目录

[APACHE CAlCIT　简介 1](#_Toc422385471)

[*1.* ReflectiveSchema　简单实现 2](#_Toc422385472)

[2. CSV Adapter 实现 2](#_Toc422385473)

[3.Solr适配 5](#_Toc422385474)

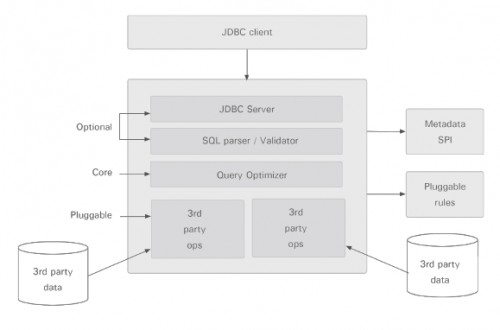
[4.HBase适配 8](#_Toc422385475)

### APACHE CAlCIT　简介

*Apache Calcite is a dynamic data management framework.*

*It contains many of the pieces that comprise a typical database management system, but omits some key functions: storage of data, algorithms to process data, and a repository for storing metadata.*

*Calcite intentionally stays out of the business of storing and processing data. As we shall see, this makes it an excellent choice for mediating between applications and one or more data storage locations and data processing engines. It is also a perfect foundation for building a database: just add data.*



Apache doc : [*http://calcite.incubator.apache.org/docs/*](http://calcite.incubator.apache.org/docs/)

### *1.* 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);

### 2. CSV Adapter 实现

#### 2.1适配器接口简要说明

**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.

#### 2.2 使用说明

1. 定义表数据　sales/DEPTS.csv

|  |  |
| --- | --- |
| **DEPTNO:int** | **NAME:string** |
| 10 | Sales |
| 20 | Marketing |
| 30 | Accounts |

1. 定义　csv\_model.json

{

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*

}

}

]

}

1. java 调用示例

　　 Connection connection = **null**;

Statement statement = **null**;

**try** {

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

info.put("model",”***path***/csv\_model.json”);

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

statement = connection.createStatement();

final ResultSet resultSet = statement.executeQuery(sql);

　　　　}　……

### 3.Solr适配

#### 3.1 实现接口

１SolrSchemaFactory

２　SolrSchema

３SolrScannableTable

４SolrEnumerator

１创建scheam工厂　　 **implements** **SchemaFactory**

　２定义schema　　　 **extends** **AbstractSchema**

3定义表 **extends** **AbstractTable**

　４实现表数据遍历　迭代 **implements** **Enumerator**<E>

#### 3.2元数据定义及获取

ＭetaDataManager

元数据管理

TableDesc1

TableDesc2

TableDesc3

**json file parse to object**

Eg:EMPLOYY.json

　{

**connInfo**: { 　　　　　　　　　　　//solr　client 连接信息

ip: "192.168.202.129",

port: "8080",

coreName: "core1"

},

**tableColumns**: [　　　　　　　　　　//表字段定义信息 字段名称大写

{

name: "ID",

type: "int"

},

{

name: "NAME",

type: "string"

},

{

name: "DEPTNO",

type: "int"

},

{

name: "SALARY",

type: "float"

},

{

name: "COMMISSION",

type: "int"

},

{

name: "\_version\_",

type: "string"

}

],

queryFilter:{　　　　　　　　　　　　***//查询返回结果限定***

limit:"1000"

}

}

#### 3.3 使用说明

**1 定义元数据　solr/DEPTS.json**

**2 定义　solr\_model.json**

{

version: '1.0',

defaultSchema: 'SALES\_SOLR',

schemas: [

{

name: 'SALES\_SOLR',

type: 'custom',

factory: 'com.airsupply.adapter.solr.SolrSchemaFactory',

operand: {

directory: 'solr'

}

}

]

}

**3 java 调用示例**

　　 Connection connection = **null**;

Statement statement = **null**;

**try** {

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

info.put("model",”***path***/solr\_model.json”);

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

statement = connection.createStatement();

final ResultSet resultSet = statement.executeQuery(sql);

　　　　}　……

### 4.HBase适配

#### 4.1 实现接口

１HBaseSchemaFactory

２　HBaseSchema

３HBaseScannableTable

４HBaseEnumerator

1创建scheam工厂　　 **implements** **SchemaFactory**

　 2定义schema　　　 **extends** **AbstractSchema**

3定义表 **extends** **AbstractTable**

　 4实现表数据遍历　迭代 **implements** **Enumerator**<E>

#### 4.2元数据定义及获取

ＭetaDataManager

元数据管理

TableDesc1

TableDesc2

*（Families列族）*

TableDesc3

**json file parse to object**

{

connInfo: {　　　　　　　//定义连接信息

ip: "192.168.202.129",

port: "2181",

tableName:"blog";

},

rowKeyName:"ROW\_KEY", 　//定义row key的名称

families: [　　　　　　　//定义列族

{

name: "article",

columns: [

{

name: "TITLE",

type: "string"

},

{

name: "CONTENT",

type: "string"

},

{

name: "TAG",

type: "string"

}

]

},

{

name: "author",

columns: [

{

name: "NAME",

type: "string"

},

{

name: "NICKNAME",

type: "string"

}

]

}

],

queryFilter: { //过滤设置

startRow:"rowKey2",

stopRow: "rowKey5"

}

}

#### 4.3 使用说明

1 定义元数据　hbase/BLOG.json

2 定义hbase \_model.json

{

version: '1.0',

defaultSchema: 'HBASE\_SCHEMA',

schemas: [

{

name: 'HBASE\_SCHEMA',

type: 'custom',

factory: 'com.airsupply.adapter.hbase.HBaseSchemaFactory',

operand: {

directory: 'hbase'

}

}

]

}

3 java 调用示例

　　 Connection connection = **null**;

Statement statement = **null**;

//列名称:*familyName*\_*columnName*

String sql=”select ROW\_KEY ,**ARTICLE\_TITLE** from BLOG”

**try** {

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

info.put("model",”***path***/hbase\_model.json”);

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

statement = connection.createStatement();

final ResultSet resultSet = statement.executeQuery(sql);

　　　　}　……