriod](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/UpdatePeriod?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) updatePeriod)
* void addPartition([CubeFactTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeFactTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) table, Storage storage,  
  [UpdatePeriod](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/UpdatePeriod?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) updatePeriod, Date partitionTimestamp)
* void addPartition([CubeDimensionTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeDimensionTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) table, Storage storage, Date partitionTimestamp)
* boolean partitionExists([CubeFactTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeFactTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) fact,   
  Storage storage, [UpdatePeriod](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/UpdatePeriod?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) updatePeriod,  
  Date partitionTimestamp)
* [CubeFactTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeFactTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) getFactTable(String tableName)
* [CubeDimensionTable](https://twiki.corp.inmobi.com/bin/edit/TechPlatform/CubeDimensionTable?topicparent=TechPlatform.BreezeMetaStore;nowysiwyg=0) getDimensionTable(String tableName)

**Cube QL:**

Cube QL 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?)\*

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

**Assumptions**

* If No Join condition is passed for Joins, Join condition will be inferred from the fact table 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 in the following phases:

**Phase1:**

Phase1 has following steps:

1. Resolve Join conditions and add missing join conditions
2. Process Aggregations by making sure that all group by keys are projected; and all projection fields are added to group by keylist;
3. Rewrite the all the columns in the query with table alias prefixed. If col1 of table tab1 is accessed, it would be changed as tab1.col1. If tab1 is already aliased say with t1, col1 is changed as t1.col1
4. Get the candidate Fact tables which can answer the given query.
5. For all the candidate tables get corresponding partitions.

**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;

After step 1:

Select SUM(measure1), SUM(measure2) from MYCUBE join DIMTABLE on MYCUBE.dimkey = DIMTABLE.dimcolumn where time\_range\_in ("2012-10-31-23", "2012-11-08-00") group by DIMTABLE.dim1;

After step 2:

Select DIMTABLE.dim1, SUM(measure1), SUM(measure2) from MYCUBE join DIMTABLE on MYCUBE.dimkey = DIMETABLE.dimcolumn where time\_range\_in ("2012-10-31-23", "2012-11-08-00") group by DIMTABLE.dim1;

After step 3:

Select DIMTABLE.dim1, SUM(MYCUBE.measure1), SUM(MYCUBE.measure2) from MYCUBE join DIMTABLE on MYCUBE.dimkey = DIMTABLE.dimcolumn where time\_range\_in ("2012-10-31-23", "2012-11-08-00") group by DIMTABLE.dim1;

After step 4:

Candidate fact tables: MYCUBE\_FACT1, MYCUBE\_FACT2, MYCUBE\_FACT3

Candidate Dimension tables: DIMTABLE

After step 5:

Get Fact tables update periods and partition cols:

MYCUBE\_FACT1: <Daily: 2012-11-01, 2012-11-02, 2012-11-03, 2012-11-04, 2012-11-05, 2012-11-06, 2012-11-07>, <Hourly: "2012-10-31-23", "2012-11-08-00">

MYCUBE\_FACT2: <Daily: 2012-11-01, 2012-11-02, 2012-11-03, 2012-11-04, 2012-11-05, 2012-11-06, 2012-11-07>, <Hourly: "2012-10-31-23", "2012-11-08-00">

MYCUBE\_FACT3: <Hourly: all hourly partitions from "2012-10-31-23" to "2012-11-08-00">

**Phase2:**

Input: The above candidate tables, with update periods and partition columns and the rewritten query after Step 5.

The phase 2 involves rewriting the query wrt to the supported storages it can work on. Illustrated with an example below:

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

For all the candidate fact tables, with update periods, construct the storage table names :

S1\_MYCUBE\_FACT1\_DAILY, S1\_MYCUBE\_FACT1\_HOURLY, S2\_MYCUBE\_FACT1\_DAILY, S3\_MYCUBE\_FACT1\_HOURLY

S1\_MYCUBE\_FACT2\_DAILY, S1\_MYCUBE\_FACT2\_HOURLY, S2\_MYCUBE\_FACT2\_DAILY, S3\_MYCUBE\_FACT2\_HOURLY

S1\_MYCUBE\_FACT3\_HOURLY, S3\_MYCUBE\_FACT3\_HOURLY

* 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.

In case of D2, Driver cannot answer the query because storage S2 has only Daily dumps. Now the candidate tables for each case above are:

D1: {S1\_MYCUBE\_FACT1\_DAILY, S1\_MYCUBE\_FACT1\_HOURLY; S3\_MYCUBE\_FACT1\_HOURLY; S1\_MYCUBE\_FACT2\_DAILY, S1\_MYCUBE\_FACT2\_HOURLY; S3\_MYCUBE\_FACT2\_HOURLY; S1\_MYCUBE\_FACT3\_HOURLY, S3\_MYCUBE\_FACT3\_HOURLY;}

D2: {}

D3: {S3\_MYCUBE\_FACT1\_HOURLY; S3\_MYCUBE\_FACT2\_HOURLY; S3\_MYCUBE\_FACT3\_HOURLY;}

* Driver will choose tables with least number of partitions. So, the candidates are:

D1: {S1\_MYCUBE\_FACT1\_DAILY, S1\_MYCUBE\_FACT1\_HOURLY; S1\_MYCUBE\_FACT2\_DAILY, S1\_MYCUBE\_FACT2\_HOURLY;}

D3: {S3\_MYCUBE\_FACT1\_HOURLY; S3\_MYCUBE\_FACT2\_HOURLY; S3\_MYCUBE\_FACT3\_HOURLY;}

* Driver will choose fact tables with least joining dimension table cardinality. Say the dimension cardinality order of the facts are MYCUBE\_FACT3 < MYCUBE\_FACT2 < MYCUBE\_FACT1.

D1: {S1\_MYCUBE\_FACT2\_DAILY, S1\_MYCUBE\_FACT2\_HOURLY;}

D1: {S1\_MYCUBE\_FACT2\_DAILY, S1\_MYCUBE\_FACT2\_HOURLY;}

D3: {S3\_MYCUBE\_FACT3\_HOURLY;}

* Query would be rewritten with candidate tables for driver by replacing the table names and rewriting the where clause as follows:

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;

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;