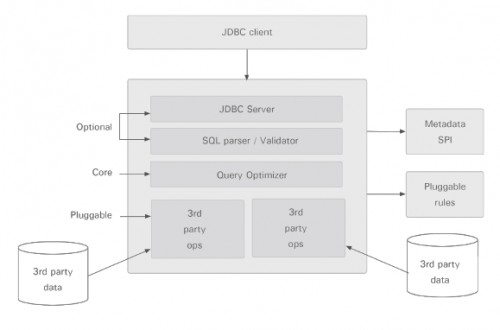
### APACHE CAlCIT　简介

动态数据管理框架　，主要功能是存储数据，线性处理　数据，保存元数据

　它是程序与数据处理与数据库存储的中介者



#### *Example1:* ReflectiveSchema　简单实现

定义　Employee,Department类　代表员工表对象和部门表对象

定义　HrSchema类代表人事数据库对象

Class.*forName*("org.apache.calcite.jdbc.Driver");

Properties info = **new** Properties();

info.setProperty("lex", "JAVA");

Connection connection = DriverManager.*getConnection*("jdbc:calcite:", info);

CalciteConnection calciteConnection =

connection.unwrap(CalciteConnection.**class**);

**SchemaPlus rootSchema = calciteConnection.getRootSchema();**

**rootSchema.add("hr", new ReflectiveSchema(new HrSchema()));**

Statement statement = calciteConnection.createStatement();

ResultSet resultSet = statement.executeQuery(sql);

#### *Example２: CSV Adapter 实现*

**model.json 结构**

{

version: '1.0',

defaultSchema: 'SALES',

schemas: [

{

name: 'SALES',

type: 'custom',

factory: 'org.apache.calcite.adapter.csv.CsvSchemaFactory',

operand: {

directory: 'sales'　　*//扫描文件的路径*

}

}

]

}

**定义CsvFieldType　字段类型**

**定义CsvTable　 extends AbstractTable**

**public** RelDataType getRowType(RelDataTypeFactory typeFactory)

定义表的字段类型　CsvEnumerator.*deduceRowType*()方法中实现

**实现Schema：CsvSchema extends AbstractSchema**

[getTableMap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getTableMap%E2%98%82). 　获取Scheam的表Map

[getFunctionMultimap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getFunctionMultimap%E2%98%82).

[getSubSchemaMap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getSubSchemaMap%E2%98%82).

[isMutable()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82isMutable%E2%98%82). 　Schema中表是否可以动态增加

**createTable() 创建表的三种实现方式**

　CsvScannableTable implements ScannableTable

　CsvTranslatableTable　　implements TranslatableTable

　CsvFilterableTable　　　 implements FilterableTable

**实现SchemaFactory接口创建CsvSchema**

create(SchemaPlus parentSchema, String name,

Map<String, Object> operand)

**CsvScannableTable**

**public** Enumerable<Object[]> scan(DataContext root)

返回自定义的枚举器CsvEnumerator

实现以下方法

T current();　//获取collection中当前元素

Gets the current element in the collection.

**boolean** moveNext(); //移动到下一元素

Advances the enumerator to the next element of the collection.

**void** reset();//游标重置

Sets the enumerator to its initial position, which is before the first element in the collection.

**void** close();//关闭

Closes this enumerable and releases resources.

## Kylin Query

# The metadata store in hbase

kylin.metadata.url=kylin\_metadata@hbase:kylin-host:2181:/hbase

# The storage for final cube file in hbase

kylin.storage.url=hbase:kylin-host:2181:/hbase

**QueryService excute()**

File modelJson = OLAPSchemaFactory.*createTempOLAPJson*(project, getConfig());

DriverManagerDataSource ds = **new** DriverManagerDataSource();

Properties props = **new** Properties();

props.setProperty(OLAPQuery.***PROP\_SCAN\_THRESHOLD***, String.*valueOf*(KylinConfig.*getInstanceFromEnv*().getScanThreshold()));

ds.setConnectionProperties(props);

ds.setDriverClassName("net.hydromatic.optiq.jdbc.Driver"); ds.setUrl("jdbc:calcite:model=" modelJson.getAbsolutePath());

modelJson 内容：

*{*

*"version": "1.0",*

*"defaultSchema": "DEFAULT",*

*"schemas": [*

*{*

*"type": "custom",*

*"name": "DEFAULT",*

*"factory": "org.apache.kylin.query.schema.OLAPSchemaFactory",*

*"operand": {*

*"project": "LEARN\_KYLIN"*

*},*

*"functions": [*

*{*

*"name": "QUARTER",*

*"className": "org.apache.kylin.query.sqlfunc.QuarterFunc"*

*}*

*]*

*}*

*]*

*}*

**定义OLAPTable extends AbstractQueryableTable implements TranslatableTabl**

*SQLTYPE\_MAPPING*.put("char", SqlTypeName.***CHAR***);

*SQLTYPE\_MAPPING*.put("varchar", SqlTypeName.***VARCHAR***);

*SQLTYPE\_MAPPING*.put("boolean", SqlTypeName.***BOOLEAN***);

*SQLTYPE\_MAPPING*.put("integer", SqlTypeName.***INTEGER***);

*SQLTYPE\_MAPPING*.put("tinyint", SqlTypeName.***TINYINT***);

*SQLTYPE\_MAPPING*.put("smallint", SqlTypeName.***SMALLINT***);

*SQLTYPE\_MAPPING*.put("bigint", SqlTypeName.***BIGINT***);

*SQLTYPE\_MAPPING*.put("decimal", SqlTypeName.***DECIMAL***);

*SQLTYPE\_MAPPING*.put("numeric", SqlTypeName.***DECIMAL***);

*SQLTYPE\_MAPPING*.put("float", SqlTypeName.***FLOAT***);

*SQLTYPE\_MAPPING*.put("real", SqlTypeName.***REAL***);

*SQLTYPE\_MAPPING*.put("double", SqlTypeName.***DOUBLE***);

*SQLTYPE\_MAPPING*.put("date", SqlTypeName.***DATE***);

*SQLTYPE\_MAPPING*.put("time", SqlTypeName.***TIME***);

*SQLTYPE\_MAPPING*.put("timestamp", SqlTypeName.***TIMESTAMP***);

*SQLTYPE\_MAPPING*.put("any", SqlTypeName.***ANY***);

**public** RelDataType getRowType(RelDataTypeFactory typeFactory)

定义表的字段类型　CsvTable. **deduceRowType** ()方法中实现

**实现Schema：OlapSchema extends AbstractSchema**

初始化方法

**private** **void** init() {

**this**.config = KylinConfig.*getInstanceFromEnv*();

**this**.storageUrl = config.getStorageUrl();

**this**.starSchemaUrl = config.getHiveUrl();

**this**.starSchemaUser = config.getHiveUser();

**this**.starSchemaPassword = config.getHivePassword();

}

[getTableMap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getTableMap%E2%98%82). 　获取Scheam的表Map

ProjectManager

ProjectL2Cache

Set<TableDesc> listExposedTables

[getFunctionMultimap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getFunctionMultimap%E2%98%82).

[getSubSchemaMap()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82getSubSchemaMap%E2%98%82).

[isMutable()](eclipse-javadoc:%E2%98%82=calcite-test/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/1.3.0-incubating%5C/calcite-core-1.3.0-incubating.jar%3Corg.apache.calcite.schema.impl(AbstractSchema.class%E2%98%83AbstractSchema%E2%98%82%E2%98%82isMutable%E2%98%82). 　Schema中表是否可以动态增加

**实现**OLAPSchemaFactory**接口创建**OLAPSchema

@Override

**public** Schema create(SchemaPlus parentSchema, String schemaName, Map<String, Object> operand) {

String project = (String) operand.get(***SCHEMA\_PROJECT***);

Schema newSchema = **new** OLAPSchema(project, schemaName);

**return** newSchema;

}

Converts this table into a [relational expression](eclipse-javadoc:%E2%98%82=kylin-query/E:%5C/maven%5C/repository%5C/org%5C/apache%5C/calcite%5C/calcite-core%5C/0.9.1-incubating%5C/calcite-core-0.9.1-incubating.jar%3Cnet.hydromatic.optiq(TranslatableTable.class%E2%98%83TranslatableTable~toRel~Lorg.eigenbase.relopt.RelOptTable$ToRelContext;~Lorg.eigenbase.relopt.RelOptTable;%E2%98%82RelNode).

**如何遍历表数据：**

**ＯlapTable 继承自AbstractQueryableTable**

@Override

**public** <T> Queryable<T> asQueryable(QueryProvider queryProvider, SchemaPlus schema, String tableName) {

**return** **new** AbstractTableQueryable<T>(queryProvider, schema, **this**, tableName) {

@SuppressWarnings("unchecked")

**public** Enumerator<T> enumerator() {

**final** OLAPQuery query = **new** OLAPQuery(EnumeratorTypeEnum.***INDEX***, 0);

**return** (Enumerator<T>) query.enumerator();

}

};

**public** **class** OLAPQuery **extends** AbstractEnumerable<Object[]> **implements** Enumerable<Object[]>

**public** Enumerator<Object[]> enumerator() {

　 …

**//cube 枚举器**

**case *INDEX*:**

**return new CubeEnumerator(olapContext,** optiqContext);

**case** ***LOOKUP\_TABLE***:

**return** **new** LookupTableEnumerator(olapContext);

//hive 枚举器

**case** ***HIVE***:

**return** **new** HiveEnumerator(olapContext);

…

}

以上三种枚举器均实现如下方法:

T current();　//获取collection中当前元素Gets the current element in the collection.

**boolean** moveNext(); //移动到下一元素Advances the enumerator to the next element of the collection.

**void** reset();//游标重置Sets the enumerator to its initial position, which is before the first element in the collection.

**void** close();//关闭

Closes this enumerable and releases resources.

**CubeEnumerator**

**立方体枚举器**

**CubeStorageEngin**

**立方体存储引擎**

**public** ITupleIterator search(StorageContext context, SQLDigest sqlDigest)

**SerializedHBaseTupleIterator**

Ｈbase迭代器

**CubeSegmentTupleIterator**

//构建scan

**private** Scan buildScan(HBaseKeyRange keyRange) {

Scan scan = **new** Scan();

scan.setCaching(***SCAN\_CACHE***);

scan.setCacheBlocks(**true**);

scan.setAttribute(Scan.***SCAN\_ATTRIBUTES\_METRICS\_ENABLE***, Bytes.*toBytes*(Boolean.***TRUE***));

**for** (RowValueDecoder valueDecoder : **this**.rowValueDecoders) {

HBaseColumnDesc hbaseColumn = valueDecoder.getHBaseColumn();

**byte**[] byteFamily = Bytes.*toBytes*(hbaseColumn.getColumnFamilyName());

**byte**[] byteQualifier = Bytes.*toBytes*(hbaseColumn.getQualifier());

scan.addColumn(byteFamily, byteQualifier);

}

scan.setStartRow(keyRange.getStartKey());

scan.setStopRow(keyRange.getStopKey());

**return** scan;

}

**ＯlapTable implements** TranslatableTable

net.hydromatic.optiq. TranslatableTable 类

/\*\*

\* Extension to {@link Table} that specifies how it is to be translated to

\* a {@link org.eigenbase.rel.RelNode planner node}.

\*

\* <p>It is optional for a Table to implement this interface. A Table that does

\* not implement this interface, a Table will be converted to anEnumerableTableAccessRel. Generally a Table will implements this interface to

create a particular subclass of RelNode, and also register rules that act

\* on that particular subclass of RelNode.</p>

\*/

*public interface* ***TranslatableTable*** *extends Table {*

*/\*\* Converts this table into a {@link RelNode relational expression}. \*/*

*RelNode toRel(*

*RelOptTable.ToRelContext context,*

*RelOptTable relOptTable);*

*}*

@Override

**public** RelNode toRel(ToRelContext context, RelOptTable relOptTable) {

**int** fieldCount = relOptTable.getRowType().getFieldCount();

**int**[] fields = identityList(fieldCount);

**return** **new** OLAPTableScan(context.getCluster(), relOptTable, **this**, fields);

}

OLAPTableScan extends TableAccessRelBase implements OLAPRel, EnumerableRel

　　//注册转换规则

　　@Override

**public** **void** register(RelOptPlanner planner) {

// force clear the query context before traversal relational operators

OLAPContext.*clearThreadLocalContexts*();

// register OLAP rules

planner.addRule(OLAPToEnumerableConverterRule.***INSTANCE***);

planner.addRule(OLAPFilterRule.***INSTANCE***);

planner.addRule(OLAPProjectRule.***INSTANCE***);

planner.addRule(OLAPAggregateRule.***INSTANCE***);

planner.addRule(OLAPJoinRule.***INSTANCE***);

planner.addRule(OLAPLimitRule.***INSTANCE***);

planner.addRule(OLAPSortRule.***INSTANCE***);

// since join is the entry point, we can't push filter past join

planner.removeRule(PushFilterPastJoinRule.***FILTER\_ON\_JOIN***);

planner.removeRule(PushFilterPastJoinRule.***JOIN***);

// **TODO** : since we don't have statistic of table, the optimization of join is too cost

planner.removeRule(SwapJoinRule.***INSTANCE***);

planner.removeRule(PushJoinThroughJoinRule.***LEFT***);

planner.removeRule(PushJoinThroughJoinRule.***RIGHT***);

// for columns in having clause will enable table scan filter rule

// cause kylin does not depend on MPP

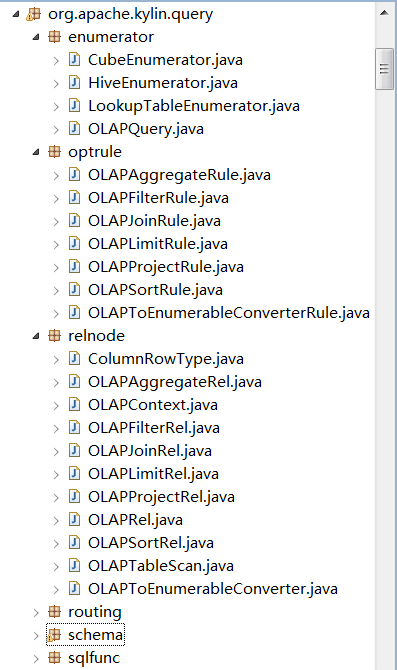
planner.removeRule(PushFilterPastProjectRule.***INSTANCE***);

// distinct count will be split into a separated query that is joined with the left query

planner.removeRule(RemoveDistinctAggregateRule.***INSTANCE***);

planner.removeRule(ExpandConversionRule.***INSTANCE***);

}



*{*

*"version": "1.0",*

*"defaultSchema": "DEFAULT",*

*"schemas": [*

*{*

*"type": "custom",*

*"name": "DEFAULT",*

*"factory": "org.apache.kylin.query.schema.OLAPSchemaFactory",*

*"operand": {*

*"project": "LEARN\_KYLIN"*

*},*

*"functions": [*

*{*

*"name": "QUARTER",*

*"className": "org.apache.kylin.query.sqlfunc.QuarterFunc"*

*}*

*]*

*}*

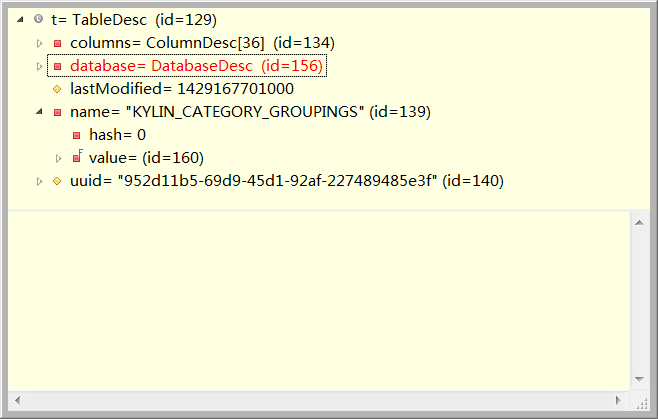
*]*

*}*

MetadataManager

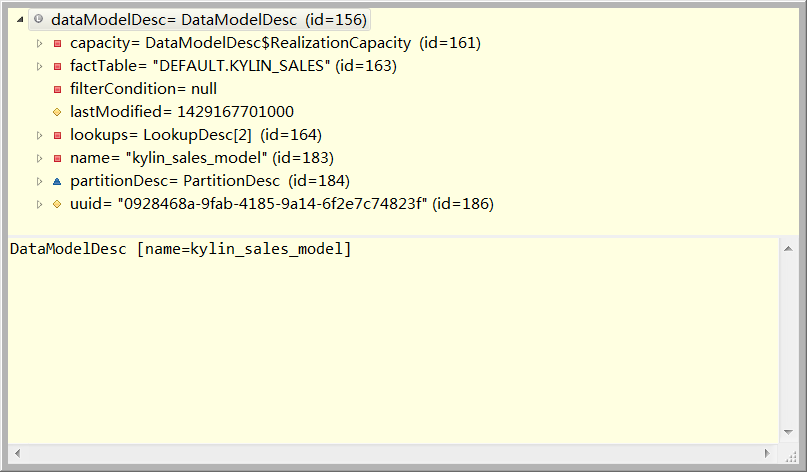
reloadAllSourceTable

[/table/DEFAULT.KYLIN\_CAL\_DT.json, /table/DEFAULT.KYLIN\_CATEGORY\_GROUPINGS.json, /table/DEFAULT.KYLIN\_SALES.json]



reloadAllSourceTableExd();

reloadAllDataModel



Kylin\_metadata (参见Kylin\_metadata.txt)

--------------------

/cube/kylin\_sales\_cube.json

--------------------

/cube\_desc/kylin\_sales\_cube\_desc.json

--------------------

/model\_desc/kylin\_sales\_model.json

--------------------

/project/learn\_kylin.json

-------------------

/table/DEFAULT.KYLIN\_CAL\_DT.json

--------------------

/table/DEFAULT.KYLIN\_CATEGORY\_GROUPINGS.json

--------------------

/table/DEFAULT.KYLIN\_SALES.json

--------------------

/table\_exd/DEFAULT.KYLIN\_CAL\_DT.json

**CubeStorageEngine**

@Override

**public** ITupleIterator search(StorageContext context, SQLDigest sqlDigest) {

Collection<TblColRef> groups = sqlDigest.groupbyColumns;

TupleFilter filter = sqlDigest.filter;

// build dimension & metrics

Collection<TblColRef> dimensions = **new** HashSet<TblColRef>();

Collection<FunctionDesc> metrics = **new** HashSet<FunctionDesc>();

buildDimensionsAndMetrics(dimensions, metrics, sqlDigest);

// all dimensions = groups + others

Set<TblColRef> others = Sets.*newHashSet*(dimensions);

others.removeAll(groups);

// expand derived

Set<TblColRef> derivedPostAggregation = Sets.*newHashSet*();

Set<TblColRef> groupsD = expandDerived(groups, derivedPostAggregation);

Set<TblColRef> othersD = expandDerived(others, derivedPostAggregation);

othersD.removeAll(groupsD);

derivedPostAggregation.removeAll(groups);

// identify cuboid

Set<TblColRef> dimensionsD = Sets.*newHashSet*();

dimensionsD.addAll(groupsD);

dimensionsD.addAll(othersD);

Cuboid cuboid = identifyCuboid(dimensionsD);

context.setCuboid(cuboid);

// isExactAggregation? meaning: tuples returned from storage requires no further aggregation in query engine

Set<TblColRef> singleValuesD = findSingleValueColumns(filter);

**boolean** isExactAggregation = isExactAggregation(cuboid, groups, othersD, singleValuesD, derivedPostAggregation);

context.setExactAggregation(isExactAggregation);

// translate filter for scan range and compose returning groups for coprocessor, note:

// - columns on non-evaluatable filter have to return

// - columns on loosened filter (due to derived translation) have to return

Set<TblColRef> groupsCopD = Sets.*newHashSet*(groupsD);

collectNonEvaluable(filter, groupsCopD);

TupleFilter filterD = translateDerived(filter, groupsCopD);

// flatten to OR-AND filter, (A AND B AND ..) OR (C AND D AND ..) OR ..

TupleFilter flatFilter = flattenToOrAndFilter(filterD);

// translate filter into segment scan ranges

List<HBaseKeyRange> scans = buildScanRanges(flatFilter, dimensionsD);

// check involved measures, build value decoder for each each family:column

List<RowValueDecoder> valueDecoders = translateAggregation(cubeDesc.getHBaseMapping(), metrics, context);

setThreshold(dimensionsD, valueDecoders, context); // set cautious threshold to prevent out of memory

setCoprocessor(groupsCopD, valueDecoders, context); // enable coprocessor if beneficial

setLimit(filter, context);

HConnection conn = HBaseConnection.*get*(context.getConnUrl());

**return** **new** SerializedHBaseTupleIterator(conn, scans, cubeInstance, dimensionsD, filterD, groupsCopD, valueDecoders, context);

}

Column metadat name must be upper case letters