大数据技术之Kudu

第1章 Kudu概述

1.1 定义

Kudu 是一个针对 Apache Hadoop 平台而开发的列式存储管理器。

1.2 基础架构

Kudu也采用了Master-Slave形式的中心节点架构，管理节点被称作Kudu Master，数据节点被称作Tablet Server（可对比理解HBase中的RegionServer角色）。一个表的数据，被分割成1个或多个Tablet，Tablet被部署在Tablet Server来提供数据读写服务。

Kudu Master在Kudu集群中，发挥如下的一些作用：

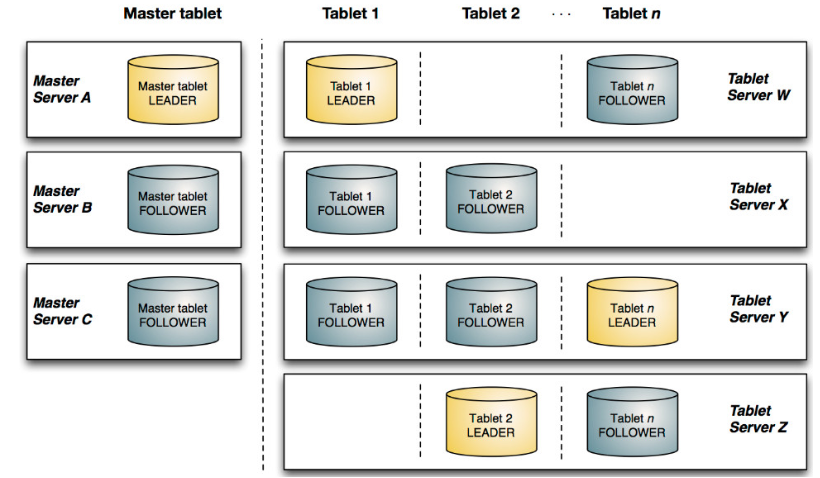
1. 用来存放一些表的Schema信息，且负责处理建表等请求。

2. 跟踪管理集群中的所有的Tablet Server，并且在Tablet Server异常之后协调数据的重部署。

3. 存放Tablet到Tablet Server的部署信息。

Tablet与HBase中的Region大致相似，但存在如下一些明显的区别点：

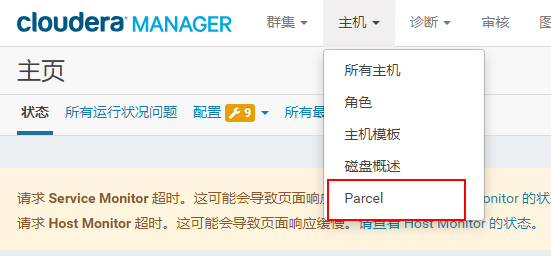
Tablet包含两种分区策略，一种是基于Hash Partition方式，在这种分区方式下用户数据可较均匀的分布在各个Tablet中，但原来的数据排序特点已被打乱。另外一种是基于Range Partition方式，数据将按照用户数据指定的有序的Primary Key Columns的组合String的顺序进行分区。而HBase中仅仅提供了一种按用户数据RowKey的Range Partition方式。



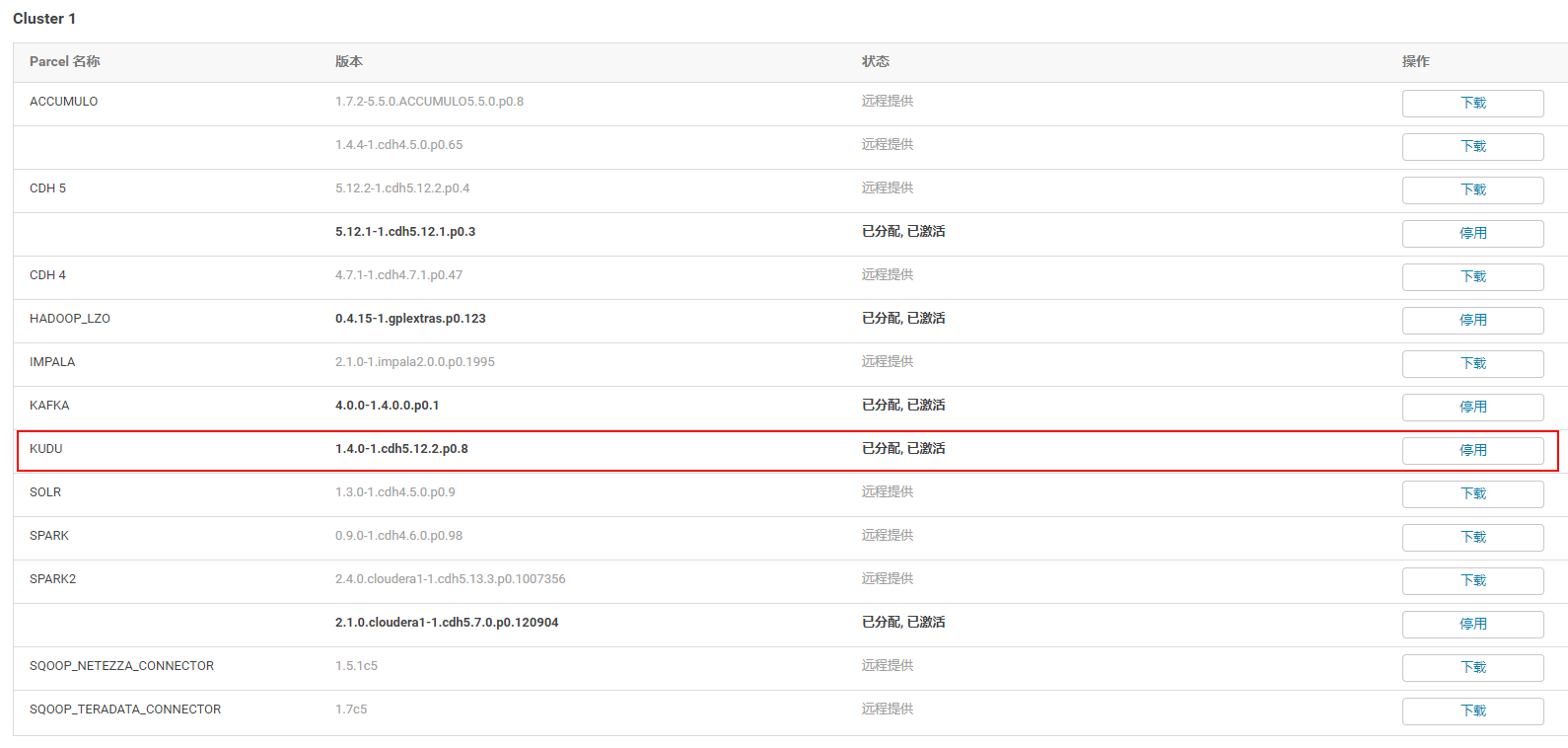
第2章 Kudu快速入门

2.1 安装

2.1.1 点击主机下面的Parcel



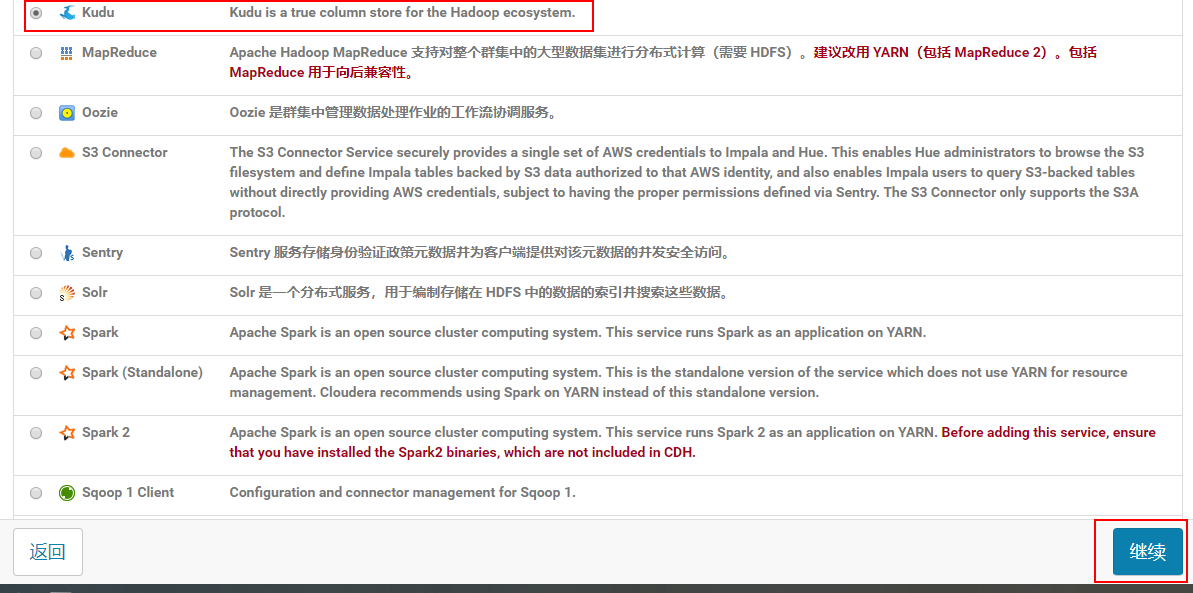
2.1.2 点击KUDU对应的下载，下载完后点击分配、激活



2.1.3 回到首页点击添加服务



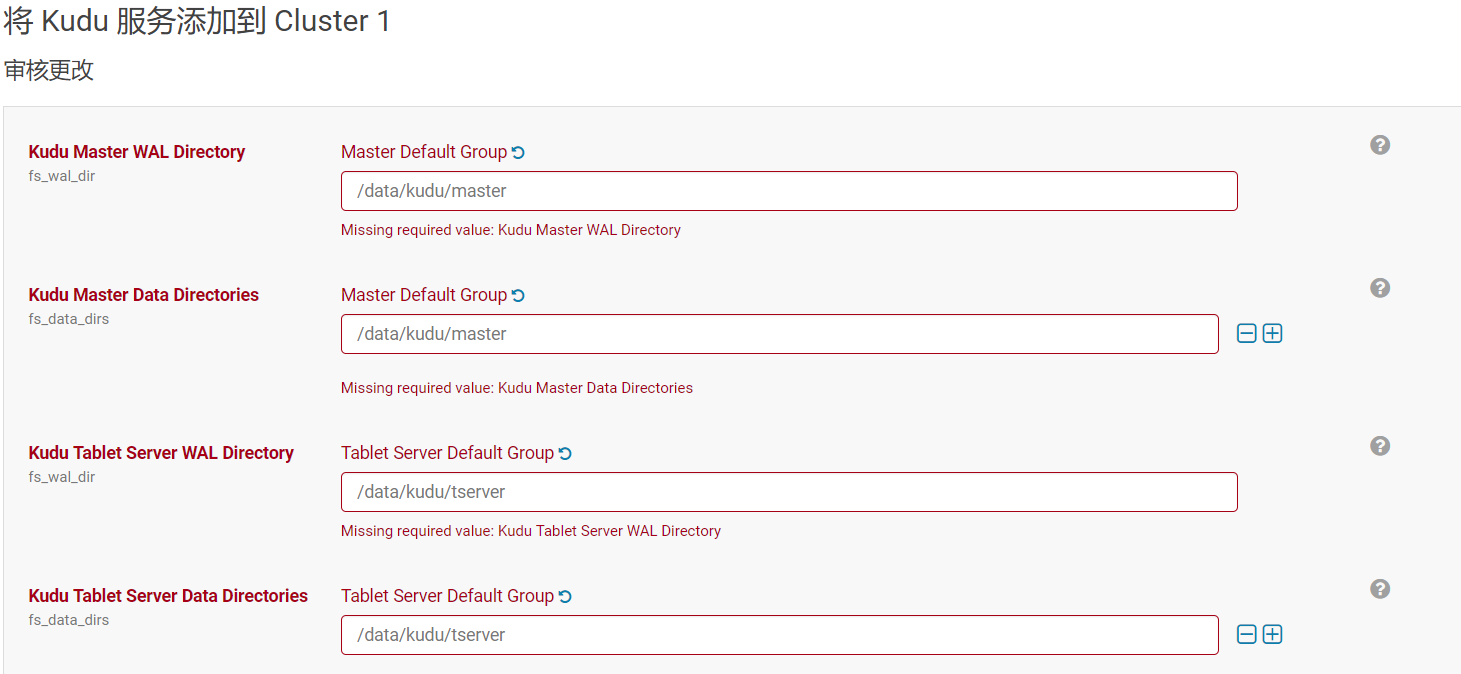
2.1.4 选择KUDU 选择继续



2.1.5 分配角色



2.1.6 设置master和Tablet路径



2.1.7 启动过程出现错误

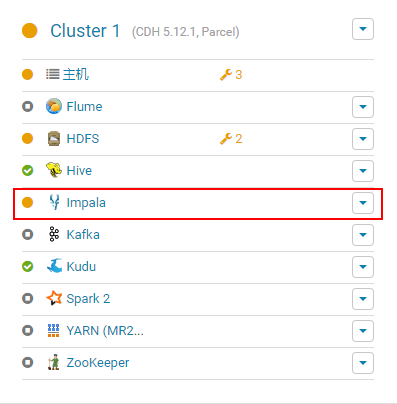
启动kudu失败，错误日志提示：Service unavailable: Cannot initialize clock: Error reading clock. Clock considered unsynchronized

解决： 检查ntp服务，安装或重启ntp服务

/etc/init.d/ntpd restart

2.2 配置impala支持kudu

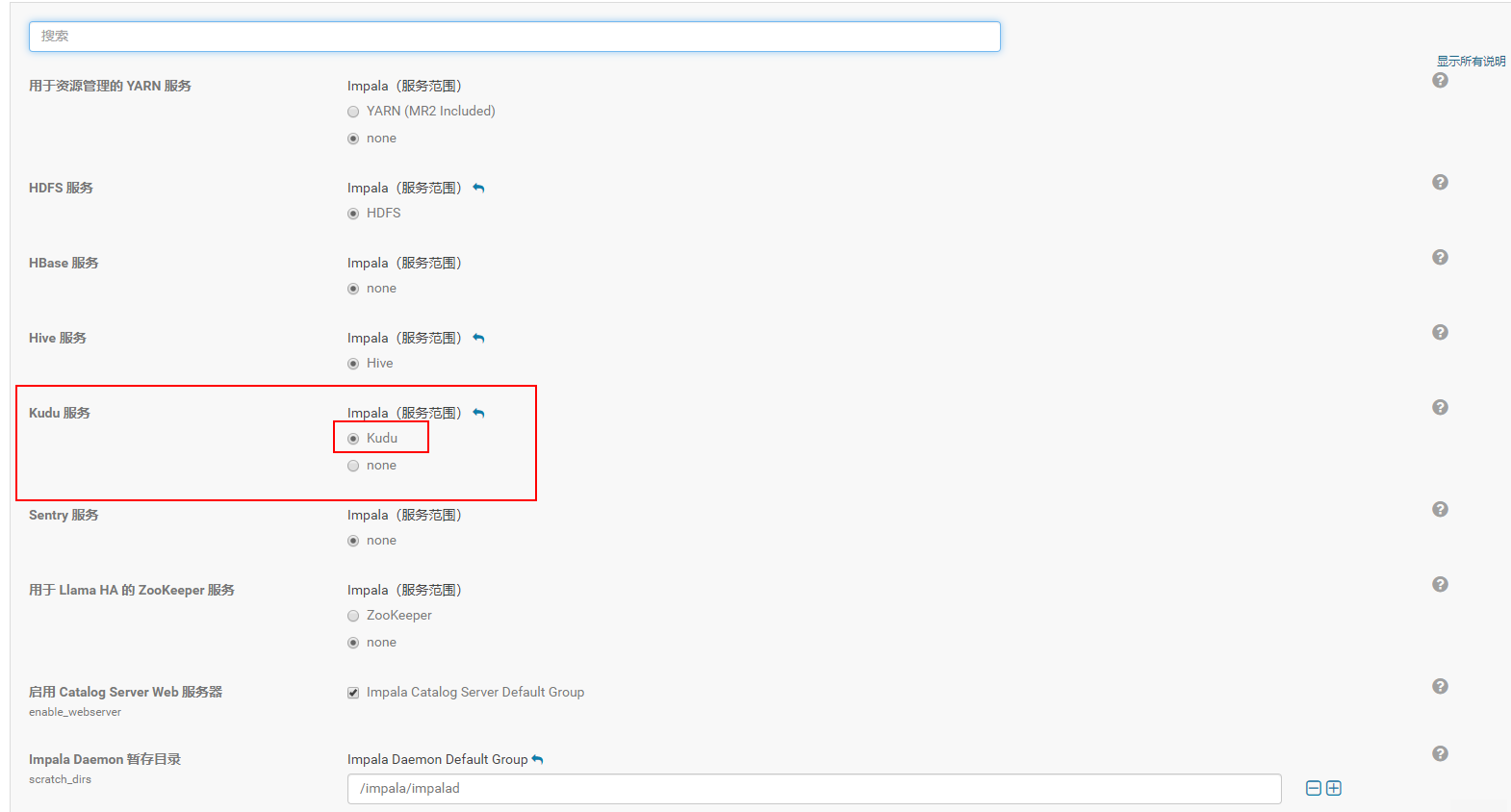
2.2.1 点击impala



2.2.2 点击配置



2.2.3 找到Kudu服务，选择Kudu后重启impala



2.3 使用案例

2.3.1 创建表

从 Impala 在 Kudu 中创建一个新表类似于将现有的 Kudu 表映射到 Impala 表，但需要自己指定模式和分区信息。

在 CREATE TABLE 语句中，必须首先列出构成主键的列。此外，主键列隐式标记为 NOT NULL 。

创建新的 Kudu 表时，需要指定一个分配方案。

CREATE TABLE kudu\_table(

id INT,

name STRING,

PRIMARY KEY(id)

)

PARTITION BY HASH PARTITIONS 16

STORED AS KUDU;

2.3.2 查询 Impala 中现有的 Kudu 表

通过 Kudu API 或其他集成（如 Apache Spark ）创建的表不会在 Impala 中自动显示。要查询它们，必须先在 Impala 中创建外部表以将 Kudu 表映射到 Impala 数据库中：

CREATE EXTERNAL TABLE my\_mapping\_table

STORED AS KUDU

TBLPROPERTIES (

'kudu.table\_name' = 'kudu中的tableName'

);

2.3.3使用 CREATE TABLE ... AS SELECT 语句查询 Impala 中的任何其他表或来创建表。

以下示例将现有表 old\_table 中的所有行导入到 Kudu 表 new\_table 中。 new\_table 中的列的名称和类型将根据 SELECT 语句的结果集中的列确定。

注意，必须另外指定主键和分区。

CREATE TABLE new\_table

PRIMARY KEY (id)

PARTITION BY HASH(id) PARTITIONS 8

STORED AS KUDU

AS SELECT id, name FROM old\_table;

2.3.4 不支持 Kudu 表的 Impala 关键字

创建 Kudu 表时不支持以下 Impala 关键字：

- PARTITIONED

- LOCATION

- ROW FORMAT

2.3.5 将数据插入 Kudu 表

Impala 允许使用 SQL 语句将数据插入 Kudu表 。

插入单个值：

INSERT INTO table\_name VALUES (1001, "zhangsan");

插入多个值：

INSERT INTO table\_name VALUES (1002, "lisi"), (3, "wangwu");

插入其他表的值：

INSERT INTO table\_name select \* from other\_table;

注意：kudu表的update操作不能更改主键的值，其他与标准SQL语法相同。

第3章 API操作（了解）

3.1 添加依赖

首先创建一个maven工程，添加一下依赖

<dependency>

<groupId>org.apache.kudu</groupId>

<artifactId>kudu-client</artifactId>

<version>1.4.0</version>

</dependency>

3.2 创建表

import org.apache.kudu.ColumnSchema;

import org.apache.kudu.Schema;

import org.apache.kudu.Type;

import org.apache.kudu.client.CreateTableOptions;

import org.apache.kudu.client.KuduClient;

import org.apache.kudu.client.KuduException;

import java.util.LinkedList;

import java.util.List;

public class CreateTable {

private static ColumnSchema newColumn(String name, Type type, boolean iskey) {

ColumnSchema.ColumnSchemaBuilder column = new ColumnSchema.ColumnSchemaBuilder(name, type);

column.key(iskey);

return column.build();

}

public static void main(String[] args) throws KuduException {

// master地址

String masteraddr = "hadoop102,hadoop103,hadoop104";

// 创建kudu的数据库链接

KuduClient client = new KuduClient.KuduClientBuilder(masteraddr).defaultSocketReadTimeoutMs(6000).build();

// 设置表的schema

List<ColumnSchema> columns = new LinkedList<ColumnSchema>();

/\*\*

与 RDBMS 不同，Kudu 不提供自动递增列功能，因此应用程序必须始终在插入期间提供完整的主键

\*/

columns.add(newColumn("id", Type.INT32, true));

columns.add(newColumn("name", Type.STRING, false));

Schema schema = new Schema(columns);

//创建表时提供的所有选项

CreateTableOptions options = new CreateTableOptions();

// 设置表的replica备份和分区规则

List<String> parcols = new LinkedList<String>();

parcols.add("id");

//设置表的备份数

options.setNumReplicas(1);

//设置hash分区和数量

options.addHashPartitions(parcols, 3);

try {

client.createTable("student", schema, options);

} catch (KuduException e) {

e.printStackTrace();

} finally {

client.close();

}

}

}

3.3 删除表

import org.apache.kudu.client.KuduClient;

import org.apache.kudu.client.KuduException;

public class DropTable {

public static void main(String[] args) throws KuduException {

String masterAddress = "hadoop102,hadoop103,hadoop104";

KuduClient client = new KuduClient.KuduClientBuilder(masterAddress).defaultSocketReadTimeoutMs(6000).build();

try {

client.deleteTable("student");

} catch (KuduException e) {

e.printStackTrace();

} finally {

client.close();

}

}

}

3.4 插入数据

import org.apache.kudu.client.\*;

public class InsertRow {

public static void main(String[] args) throws KuduException {

String masterAddr = "hadoop102,haoop103,hadoop104";

KuduClient client = new KuduClient.KuduClientBuilder(masterAddr).defaultSocketReadTimeoutMs(6000).build();

try {

KuduTable table = client.openTable("student");

KuduSession kuduSession = client.newSession();

kuduSession.setFlushMode(SessionConfiguration.FlushMode.MANUAL\_FLUSH);

kuduSession.setMutationBufferSpace(3000);

for (int i = 1; i < 10; i++) {

Insert insert = table.newInsert();

insert.getRow().addInt("id", i);

insert.getRow().addString("name", i + "号");

kuduSession.flush();

kuduSession.apply(insert);

}

kuduSession.close();

} catch (KuduException e) {

e.printStackTrace();

} finally {

client.close();

}

}

}

3.5 查询数据

public class FindRow {

import org.apache.kudu.client.\*;

public class ScanTable {

public static void main(String[] args) throws KuduException {

// master地址

final String masteraddr = "hadoop102,hadoop103,hadoop104";

// 创建kudu的数据库链接

KuduClient client = new KuduClient.KuduClientBuilder(masteraddr).defaultSocketReadTimeoutMs(6000).build();

//打开kudu表

KuduTable student = client.openTable("student");

//创建scanner扫描

KuduScanner scanner = client.newScannerBuilder(student).build();

//遍历数据

while (scanner.hasMoreRows()){

for (RowResult rowResult : scanner.nextRows()) {

System.out.println(rowResult.getInt("id") + "\t" + rowResult.getString("name")) ;

}

}

}

}

3.6 更改表数据

import org.apache.kudu.client.\*;

public class UpdateTable {

public static void main(String[] args) throws KuduException{

// master地址

String masteraddr = "hadoop102,hadoop103,hadoop104";

// 创建kudu的数据库链接

KuduClient client = new KuduClient.KuduClientBuilder(masteraddr).build();

// 打开表

KuduSession session = null;

try {

KuduTable table = client.openTable("student");

session = client.newSession();

session.setFlushMode(SessionConfiguration.FlushMode.AUTO\_FLUSH\_SYNC);

//更新数据

Update update = table.newUpdate();

PartialRow row = update.getRow();

row.addInt("id", 1);

row.addString("name", "di");

session.apply(update);

} catch (KuduException e) {

e.printStackTrace();

} finally {

session.close();

client.close();

}

}

}

3.7 **删除指定行**

public class DeleteRow {

public static void main(String[] args) throws KuduException {

// master地址

String masteraddr = "hadoop102,hadoop103,hadoop104";

// 创建kudu的数据库链接

KuduClient client = new KuduClient.KuduClientBuilder(masteraddr).build();

// 打开表

KuduTable table = client.openTable("student");

// 创建写session,kudu必须通过session写入

KuduSession session = client.newSession();

final Delete delete = table.newDelete();

//TODO 注意：行删除和更新操作必须指定要更改的行的完整主键;

delete.getRow().addInt("id" , 5);

session.flush();

session.apply(delete);

session.close();

client.close();

}

}

3.8 获取所有kudu表

import org.apache.kudu.client.KuduClient;

import org.apache.kudu.client.KuduException;

import org.apache.kudu.client.ListTablesResponse;

import java.util.List;

public class ShowTables {

public static void main(String[] args) throws KuduException {

// master地址

final String masteraddr = "hadoop102,hadoop103,hadoop104";

// 创建kudu的数据库链接

KuduClient client = new KuduClient.KuduClientBuilder(masteraddr).defaultSocketReadTimeoutMs(6000).build();

try {

//获取现有表的列表

ListTablesResponse tablesList = client.getTablesList();

List<String> tablesList1 = tablesList.getTablesList();

//遍历列表中所有信息

for (String s : tablesList1) {

System.out.println(s);

}

} catch (KuduException e) {

e.printStackTrace();

} finally {

client.close();

}

}

}