## 第 1 章 Phoenix 简介



### 1.1 Phoenix 定义

Phoenix 最早是 saleforce 的一个开源项目，后来成为 Apache 的顶级项目。

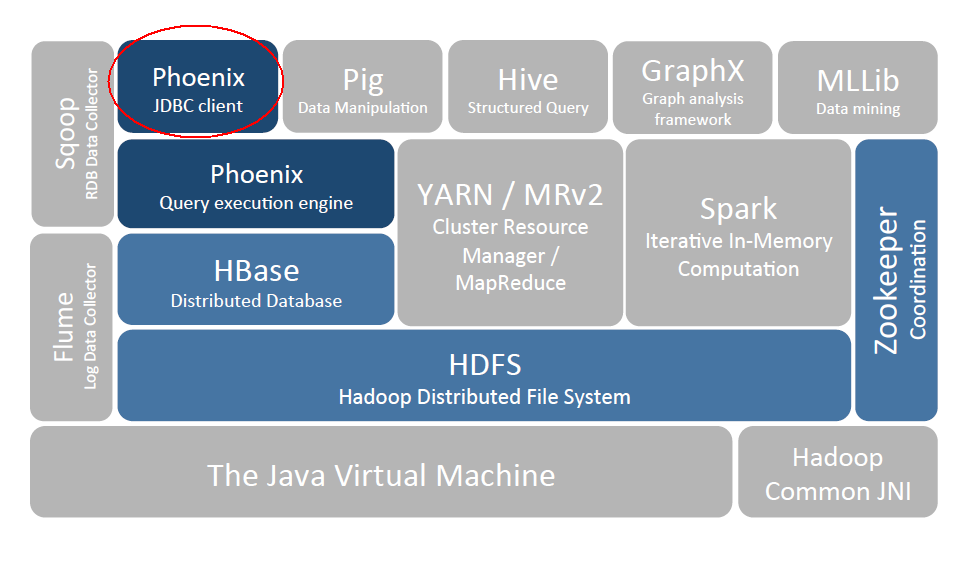
Phoenix 构建在 HBase 之上的开源 SQL 层. 能够让我们使用标准的 JDBC API 去建表, 插入数据和查询 HBase 中的数据, 从而可以避免使用 HBase 的客户端 API.

在我们的应用和 HBase 之间添加了 Phoenix, 并不会降低性能, 而且我们也少写了很多代码.

### 1.2 Phoenix 特点

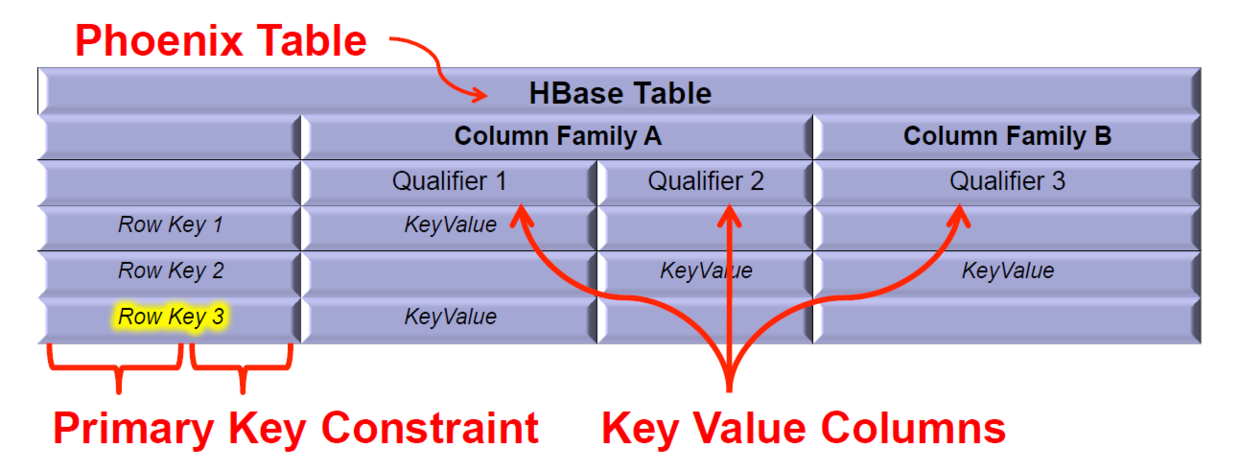
1. 将 SQl 查询编译为 HBase 扫描
2. 确定扫描 Rowkey 的最佳开始和结束位置
3. 扫描并行执行
4. 将 where 子句推送到服务器端的过滤器【hbase中有很多filter】
5. 通过协处理器进行聚合操作【协处理器运行在hbase的rs上，predo,posdo】
6. 完美支持 HBase 二级索引创建
7. DML命令以及通过DDL命令创建和操作表和版本化增量更改。【增删改查】
8. 容易集成：如Spark，Hive，Pig，Flume和Map Reduce。

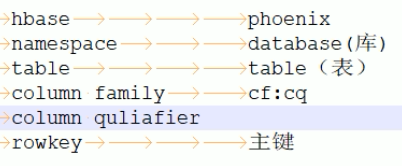
### 1.3 Phoenix 架构



### 1.4 Phoenix 数据存储

Phoenix 将 HBase 的数据模型映射到关系型世界





## 第 2 章 Phoenix 快速入门

### 2.1 安装部署

#### 步骤 1: 下载 Phoenix

http://archive.apache.org/dist/phoenix/apache-phoenix-4.14.2-HBase-1.3/

#### 步骤 2: 解压 jar 包

tar -zxvf apache-phoenix-4.14.2-HBase-1.3-bin.tar.gz -C /opt/module  
  
mv apache-phoenix-4.14.2-HBase-1.3-bin phoenix

#### 步骤 3: 复制 jar 包

复制 HBase 需要用到 server 和 client 2 个 jar 包

cp phoenix-4.14.2-HBase-1.3-server.jar /opt/module/hbase-1.3.1/lib  
cp phoenix-4.14.2-HBase-1.3-client.jar /opt/module/hbase-1.3.1/lib

#### 步骤 4: 分发 jar 包

需要把刚才 copy 的 2个jar 包分发到其他 HBase 节点

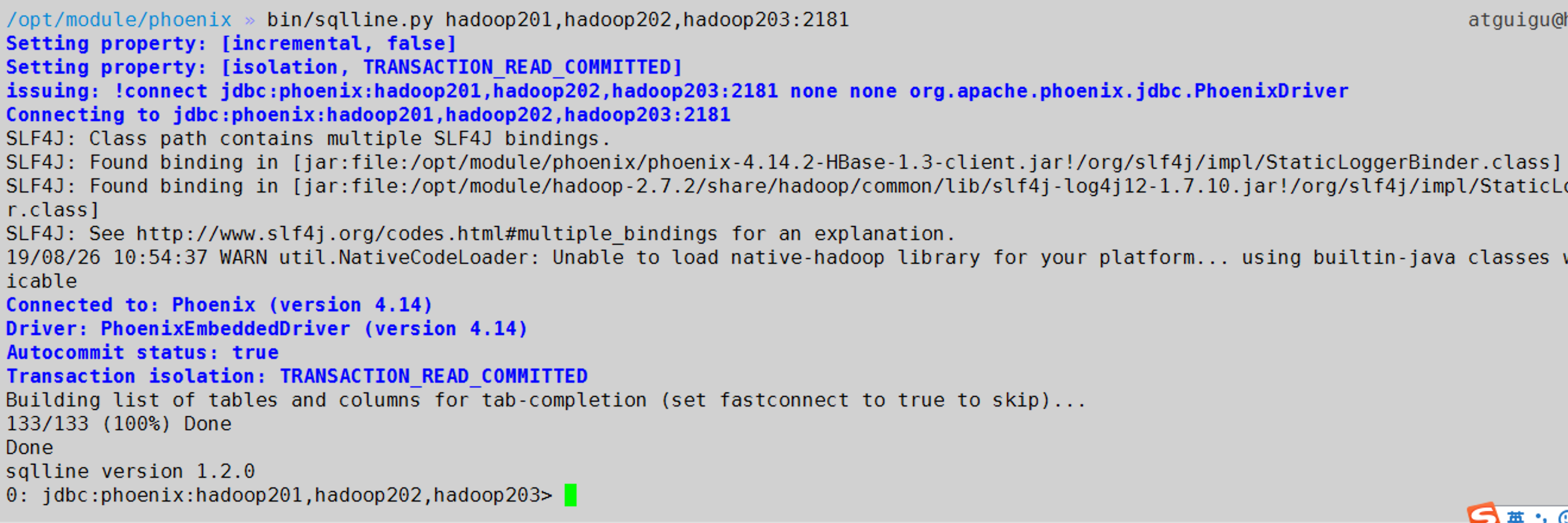
#### 步骤 5: 配置环境变量

export PHOENIX\_HOME=/opt/module/phoenix  
export PHOENIX\_CLASSPATH=$PHOENIX\_HOME  
export PATH=$PATH:$PHOENIX\_HOME/bin

#### 步骤 6: 启动 hadoop, zookeeper, HBase

#### 步骤 7: 启动 Phoenix

/opt/module/phoenix » bin/sqlline.py hadoop201,hadoop202,hadoop203:2181



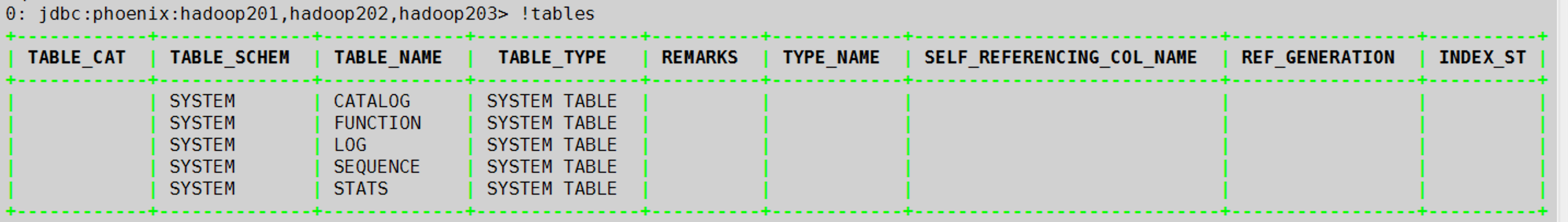
！quit 退出

### 2.2 Phoenix 表操作

【需要和HBase一一对应】

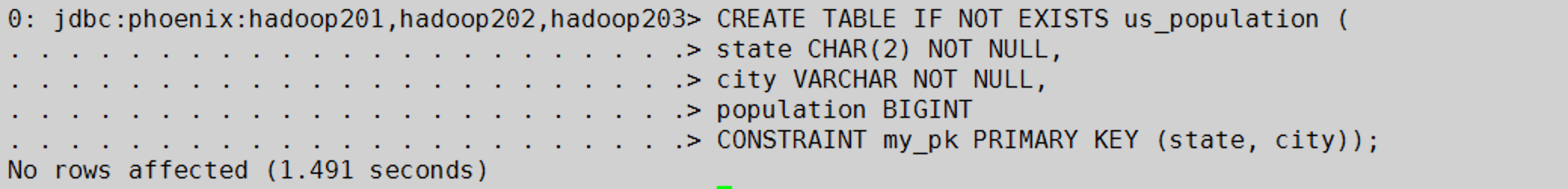
#### 2.2.1 显示所有表

!tables  
  
# 或者  
!table



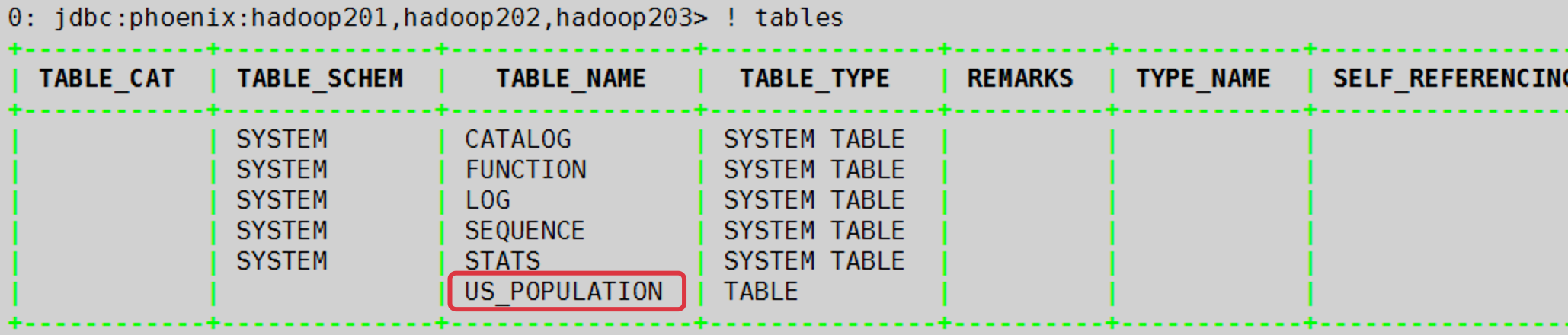
#### 2.2.2 创建表

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));



说明: 在phoenix中建表在hbase中可以查询到

1. char类型必须添加长度限制
2. varchar 可以不用长度限制
3. 主键映射到 HBase 中会成为 Rowkey. 如果有多个主键(联合主键), 会把多个主键的值拼成 rowkey
4. 在 Phoenix 中, 默认会把表名,字段名等自动转换成大写. 如果要使用小写, 需要把他们用双引号括起来.【upsert 中value后面的值用单引号，双引号会报错】
5. Sql末尾有分号；



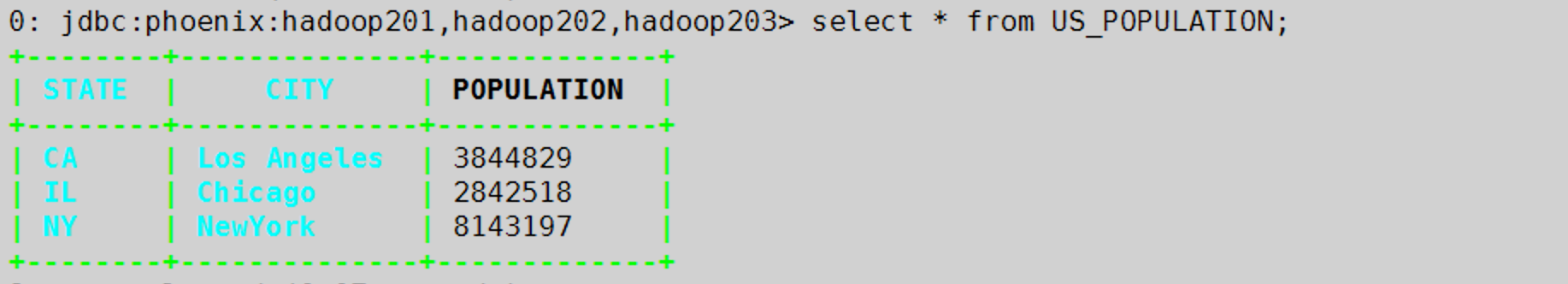
#### 2.2.3 插入记录

upsert into us\_population values('NY','NewYork',8143197);  
upsert into us\_population values('CA','Los Angeles',3844829);  
upsert into us\_population values('IL','Chicago',2842518);

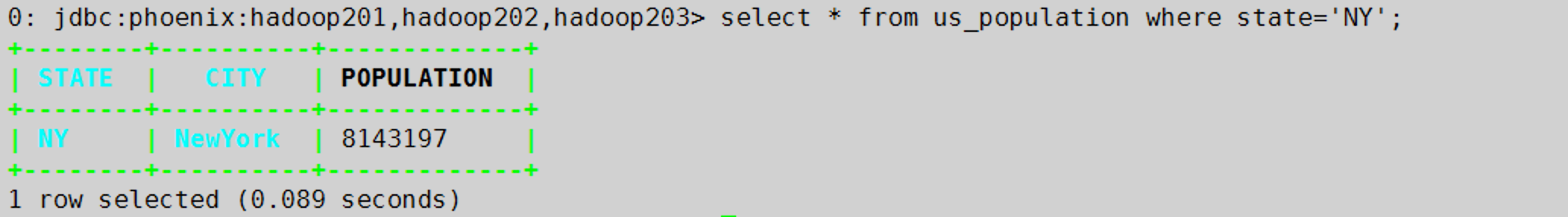
说明: upset可以看成是update和insert的结合体.

#### 2.2.4 查询记录

select \* from US\_POPULATION;

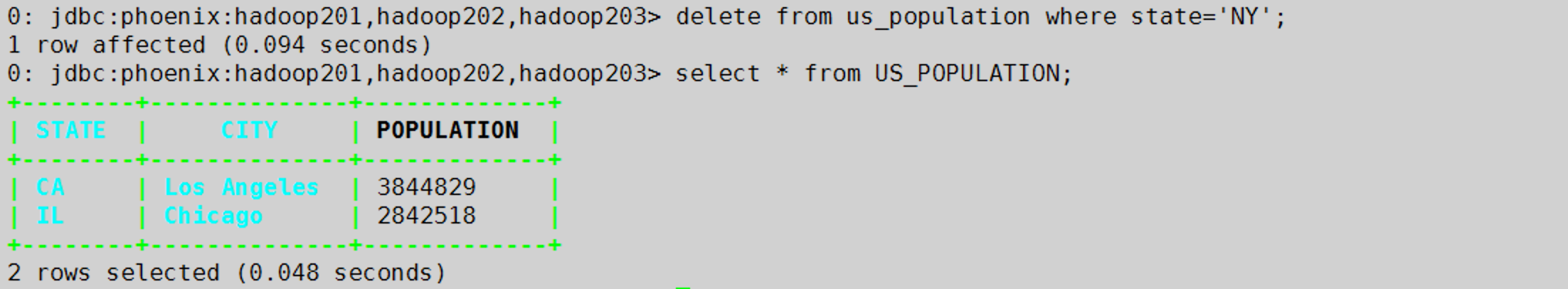


select \* from us\_population where state='NY';



#### 2.2.5 删除记录

delete from us\_population where state='NY';



#### 2.2.5 删除表

drop table us\_population;

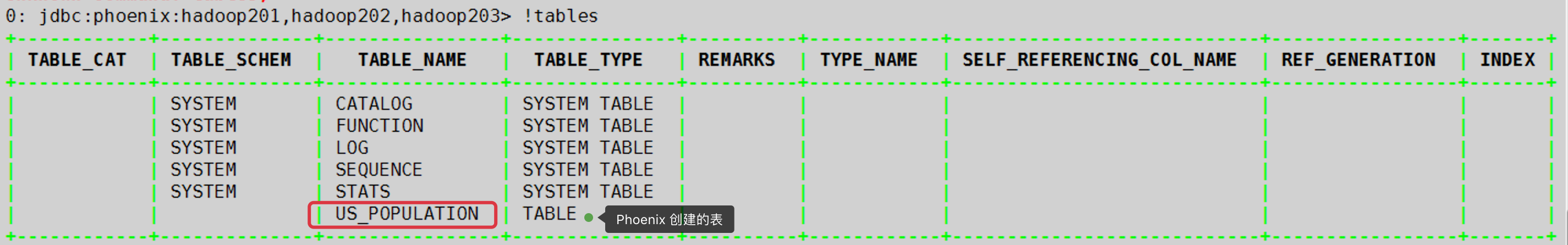
#### 2.2.6 退出命令行

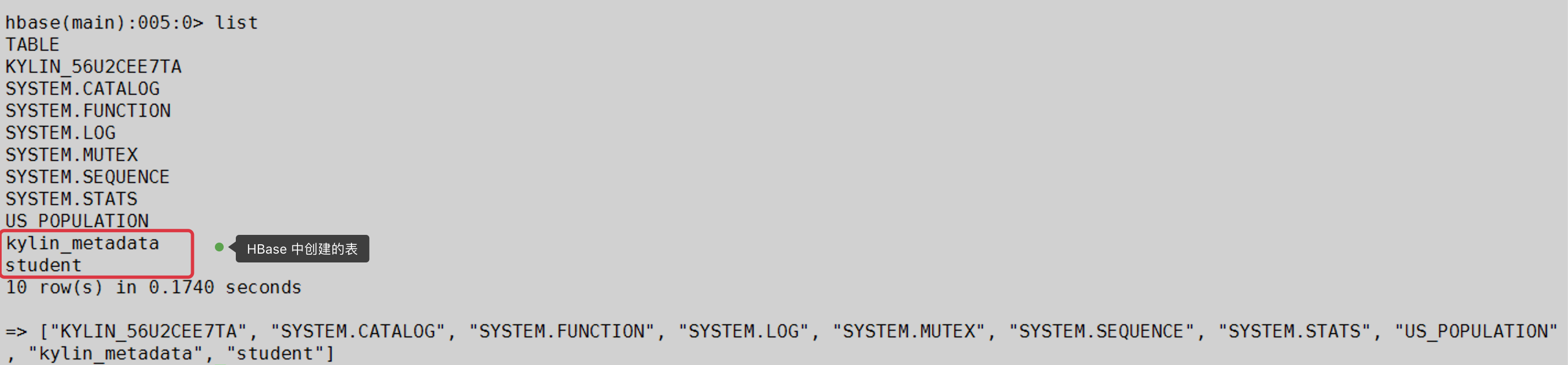
! quit

### 2.3 Phoenix 表映射

#### 2.3.1 Phoenix 表和 HBase 表的关系

默认情况下, 直接在 HBase 中创建的表通过 Phoenix 是查不到的.



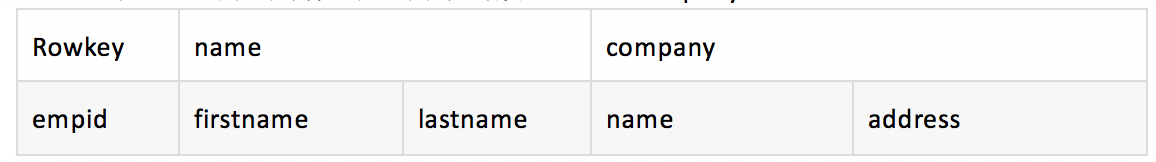


如果要在 Phoenix 中操作直接在 HBase 中创建的表，则需要在 Phoenix 中进行表的映射。

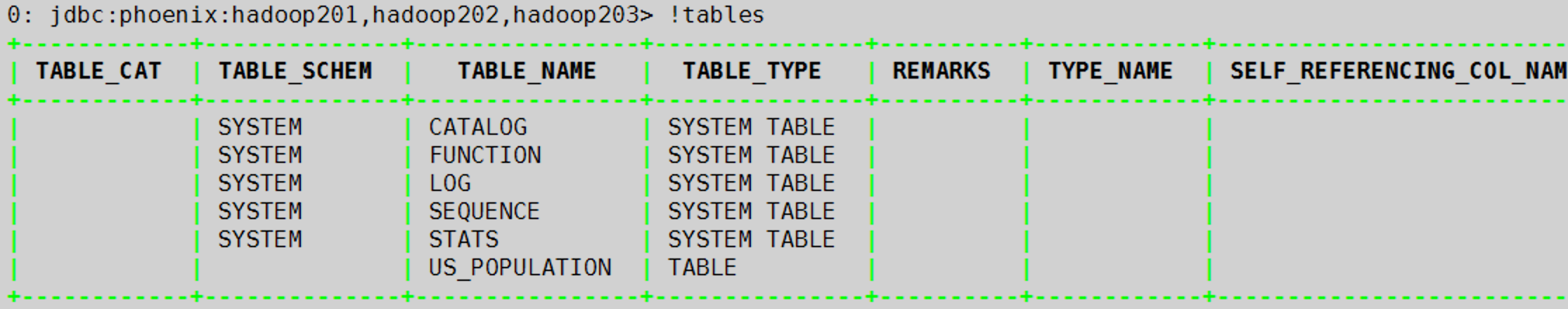
映射方式有两种： 1. 视图映射 2. 表映射

#### 2.3.2 准备工作

在 HBase 中创建表: test, 两个列簇name、company.



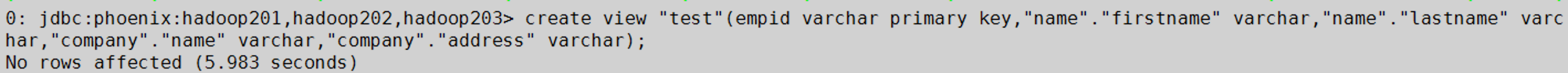
1. 启动 HBase shell
2. 创建表: test

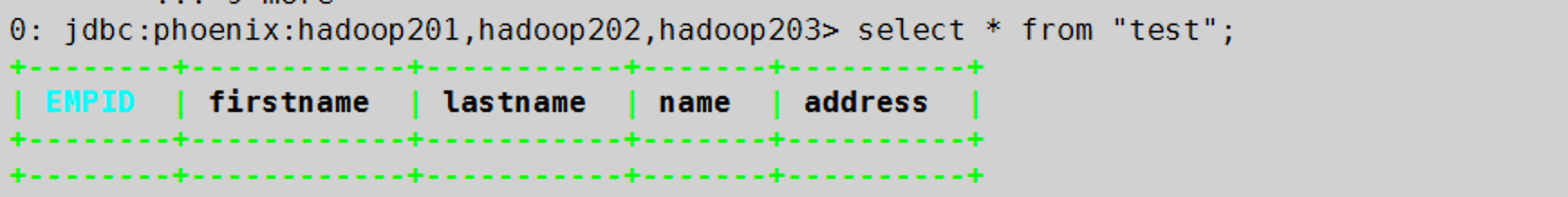
* create 'test', 'name', 'company'
* 在 Phoenix 中是看不到这个表的: 

#### 2.3.3 视图映射

Phoenix 创建的视图是只读的, 所以只能用来查询, 无法通过视图对数据进行修改等操作.

create view "test"(empid varchar primary key,"name"."firstname" varchar,"name"."lastname" varchar,"company"."name" varchar,"company"."address" varchar);





#### 2.3.4 表映射

使用 Phoenix创建对 HBase 的表映射，有两种方法：

1. 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。 在 HBase 中创建表:

create 'test1', 'name', 'company'

在 Phoenix 中创建关联表:

create table "test1"(empid varchar primary key,"name"."firstname" varchar,"name"."lastname" varchar,"company"."name" varchar,"company"."address" varchar) column\_encoded\_bytes=0; // 取消数据编码，不然在hbase中查不到

说明: 添加这个参数之后, 在 HBase 中添加的数据在 Phoenix 中也可以查询到. 否则 HBase 中添加的数据在 Phoenix 中查询不到.

1. 当 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。

#### 2.3.5 视图映射和表映射的对比与总结

相比于直接创建映射表，视图的查询效率会低， 原因是：创建映射表的时候，Phoenix 会在表中创建一些空的键值对，这些空键值对的存在可以用来提高查询效率。

使用create table创建的关联表，如果对表进行了修改，源数据也会改变，同时如果关联表被删除，源表也会被删除。但是视图就不会，如果删除视图，源数据不会发生改变。

## 第 3 章 Phoenix 创建 HBase 二级索引

### 3.1 HBase 的二级索引

在前面的学习中, 我们知道 HBase 只能通过 rowkey 进行搜索, 一般把 rowkey 称作一级索引. 在很长的一段时间里 HBase 就只支持一级索引.

HBase 里面只有 rowkey 作为一级索引， 如果要对库里的非 rowkey 字段进行数据检索和查询， 往往要通过 MapReduce/Spark 等分布式计算框架进行，硬件资源消耗和时间延迟都会比较高。

为了 HBase 的数据查询更高效、适应更多的场景， 诸如使用非 rowkey 字段检索也能做到秒级响应，或者支持各个字段进行模糊查询和多字段组合查询等， 因此需要在 HBase 上面构建二级索引， 以满足现实中更复杂多样的业务需求。

从 0.94 版本开始, HBase 开始支持二级索引.

HBase 索引有多种放方案, 我们今天要做的是使用 Phoenix 给 HBase 添加二级索引.

### 3.2 配置 HBase 支持 Phoenix 创建二级索引

需要先给 HBase 配置支持创建二级索引

#### 步骤 1: 添加如下配置到 HBase 的 Hregionerver 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>

#### 步骤 2: 添加如下配置到 HBase 的 Hmaster 节点的 hbase-site.xml

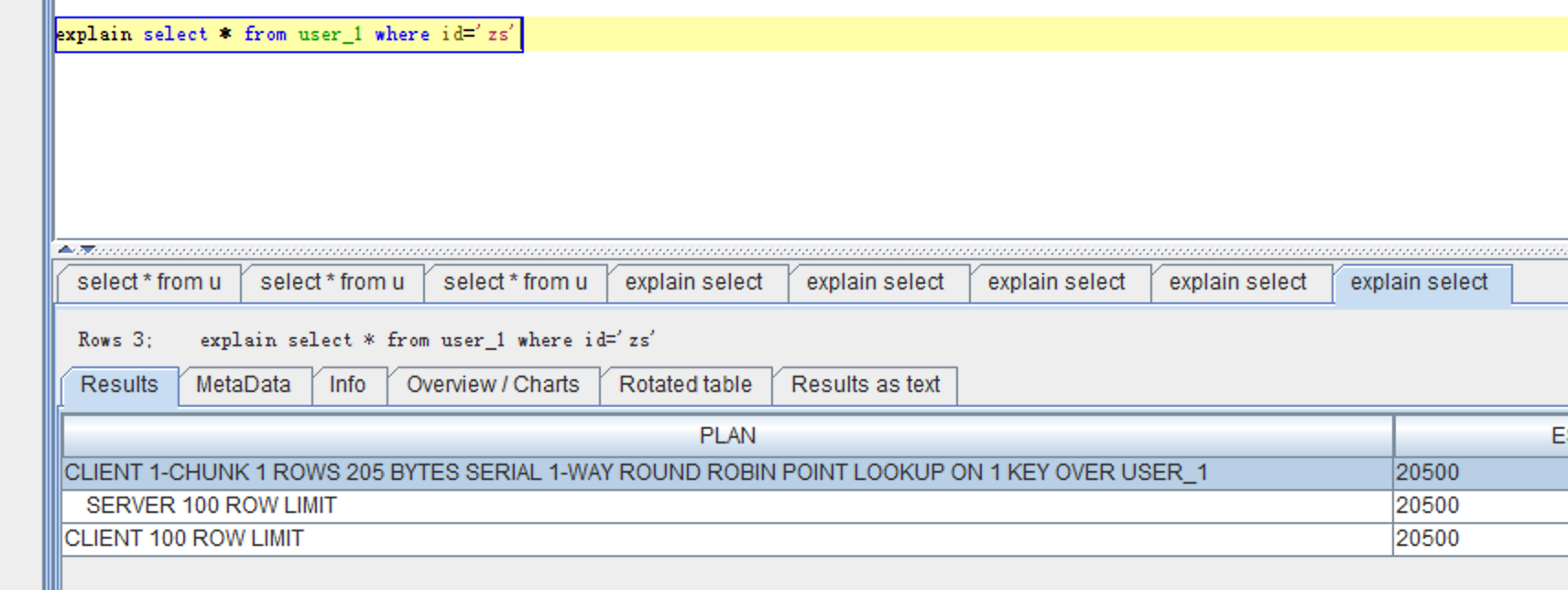
<!-- phoenix master 配置参数 -->  
<property>  
 <name>hbase.master.loadbalancer.class</name>  
 <value>org.apache.phoenix.hbase.index.balancer.IndexLoadBalancer</value>  
</property>  
  
<property>  
 <name>hbase.coprocessor.master.classes</name>  
 <value>org.apache.phoenix.hbase.index.master.IndexMasterObserver</value>  
</property>

#### 步骤 3: 测试是否支持

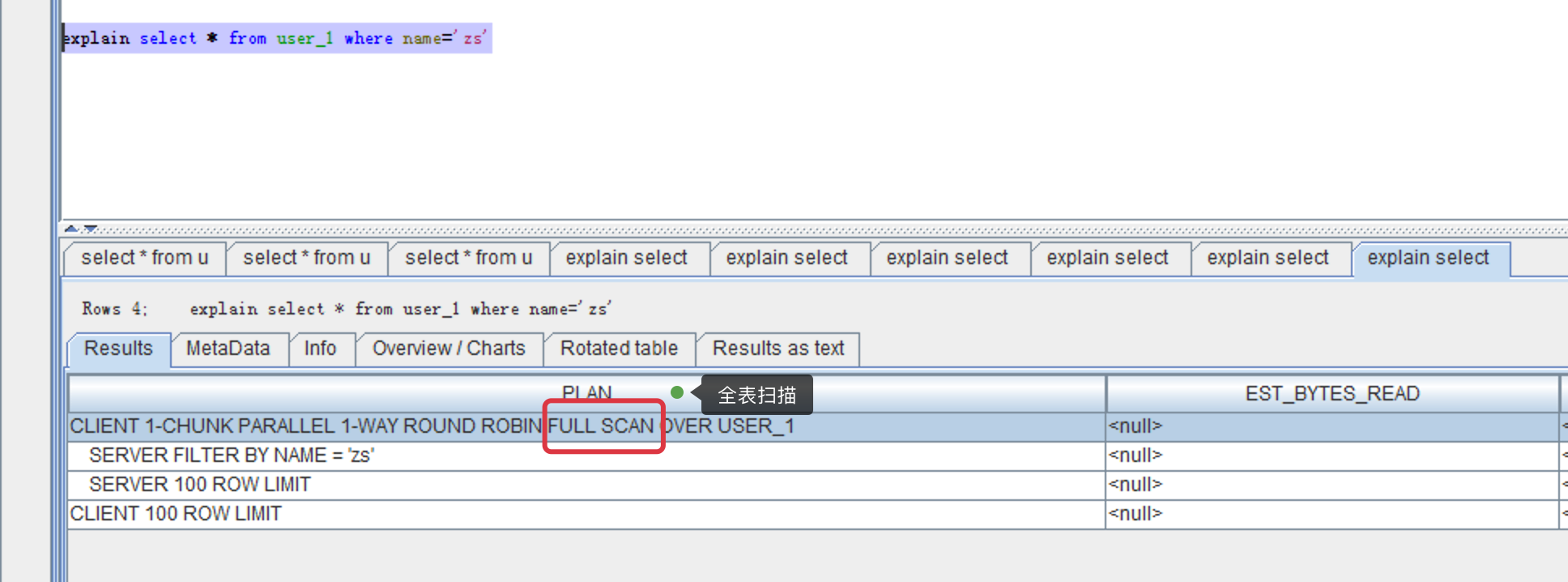
准备数据:

create table user\_1(id varchar primary key, name varchar, addr varchar)  
  
upsert into user\_1 values ('1', 'zs', 'beijing');  
upsert into user\_1 values ('2', 'lisi', 'shanghai');  
upsert into user\_1 values ('3', 'ww', 'sz');

默认情况下, 只要 rowkey 支持索引(就是上面的 id)

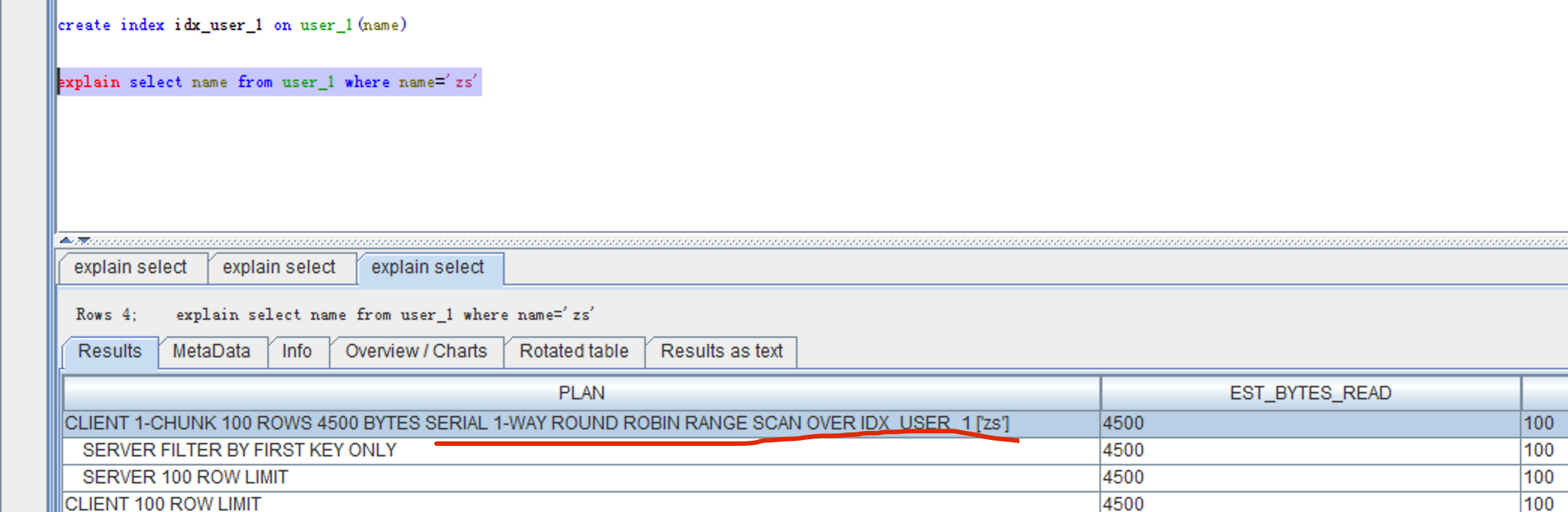


其他字段是不支持索引的:



给 name 字段添加索引:

create index idx\_user\_1 on user\_1(name)

 注意: 这种索引, 对 name 创建的索引, 则查询的时候也必须只查询 name 字段.

### 3.3 Phoenix 创建索引

#### 3.3.1 Phoenix 索引分类

Phoenix 索引分全局索引和局部索引

##### 全局索引

global index 是默认的索引格式。

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。

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。

创建全局索引的方法:

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

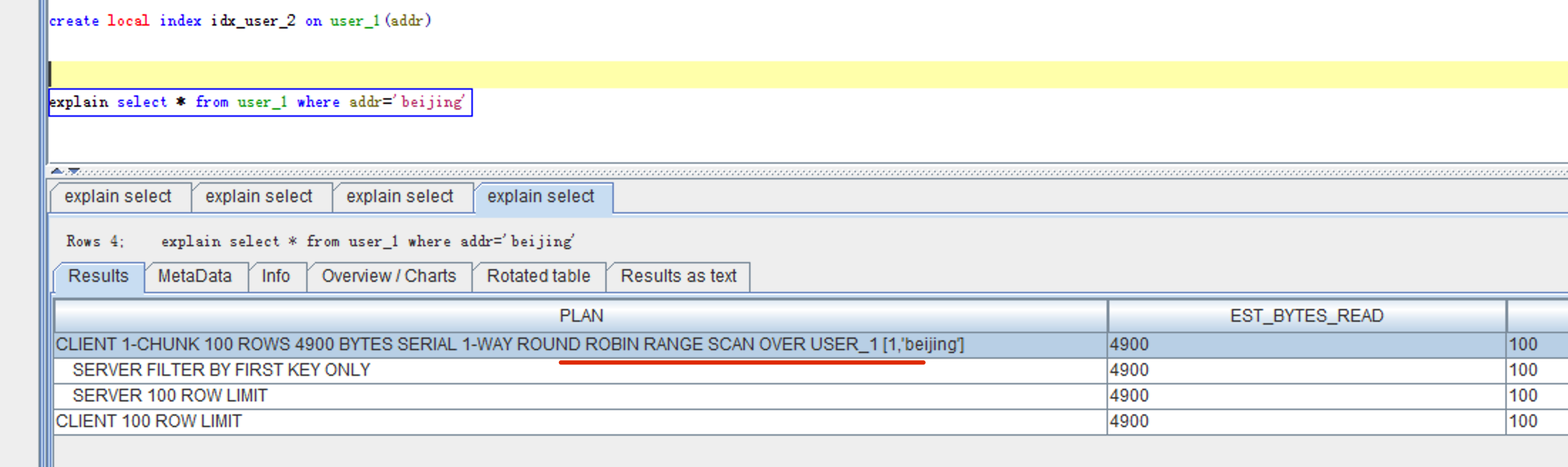
##### 局部索引

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相同的服务器中的，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。

创建局部索引的方法(相比全局索引多了一个关键字 local):

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





#### Local index 和 Global index区别：

Local index 由于是数据与索引在同一服务器上，所以要查询的数据在哪台服务器的哪个region是无法定位的，只能先找到region然后再利用索引。

Global index 是一种分布式索引，可以直接利用索引定位服务器和region，速度更快，但是由于分布式的原因，数据一旦出现新增变化，分布式的索引要进行跨服务的同步操作，带来大量的通信消耗。所以在写操作频繁的字段上不适合建立Global index。

#### 3.3.2 创建索引总结

1. 创建全局索引, 也支持查询其他字段

* CREATE INDEX my\_index ON my\_table (v1) INCLUDE (v2)  
    
  SELECT v2 FROM my\_table WHERE v1 = 'foo'

1. 创建局部索引

* CREATE LOCAL INDEX my\_index ON my\_table (v1)

#### 3.3.3 删除索引

DROP INDEX my\_index ON my\_table

