# 1.1 什么是Phoenix

## 1.1.1 Phoenix

Phoenix是构建在HBase上的一个SQL层，是HBase的开源SQL皮肤。它不仅可以使用标准的JDBC API替代HBase client API 创建表、插入、查询HBASE表，也支持二级索引、事务以及多种SQL层优化。

Phoenix完全使用java编写，作为HBase内嵌的JDBC驱动，Phoenix查询引擎支持使用SQL进行HBase数据的查询，会将SQL查询转换为一个或多个HBase扫描（Scan），并编排执行以生成标准的JDBC结果集，直接使用HBase API协同处理器自定义过滤器，对于简单查询来说，其性能量级是毫秒，对于百万级的行数来说，其性能量级是秒。

Phoenix通过一下方式减少我们的代码量，提高生产效率，并且性能比我们自己写代码更好：

1. 将SQL编译成原生的HBase scan;
2. 确定scan关键字的最佳开始和结束；
3. 让scan并行执行。

Phoenix官网地址：http://phoenix.apache.org/ 使用Phoenix的厂家如下：

图1

## 1.1.2 Phoenix架构特点及使用场景

图2

Phoenix和HBase的表是独立的，HBase中已经存储的表可以通过Phoenix来进行查询，也可以直接使用Phoenix来创建表，直接将表数据存储在HBase中，Phoenix作为应用层和HBase之间的中间件，有如下特点：

* 支持二级索引
* 支持标准化SQL，可以使用HBase具有更加灵活处理复杂事务能力
* 编译SQL为并行执行的查询任务
* Skip Scan功能提高扫描速度

以上这些特性使它在大数据量的简单查询场景有着独有的优势。Phoenix主要应用还在于OLTP类型的操作，例如

* 大数据量下有固定查询模式的场景
* 查询过滤条件结果小于几百万行毫秒级响应场景
* 非事务或者轻量级事务读写场景
* 对于一些公司MYSQL数据量大时性能变差 成本高，分库分表太复杂时也可以将MYSQL迁移到Phoenix中

# 1.2 Phoenix4.14安装部署

Phoenix采用单机安装方式即可，这里将Phoenix安装到node4节点上。Phoenix下载及安装步骤如下：

1. 下载Phoenix

Phoenix对应的HBase有版本之分，可以从官网：http://phoenix.apache.org/download.html来下载，要对应自己安装的HBase版本下载。我们这里的HBase版本为1.2.9，这里下载Phoenix4.14版本。下载地址如下：

http://archive.apache.org/dist/phoenix/apache-phoenix-4.14.1-HBase-1.2/bin/

图3

1. 上传解压

图4

1. 拷贝Phoenix整合HBase需要的jar包

将前面解压好安装包下的phoenix-4.14.1-HBase-1.2-server.jar与phoenix-core-4.14.1-HBase-1.2.jar两个java包发送到每个HBase节点下的lib目录下

图5

1. 复制core-site.xml、hdfs-site.xml、hbase-site.xml到Phoenix

将HDFS中的core-site.xml、hdfs-site.xml、hbase-site.xml复制到Phoenix bin目录下

图6

图7

1. 启动HDFS,HBase集群，启动Phoenix

图8

# 1.3 Phoenix中的数据类型

phoenix中的数据类型主要包含以下：

表1

表2

# 1.4 Phoenix命令操作

## 1.4.1查看帮助

使用“!help”查看所有Phoenix中的命令帮助

图9

图10

## 1.4.2 查看Phoenix表

使用“!table”或者：“!tables”来查看Phoenix中的表

图11

图12

## 1.4.3 创建表/创建带列族的表

在Phoenix中创建表，创建表时必须指定主键primary key，HBase默认把phoenix表的主键对应到rowkey，column family名为0，也可以在建表的时候指定column family.

HBase是区分大小写的，Phoenix默认会把sql语句中的小写转换成大写，再建表，如果不希望转换，需要将表名，字段名等使用双引号来标记【一定是双引号】。

这里建议在Phoenix中创建表时，使用varchar类型，方便在HBase查询对应数据时，可以不用转换成16进制显示。

图13

## 1.4.4 插入数据

向Phoenix中插入数据使用upsert命令。向表中插入数据时，如果有varchar类型的数据，一定要使用单引号引起来插入的字段【一定是单引号】

图13-1

## 1.4.5 修改数据

修改数据也是直接使用upset命令，这里继续向表“TEST”中插入数据

图13-1

## 1.4.6 删除数据

删除数据使用delete

# 1.5 Phoenix表映射

默认情况下，在HBase中创建的表在Phoenix中看不到，如果想要Phoenix中操作由HBase创建的数据表，则需要在Phoenix中进行表的映射，映射的方式有两种：视图映射和表映射

## 1.5.1 视图映射

在Phoenix的4.10版本之后，推荐使用基于视图的方式映射到hbase上的数据，视图映射仅提供高速检索与查询功能，删除视图也不会影响HBase上的数据。视图只是映射到HBase表中，不支持插入、修改、删除视图中的数据，向视图中插入及删除视图上的数据会出现Error：ERROR 505 (4200)：Table is read only这样的异常

在HBase中创建表，并且插入数据：

图17

注意：HBase shell 命令

图18

在Phoenix中创建视图映射

图19

图20

## 1.5.2 表映射

使用Phoenix创建对HBabse的表映射分为两种情况：

1. 当HBase中表不存在时

这种情况可以直接使用create table 指令创建需要的表，并且在命令中可以指定映射的HBase表的列族及各个列，默认Phoenix中创建的表不使用双引号引起，都会自动转换成大写，同时映射到HBase中表名也是大写。

1. 当HBase中表存在时

这种情况可以直接创建表映射即可，与创建视图映射一样，Phoenix中主键的名称可以随意定义，会通过表内置结果自动关联起来，自动识别rowkey为主键，Phoenix中的column必须以HBase的ColumnFamily开头，列名必须和HBase对应列族下的列名一致，否则数据不能映射到Phoenix中。

在Phoenix中创建映射表时，最后需要加上column\_encode\_bytes=0，否则HBase的数据不会映射过来。另外删除映射表，HBase中的表及数据也会删除。

图21

# 1.6 Phoenix二级索引

## 1.6.1 什么是二级索引

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全表扫描，硬件资源消耗和时间延时都会比较高。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，以满足现实中更复杂多样的业务需求。

在HBase中可以使用协处理器（Coprocessor）来构建二级索引，HBase在0.92之后引入了协处理器（Coprocessor）,添加了一些新的特性，能够轻易建立二次索引、实现访问控制。HBase中协处理器分为两类：Observer和Endpoint。

Observer类似于RDMS触发器，对数据进行前置或者后置拦截操作，通过使用RegionObserver接口可以实现二级索引的创建和维护。

EndPoint类似于RDMS的存储过程，主要作用于客户端，客户端可以调用这些EndPoint

执行一段Server端代码，并将Server端代码结果返回给客户端进行处理，如常见聚合操作，找一张大表某个字段的最大值，如果不用Coprocessor只能全表扫描，在客户端遍历所有结果找出最大值，且只能利用有限的客户端资源进行迭代计算，无法利用上HBase的并发计算能力，如果用了Coprocessor，则client端可在RegionServer端执行统计每个Region最大值的逻辑，并将Server端结果返回客户端，再找到所有Server端所返回的最大值中的最大值得到最终结果，很明显，这种方式尽量将统计执行下放到Server端，Client端只执行一些最后的聚合，大幅度提高了统计效率，过程大概是在各个RegionServer上做一些计算，然后将计算的结果汇集到Client端来做最后的处理

在Phoenix中也支持二级索引，Phoenix中的二级索引其实就是使用了大量的协处理器来实现的。我们创建了二级索引其实是在hbase额外建了一张表，该表数据无需我们维护，用Phoenix自动维护。在Phoenix中构建二级索引主要目的也是为了提高查询效率，主要用在加快where过滤数据执行索引速度快。

## 1.6.2 Phoenix配置索引

在每一个HRegionServer的hbase-site.xml加入以下配置

图22

以上配置完成之后，重启HBase集群

图23

图24

## 1.6.3 Phoenix索引分类

在Phoenix中常见额索引分为两类：全局索引（Global index）和本地索引（Local index）。

### 1.6.3.1 全局索引（Global index）-多读场景

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，由于索引表是分布在不同的数据节点上，写数据索引表也要更新，跨节点额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（查询和过滤字段必须是索引字段），也就是说不会带来查询速度提升。

创建/删除全局索引语法如下：

图25

注意：由于在Phoenix中创建二级索引其实是在hbase额外建了一张表，所以创建索引名称不能相同

测试：

图26

图27

### 1.6.3.2 本地索引（Local index）-多写场景

Local index适用于写操作频繁的场景（多写少读），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（过滤字段需要是索引字段），这会带来查询速度的提示。

创建/删除本地索引语法如下：

图28

注意：由于在Phoenix中创建二级索引其实是在HBase额外建一张表，所以创建的索引名称不能相同。

测试：

图29

图30

### 1.6.3.3 全局索引&本地索引区别

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，带来大量的通讯消耗。所以在写操作频繁的字段上适合建立Global index，适合多读少写场景。

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，适合少读多写场景。

# 1.7 Phoenix 加盐表

在Phoenix中，如果主键顺序的序列，在频繁插入Phoenix到表中，最终数据不停的落在一个regionServer中，容易造成热点问题，所以尽量将id打散。解决这个问题可以在Phoenix中建表时，加一个盐值，例如：SALT\_BUCKETS = 3,利用表中定义盐值，数据会均匀的分布在各个region中。

在密码学中，加盐是指在散列之前将散列内容（例如：密码）的任意固定位置插入特定的字符串。这个散列中加入字符串的方式称为“加盐”。其作用是让加盐后的散列结果和没有加盐的结果不相同。

Phoenix中加盐散列数据的原理是利用原key经过某个hash算法，并对SALT\_BUCKET求余数的方式后转为byte，该字节数据加到原ROWKEY前缀，这样将数据较均匀的分布到集群中的所有RegionServer。

如果设定盐值？建议是HBase节点数的倍数，最少是1倍，数据量大时可增加盐数，比如对于8个region server集群的大表，可以考虑设计64~128个slat buckets。

加盐表创建时在建表语句后加上SALT\_BUCKETS = xxx即可，使用如下：

图31

图32

使用加盐表注意事项

* 在理想情况下，对于具有四核CPU的16个RS集群，选择32-64之间的SALT\_BUCKETS可获得最佳性能。
* 最多加盐256个，因为盐是1个字节（8位，0-255）。
* 加盐后，会多一步计算前缀的过程，有一定性能损耗。
* 加盐表一旦创建不可修改。
* 加盐表初始时Bucket和Region数量相同，随着数据增加Region也会跟其他Region一样拆分。
* SALT\_BUCKETS设置过大可能降低Range Scan性能。再比如用in查Phoenix主键前导字段，那么字段枚举\*盐数就有很高的并发了，致使一个较为精确的查询需要大量scan，反而增加查询时间。

# 1.8 使用JDBC连接Phoenix读取HBase中的数据

可以使用JDBC方式连接Phoenix操作HBase中的数据，需要在项目pom.xml中家人依赖

图33

图34

编写使用JDBC方式连接Phoenix代码：

图35

图36

图37

# 1.9 Phoenix使用注意事项

* 通过Phoenix创建的表必须指定Primary key，除了Primary key不需要指定列族，其他列最好指定列族名称，并且建议使用varchar类型。
* 通过Phoenix创建的表要删除，不要HBase shell或者使用HBase客户端API删除，必须在Phoenix中删除，否则会出现表删除但是Phoenix中还能看到删除表，但是查询表会抛异常。
* select语句禁止全表扫描，一定要走主键索引或者二级索引，可用explain查看sql执行计划。我们可以巧妙合理的设计我们的主键。
* 使用主键索引查询时，where条件的顺序一定要和建表定义主键的顺序一致，要不然走不到主键索引。
* 使用二级索引只能查主键字段和创建索引是INCLUDE后面添加的字段，禁止查额外字段，因为查了其他字段就不会走到二级索引，是全表扫描。非要查额外字段可以查两次，先查主键，再根据主键去查。
* 建表和索引是最好都提前加盐。

# 1.10 Phoenix图型化工具SQuirrel

SQuirrel SQL Client s是一个用Java写的数据客户端，用JDBC统一数据库访问接口以后，可以通过一个统一的用户界面来操作MYSQL PostgreSQL MSSQL Oracle等等任何支持JDBC访问的数据库，使用起来非常方便。Squirrel的安装使用步骤如下：

1. 下载安装：

SQuirrel 的下载地址：http://www.squirrelsql.org/

下载完成后，window中打开CMD,执行如下命令，打开安装界面：

java -jar ./squirrel-sql-4.2.0-standard.jar

打开安装界面一路“next”直到安装成功。

1. 配置SQuirrel连接Phoenix

将Linux Phoenix安装目录下的”phoenix-4.14.1-HBase-1.2-client.jar”复制到SQiurrel安装目录”L:\Progrem Files\SQuirrel”下的lib下

打开SQuirrel，添加Driver配置信息

图38

图39

Example URL:jdbc.phoenix:node3,node4,node5:2181

Class Name:org.apache.phoenix.jdbc.PhoenixDriver

1. 配置Aliases，设置连接Phoenix名称

图40