尚硅谷大数据技术之Pheonix

(作者：尚硅谷大数据研发部)

版本：V1.2

# 第1章 Pheonix简介

## 1.1 Pheonix定义

Phoenix是HBase的开源SQL皮肤。可以使用标准JDBC API代替HBase客户端API来创建表，插入数据和查询HBase数据。

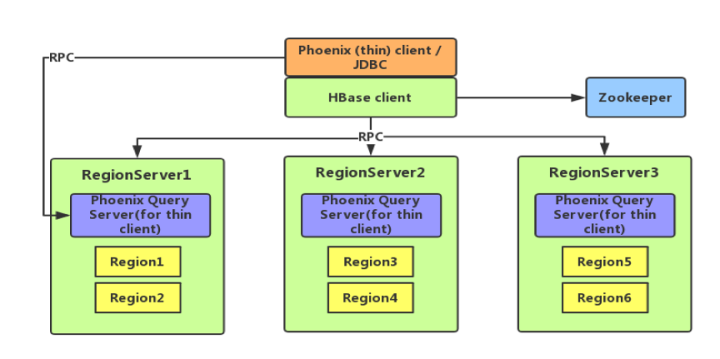
## 1.2 Pheonix特点

1）容易集成：如Spark，Hive，Pig，Flume和Map Reduce；

2）操作简单：DML命令以及通过DDL命令创建和操作表和版本化增量更改；

3）支持HBase二级索引创建。

## 1.3 Pheonix架构



# 第2章 Pheonix快速入门

## 2.1 安装

### 2.1.1 官网地址

<http://phoenix.apache.org/>

### 2.1.2 Pheonix部署

1）上传并解压tar包

[atguigu@hadoop101 module]$ tar -zxvf /opt/software/apache-phoenix-4.14.2-HBase-1.3-bin.tar.gz -C /opt/module

[atguigu@hadoop101 module]$ mv apache-phoenix-4.14.2-HBase-1.3-bin phoenix

2）复制server包并拷贝到各个节点的hbase/lib

[atguigu@hadoop102 module]$ cd /opt/module/phoenix/

[atguigu@hadoop102 phoenix]$ cp phoenix-4.14.2-HBase-1.3-server.jar /opt/module/hbase/lib/

[atguigu@hadoop102 phoenix]$ scp phoenix-4.14.2-HBase-1.3-server.jar hadoop103:/opt/module/hbase/lib/

[atguigu@hadoop102 phoenix]$ scp phoenix-4.14.2-HBase-1.3-server.jar hadoop104:/opt/module/hbase/lib/

3）复制client包并拷贝到各个节点的hbase/lib

[atguigu@hadoop102 module]$ cd /opt/module/phoenix/

[atguigu@hadoop102 phoenix]$ cp phoenix-4.14.2-HBase-1.3-client.jar /opt/module/hbase/lib/

[atguigu@hadoop102 phoenix]$ scp phoenix-4.14.2-HBase-1.3-client.jar hadoop103:/opt/module/hbase/lib/

[atguigu@hadoop102 phoenix]$ scp phoenix-4.14.2-HBase-1.3-client.jar hadoop104:/opt/module/hbase/lib/

4）配置环境变量

#phoenix

export PHOENIX\_HOME=/opt/module/phoenix

export PHOENIX\_CLASSPATH=$PHOENIX\_HOME

export PATH=$PATH:$PHOENIX\_HOME/bin

5）启动Pheonix

[atguigu@hadoop101 phoenix]$ /opt/module/phoenix/bin/sqlline.py hadoop102,hadoop103,hadoop104:2181

## 2.2 Pheonix Shell操作

### 2.2.1 表的操作

1）显示所有表

!table 或 !tables

2）创建表

直接指定单个列作为RowKey

CREATE TABLE IF NOT EXISTS student(

id VARCHAR primary key,

name VARCHAR;

在phoenix中，表名等会自动转换为大写，若要小写，使用双引号，如"us\_population"。

指定多个列的联合作为RowKey

CREATE TABLE IF NOT EXISTS us\_population (

State CHAR(2) NOT NULL,

City VARCHAR NOT NULL,

Population BIGINT

CONSTRAINT my\_pk PRIMARY KEY (state, city));

3）插入数据

upsert into student values('1001','zhangsan');

4）查询记录

select \* from student;

select \* from student where id='1001';

5）删除记录

delete from student where id='1001';

6）删除表

drop table student;

7）退出命令行

!quit

### 2.2.2 表的映射

1）表的关系

默认情况下，直接在HBase中创建的表，通过Phoenix是查看不到的。如果要在Phoenix中操作直接在HBase中创建的表，则需要在Phoenix中进行表的映射。映射方式有两种：视图映射和表映射。

2）命令行中创建表test

HBase 中test的表结构如下，两个列族info1、info2。

|  |  |  |
| --- | --- | --- |
| Rowkey | info1 | info2 |
| id | name | address |

启动HBase Shell

[atguigu@hadoop101 ~]$ /opt/module/hbase-1.3.1/bin/hbase shell

创建HBase表test

hbase(main):001:0> create 'test','info1','info2'

3）视图映射

Phoenix创建的视图是只读的，所以只能用来做查询，无法通过视图对源数据进行修改等操作。在phoenix中创建关联test表的视图

0: jdbc:phoenix:hadoop101,hadoop102,hadoop103> create view "test"(id varchar primary key,"info1"."name" varchar, "info2"."address" varchar);

删除视图

0: jdbc:phoenix:hadoop101,hadoop102,hadoop103> drop view "test";

4）表映射

使用Apache Phoenix创建对HBase的表映射，有两种方法：

（1）HBase中不存在表时，可以直接使用create table指令创建需要的表,系统将会自动在Phoenix和HBase中创建person\_infomation的表，并会根据指令内的参数对表结构进行初始化。

（2）当HBase中已经存在表时，可以以类似创建视图的方式创建关联表，只需要将create view改为create table即可。

0: jdbc:phoenix:hadoop101,hadoop102,hadoop103> create table "test"(id varchar primary key,"info1"."name" varchar, "info2"." archar) column\_encoded\_bytes=0;

## 2.3 Pheonix JDBC操作

### 2.3.1 创建项目并导入依赖

<dependencies>

<dependency>

<groupId>org.apache.phoenix</groupId>

<artifactId>phoenix-core</artifactId>

<version>4.14.2-HBase-1.3</version>

</dependency>

</dependencies>

### 2.3.2 编写代码

package com.atguigu;

import java.sql.\*;

public class PhoenixTest {

public static void main(String[] args) throws ClassNotFoundException, SQLException {

//1.定义参数

String driver = "org.apache.phoenix.jdbc.PhoenixDriver";

String url = "jdbc:phoenix:hadoop102,hadoop103,hadoop104:2181";

//2.加载驱动

Class.forName(driver);

//3.创建连接

Connection connection = DriverManager.getConnection(url);

//4.预编译SQL

PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM test");

//5.查询获取返回值

ResultSet resultSet = preparedStatement.executeQuery();

//6.打印结果

while (resultSet.next()) {

System.out.println(resultSet.getString(1) + resultSet.getString(2));

}

//7.关闭资源

resultSet.close();

preparedStatement.close();

connection.close();

}

}

# 第3章 Pheonix二级索引

## 3.1 HBase协处理器

### 3.1.1 案例需求

编写协处理器，实现在往A表插入数据的同时让HBase自身向B表中插入一条数据。

### 3.1.2 实现步骤

1）创建一个maven项目，并引入以下依赖。

<dependencies>

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-client</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-server</artifactId>

<version>1.3.1</version>

</dependency>

</dependencies>

2）定义FruitTableCoprocessor类并继承BaseRegionObserver类

package com.atguigu;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.TableName;

import org.apache.hadoop.hbase.client.\*;

import org.apache.hadoop.hbase.coprocessor.BaseRegionObserver;

import org.apache.hadoop.hbase.coprocessor.ObserverContext;

import org.apache.hadoop.hbase.coprocessor.RegionCoprocessorEnvironment;

import org.apache.hadoop.hbase.regionserver.wal.WALEdit;

import java.io.IOException;

public class FruitTableCoprocessor extends BaseRegionObserver {

@Override

public void postPut(ObserverContext<RegionCoprocessorEnvironment> e, Put put, WALEdit edit, Durability durability) throws IOException {

//获取连接

Connection connection = ConnectionFactory.createConnection(HBaseConfiguration.create());

//获取表对象

Table table = connection.getTable(TableName.valueOf("fruit"));

//插入数据

table.put(put);

//关闭资源

table.close();

connection.close();

}

}

## 3.2 二级索引配置文件

### 3.2.1 添加如下配置到HBase的HRegionserver节点的hbase-site.xml

<!-- phoenix regionserver 配置参数-->

<property>

<name>hbase.regionserver.wal.codec</name>

<value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</value>

</property>

<property>

<name>hbase.region.server.rpc.scheduler.factory.class</name>

<value>org.apache.hadoop.hbase.ipc.PhoenixRpcSchedulerFactory</value>

<description>Factory to create the Phoenix RPC Scheduler that uses separate queues for index and metadata updates</description>

</property>

<property>

<name>hbase.rpc.controllerfactory.class</name>

<value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</value>

<description>Factory to create the Phoenix RPC Scheduler that uses separate queues for index and metadata updates</description>

</property>

### 3.2.2 添加如下配置到HBase中HMaster节点的hbase-site.xml中

<!-- phoenix master 配置参数 -->

<property>

<name>hbase.master.loadbalancer.class</name>

<value>org.apache.phoenix.hbase.index.balancer.IndexLoadBalancer</value>

</property>

<property>

<name>hbase.coprocessor.master.classes</name>

<value>org.apache.phoenix.hbase.index.master.IndexMasterObserver</value>

</property>

## 3.3 全局二级索引

Global Index是默认的索引格式，创建全局索引时，会在HBase中建立一张新表。也就是说索引数据和数据表是存放在不同的表中的，因此全局索引适用于多读少写的业务场景。

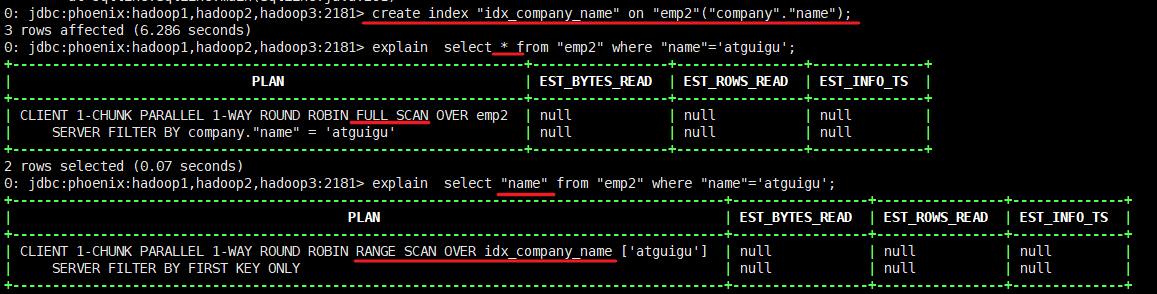
写数据的时候会消耗大量开销，因为索引表也要更新，而索引表是分布在不同的数据节点上的，跨节点的数据传输带来了较大的性能消耗。

在读数据的时候Phoenix会选择索引表来降低查询消耗的时间。

1）创建单个字段的全局索引

CREATE INDEX my\_index ON my\_table (my\_col);

如果想查询的字段不是索引字段的话索引表不会被使用，也就是说不会带来查询速度的提升。



2）创建携带其他字段的全局索引

CREATE INDEX my\_index ON my\_table (v1) INCLUDE (v2)

## 3.4 本地二级索引

Local Index适用于写操作频繁的场景。

索引数据和数据表的数据是存放在同一张表中（且是同一个Region），避免了在写操作的时候往不同服务器的索引表中写索引带来的额外开销。查询的字段不是索引字段索引表也会被使用，这会带来查询速度的提升。

CREATE LOCAL INDEX my\_index ON my\_table (my\_index);