**

**HUNAN UNIVERSITY**

《数据仓库技术与应用》

实验报告五

|  |  |
| --- | --- |
| **报告名称：** | 分布式数据库Hbase |
| **学生姓名：** | 杨超然 |
| **学生学号：** | 202106060220 |
| **专业班级：** | 电商2102班 |
| **学 院：** | 工商管理学院 |
| **指导老师：** | 李信儒 |
| **日 期：** | 2023.3.20 |

目录

[一、实验过程 3](#_Toc2032955191)

[（一）安装Hbase 3](#_Toc1413071367)

[1.Hbase1.1.2安装 3](#_Toc468812996)

[（二） Hbase配置 5](#_Toc222522929)

[1.单机模式配置 5](#_Toc1173838276)

[2.伪分布式模式配置 7](#_Toc1915123390)

[（三） 编程实践 10](#_Toc993914494)

[1.利用Shell命令 10](#_Toc1593094292)

[二、易错点总结 18](#_Toc309654848)

[（一）启动HBase时报错 18](#_Toc1016153655)

[三、学习小记 19](#_Toc1704518641)

[（1）认识Hbase 19](#_Toc412948307)

[（2） HBase存在的必要性 19](#_Toc1902532292)

[（3） HBase与传统数据库对比 20](#_Toc1976211461)

[（4） 数据模型概念 21](#_Toc1203940525)

[（5） HBase系统架构 22](#_Toc1037481641)

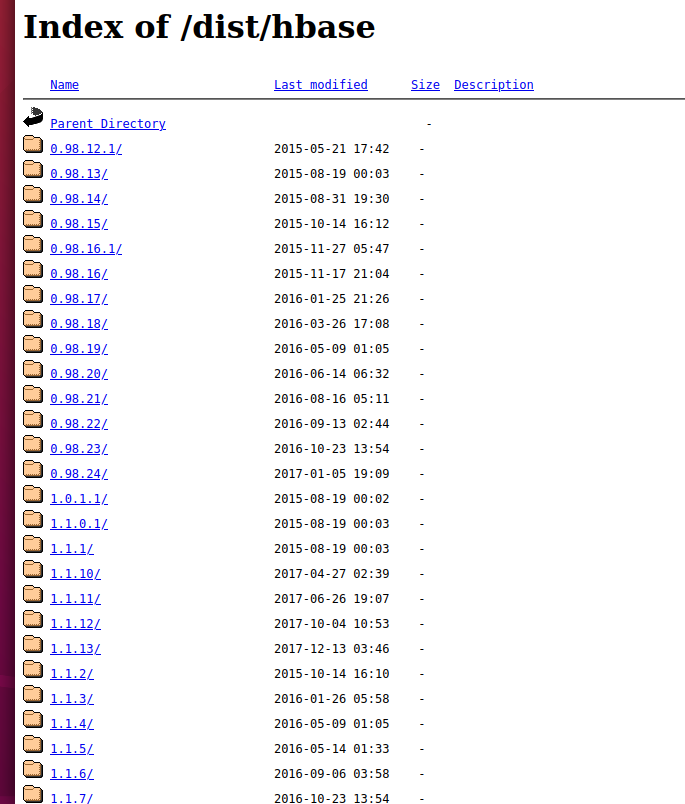
[（6） HBase性能优化方法 23](#_Toc1534210294)

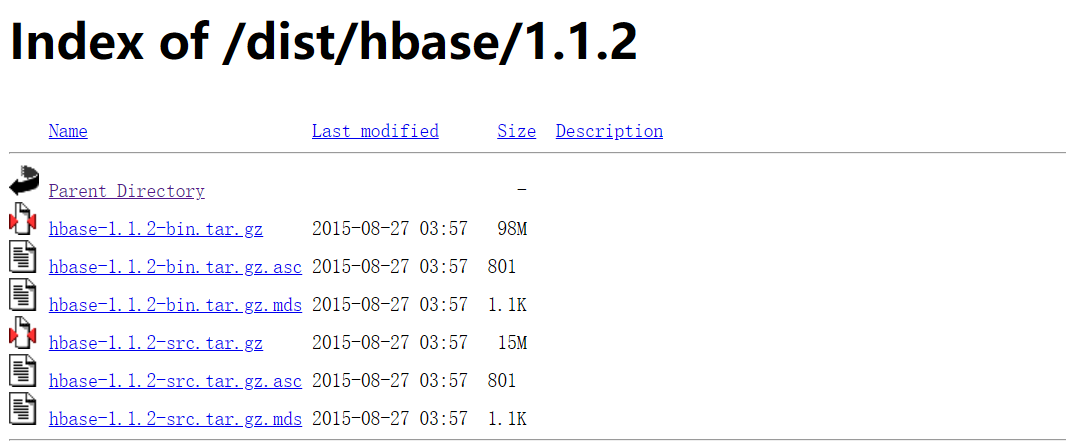
# 一、实验过程

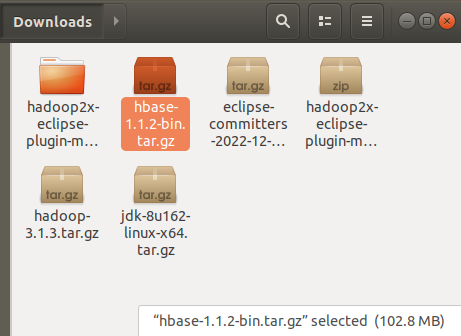
## （一）安装Hbase

1.Hbase1.1.2安装

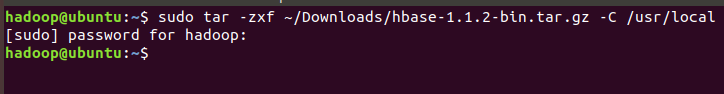
注：由于笔者采用虚拟机VMware安装Ubuntu，因此需要在虚拟机中找到相应的下载地址进行安装，不可在Windows下的浏览器下载，因为文件会被下载到Ubuntu系统中，而虚拟机中的Ubuntu无法访问外部Windows系统的文件，这会造成不必要的麻烦



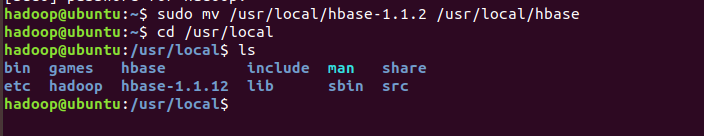




1.1解压安装包hbase-1.1.2-bin.tar.gz至路径/usr/local



1.2将解压的文件名hbase-1.1.2改为hbase，以方便后续操作

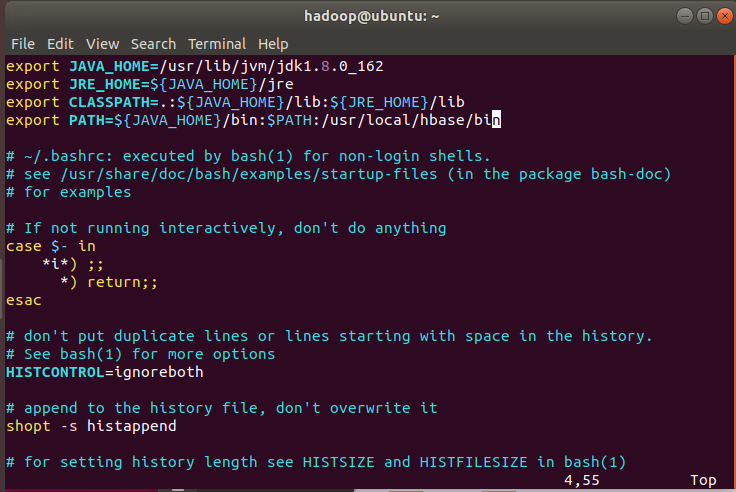


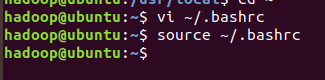
1.3配置环境变量



若没有引入PATH,则在~/.bashrc添加：export PATH=$PATH:/usr/local/hbase/bin

若已经引入PATH,则在export PATH这一行追加usr/local/hbase/bin(：是分隔符)



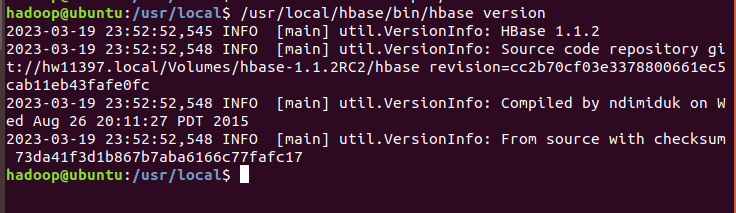


1.4添加Hbase权限



1.5查看Hbase版本以保证Hbase安装成功

若出现如下输出信息，则代表Hbase已经安装成功

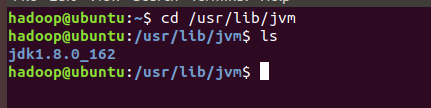


## Hbase配置

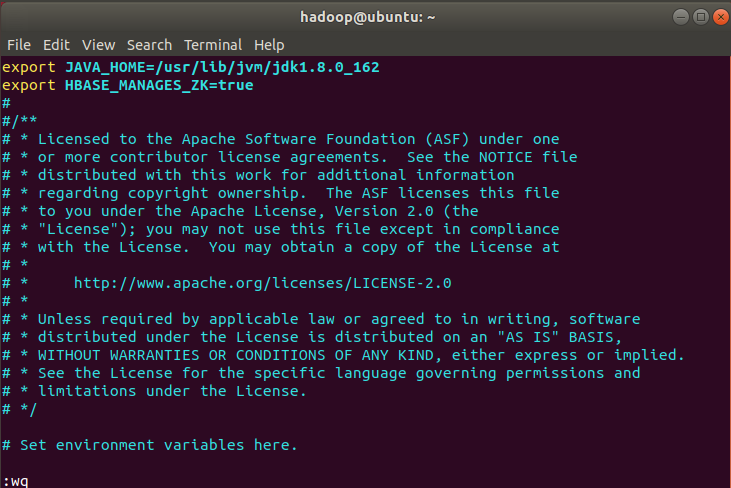
1.单机模式配置

1.1配置/usr/local/hbase/conf/hbase-env.sh,配置JAVA环境变量并添加配置

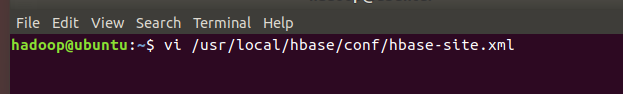
首先查看自己的jdk版本，可能和指南的jdk版本不一样，不能直接照抄

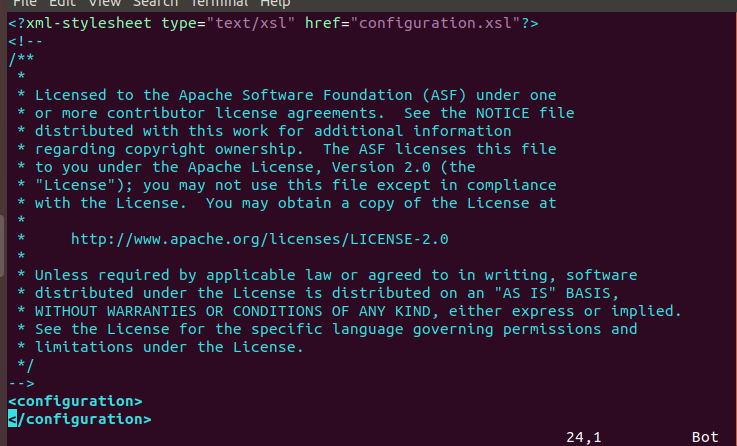




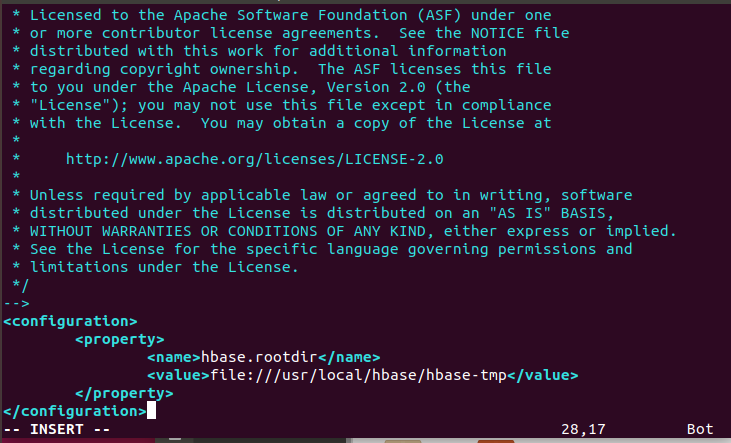


1.2配置/usr/local/hbase/conf/hbase-site.xml



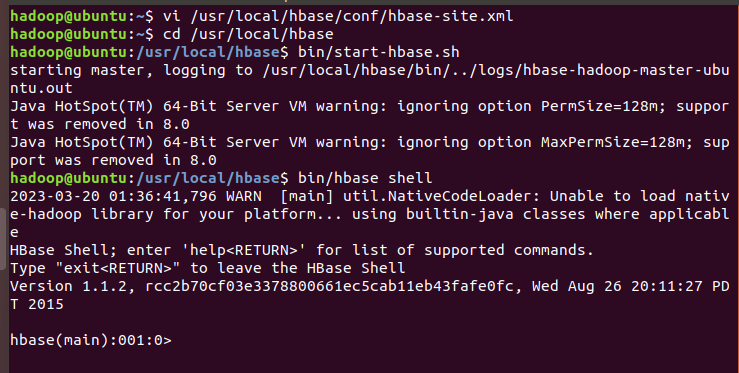


添加配置如下：

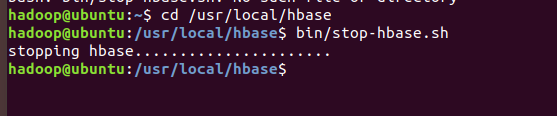


1.3测试与停止运行

测试运行：



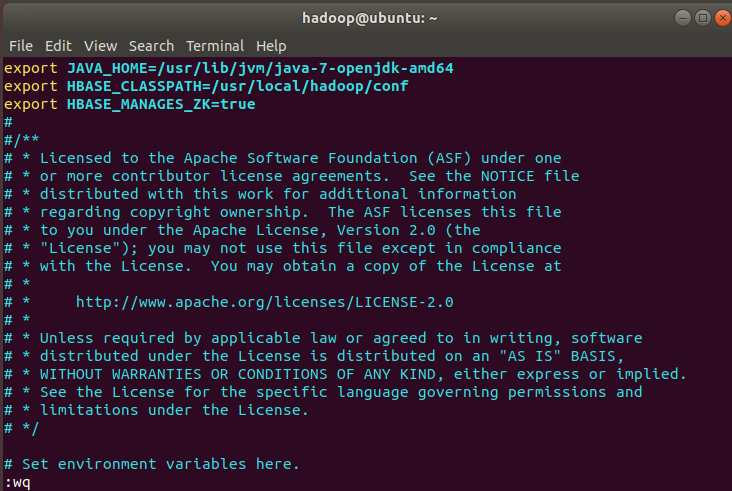
停止运行：



2.伪分布式模式配置

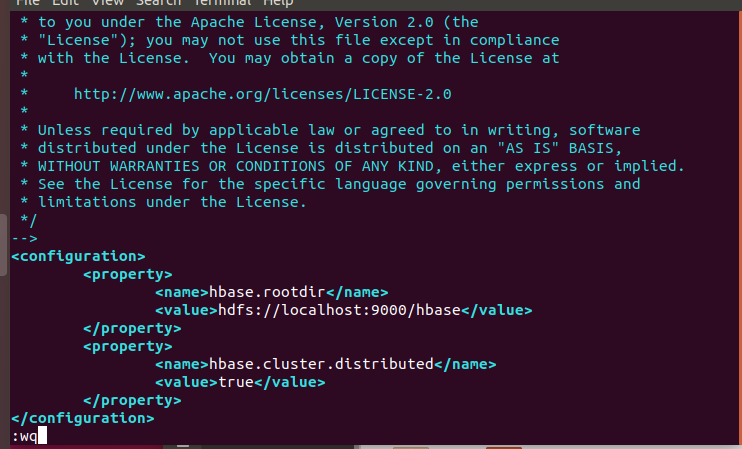
2.1配置/usr/local/hbase/conf/hbase-env.sh





2.2配置/usr/local/hbase/conf/hbase-site.xml

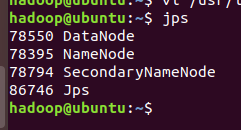




2.3测试与停止运行Hbase

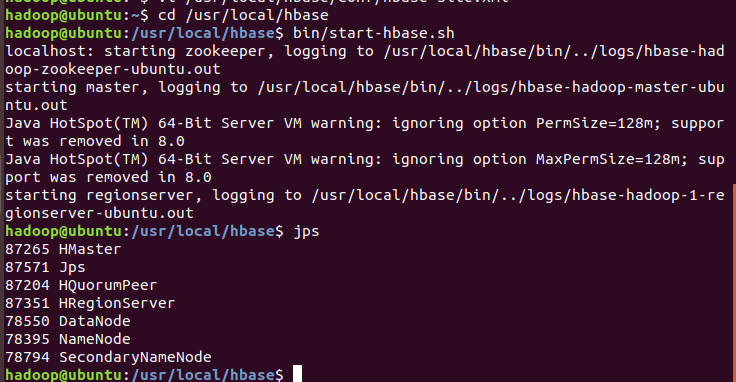
测试运行：

·首先登录ssh，切换至目录/usr/local/hadoop,在启动hadoop，输入命令jps能看到NameNode,DataNode和SecondaryNameNode都已经成功启动，表示hadoop启动成功(笔者之前已经启动hadoop，故不重复操作)

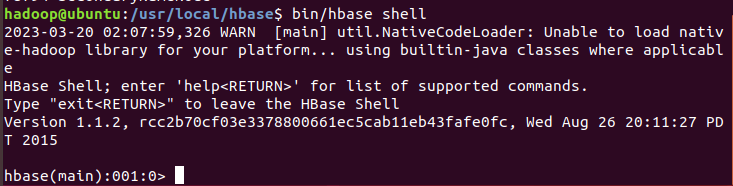


·切换目录至/usr/local/hbase再启动HBase

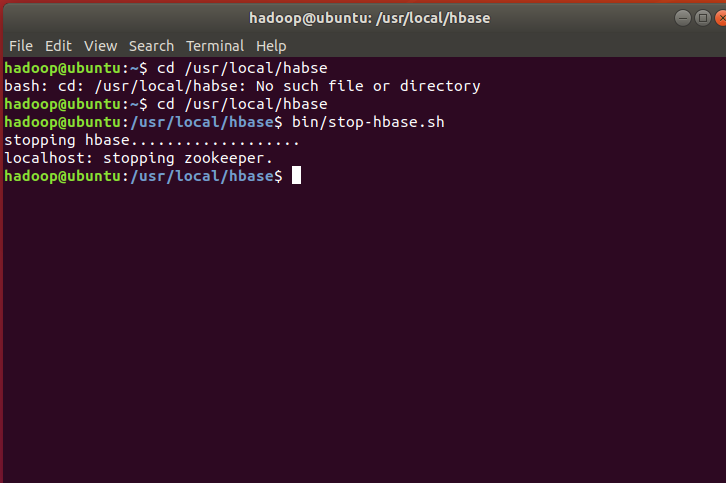
启动成功，输入jps命令，出现以下界面则表示启动成功



进入shell界面

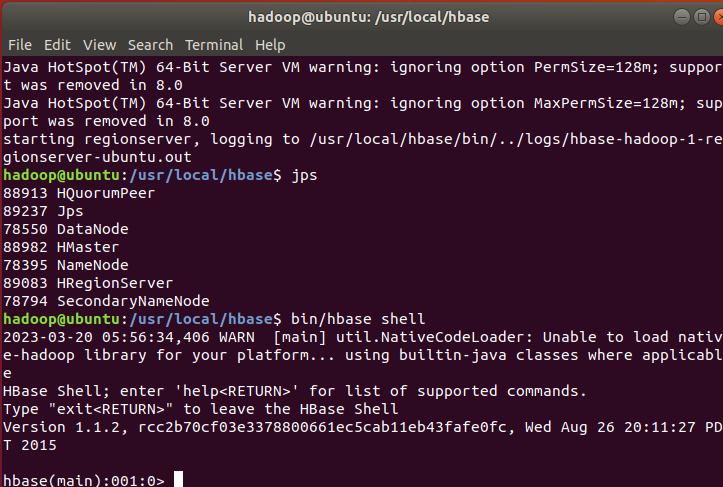


停止运行：



## 编程实践

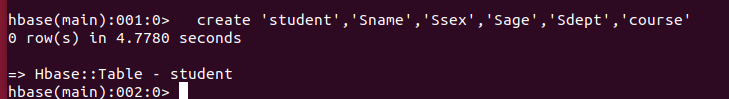
首先启动刚刚被我们关掉的HBase并进入shell界面



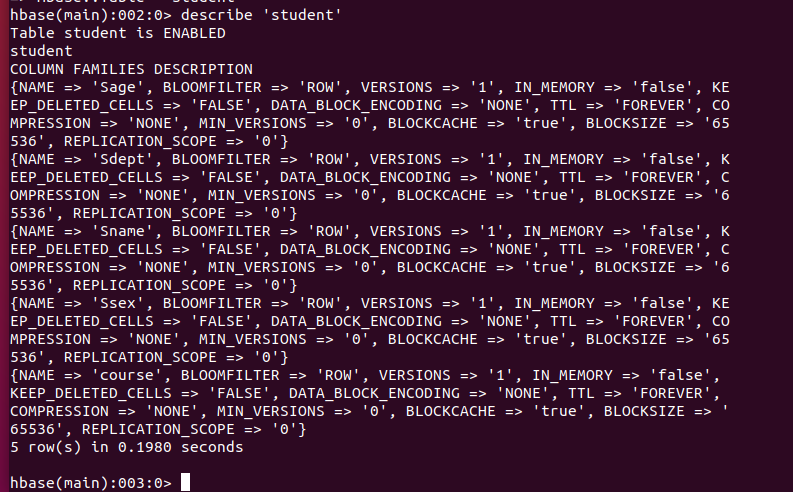
1.利用Shell命令

1.1Hbase中创建表

·使用create命令创建表



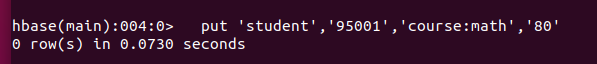
·使用describe命令查看student表的基本信息



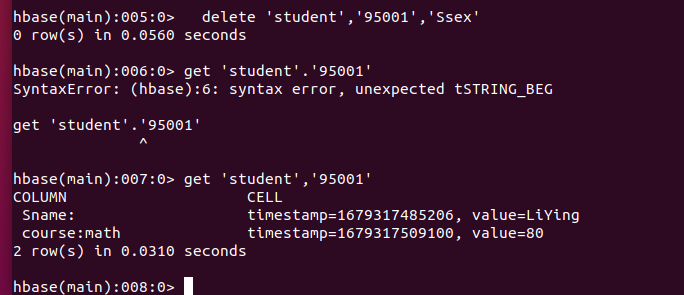
1.2Hbase数据库基本操作

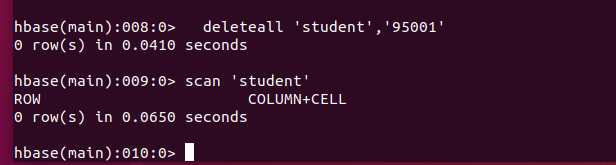
·添加数据：使用put命令添加数据（一次只能为一个表的一行数据的一个列，也就是一个单元格添加一个数据，所以直接用shell命令插入数据效率很低，在实际应用中，一般都是利用编程操作数据。）



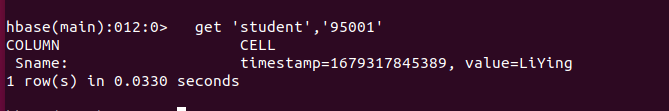


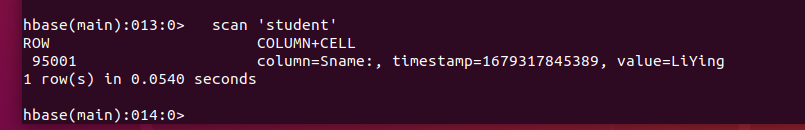
·删除数据：使用delete命令删除一个数据（视作put的反向操作），使用deleteall命令删除一行数据



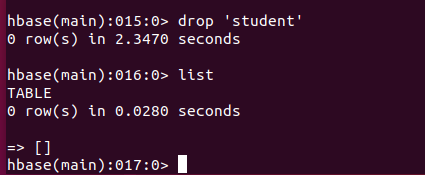


·查看数据：使用get命令查看表的某一行数据；使用scan命令查看表的全部数据



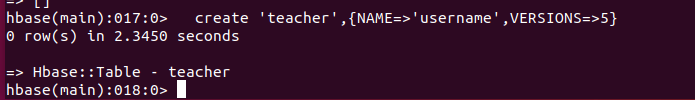


·删除表：分为两步——首先使用disable命令使表不可用，再用drop命令删除表

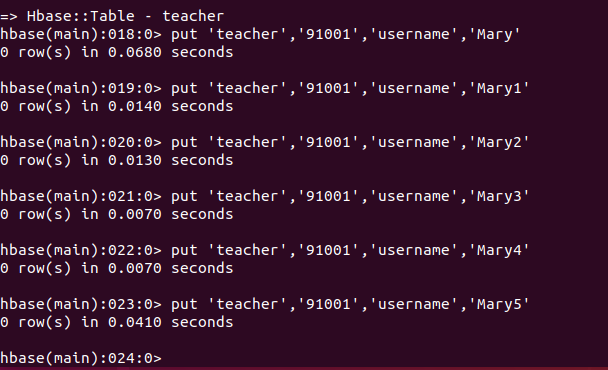


1.3查询表历史数据

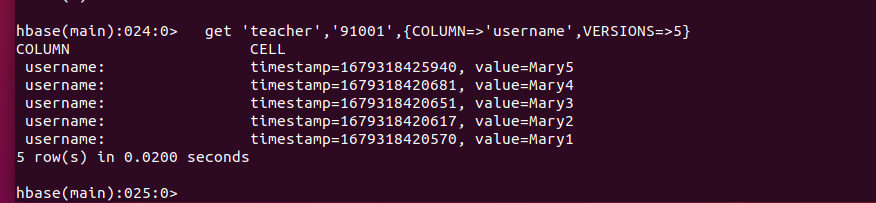
·首先在创建表的过程中指定保存的版本数（假设为5）



·然后插入新数据，使其产生历史数据（此处的插入和更新都只能使用put命令）



·查询时，指定历史版本数，默认会查询出最新数据



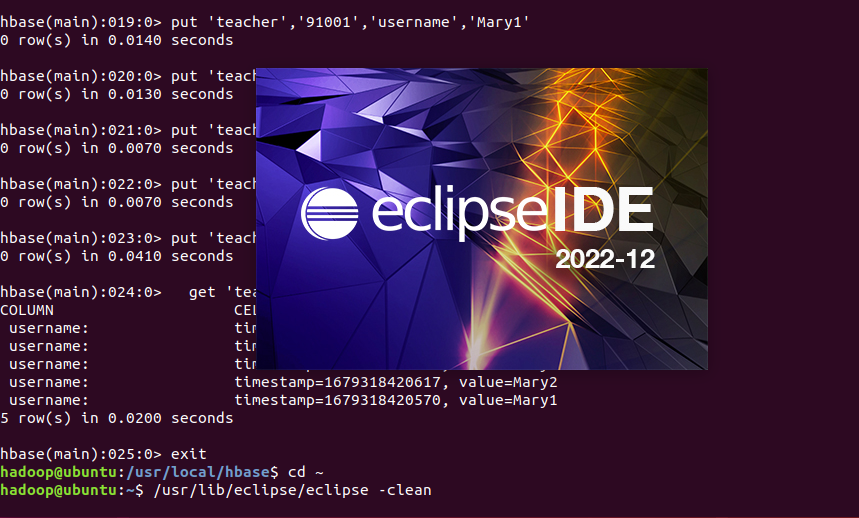
1.4退出Hbase数据库表操作

·使用exit命令退出数据库，需要注意的是，这里退出HBase数据库是对数据库表的操作，而不是停止启动HBase数据库的后台运行

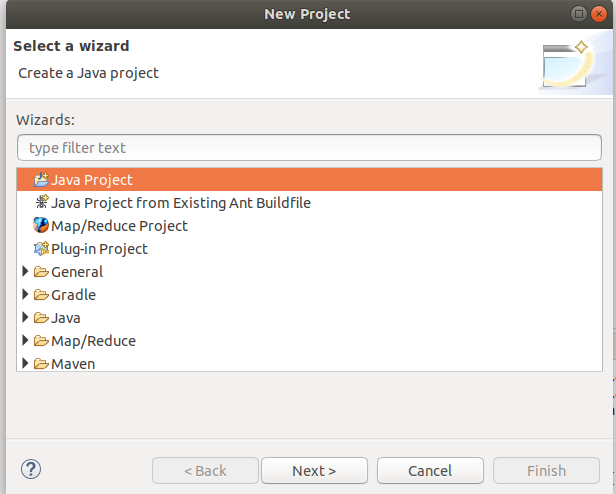


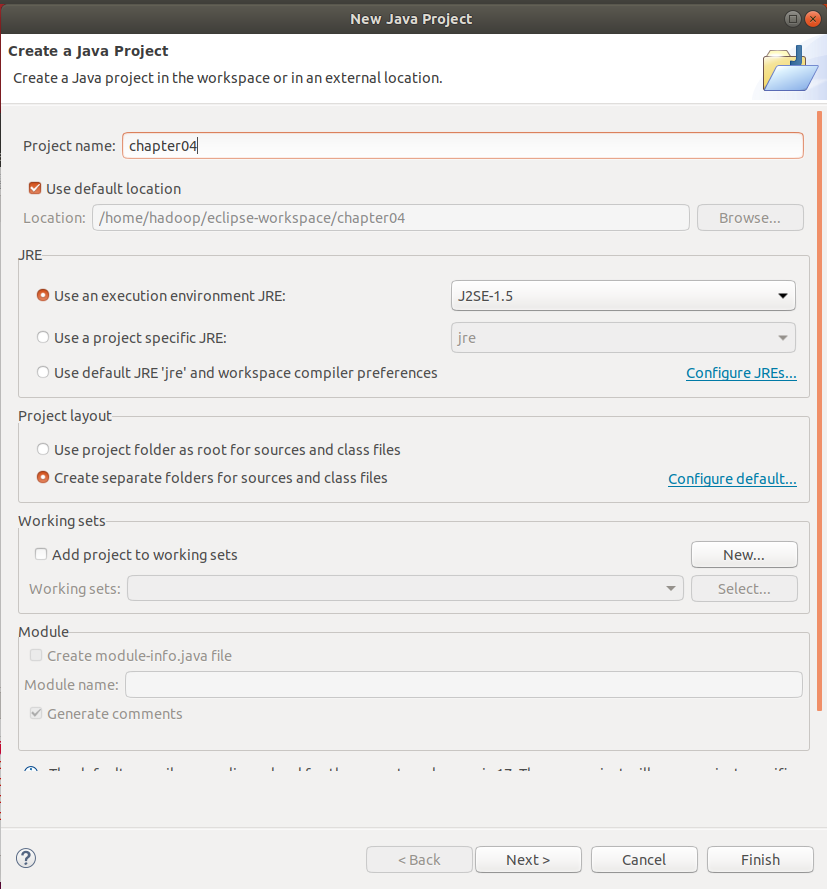
2.JAVA API编程实例

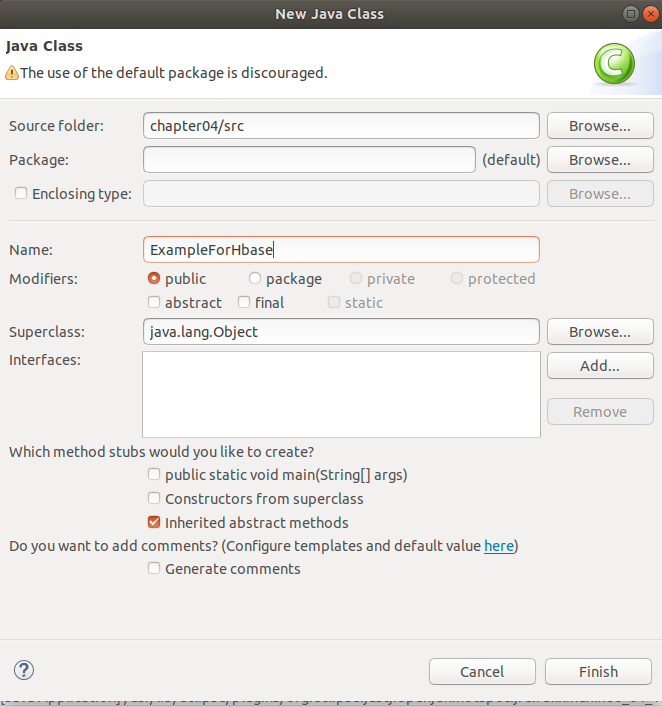
首先需要启动Hadoop和HBase，因笔者之前已经启动，故不做重复操作，之后进入eclipse



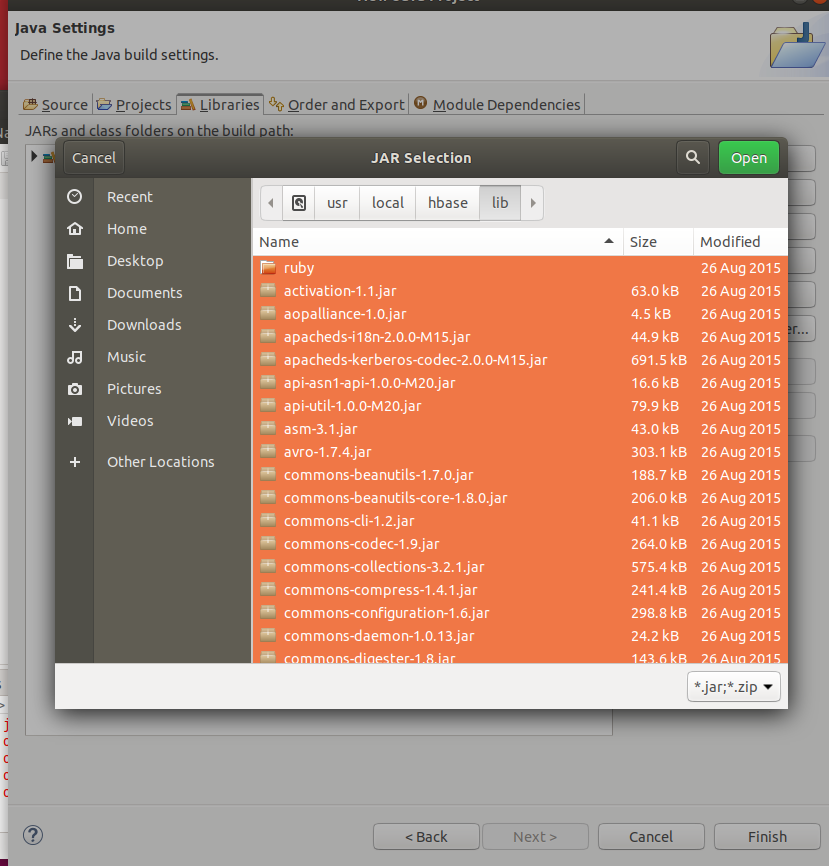
新建class



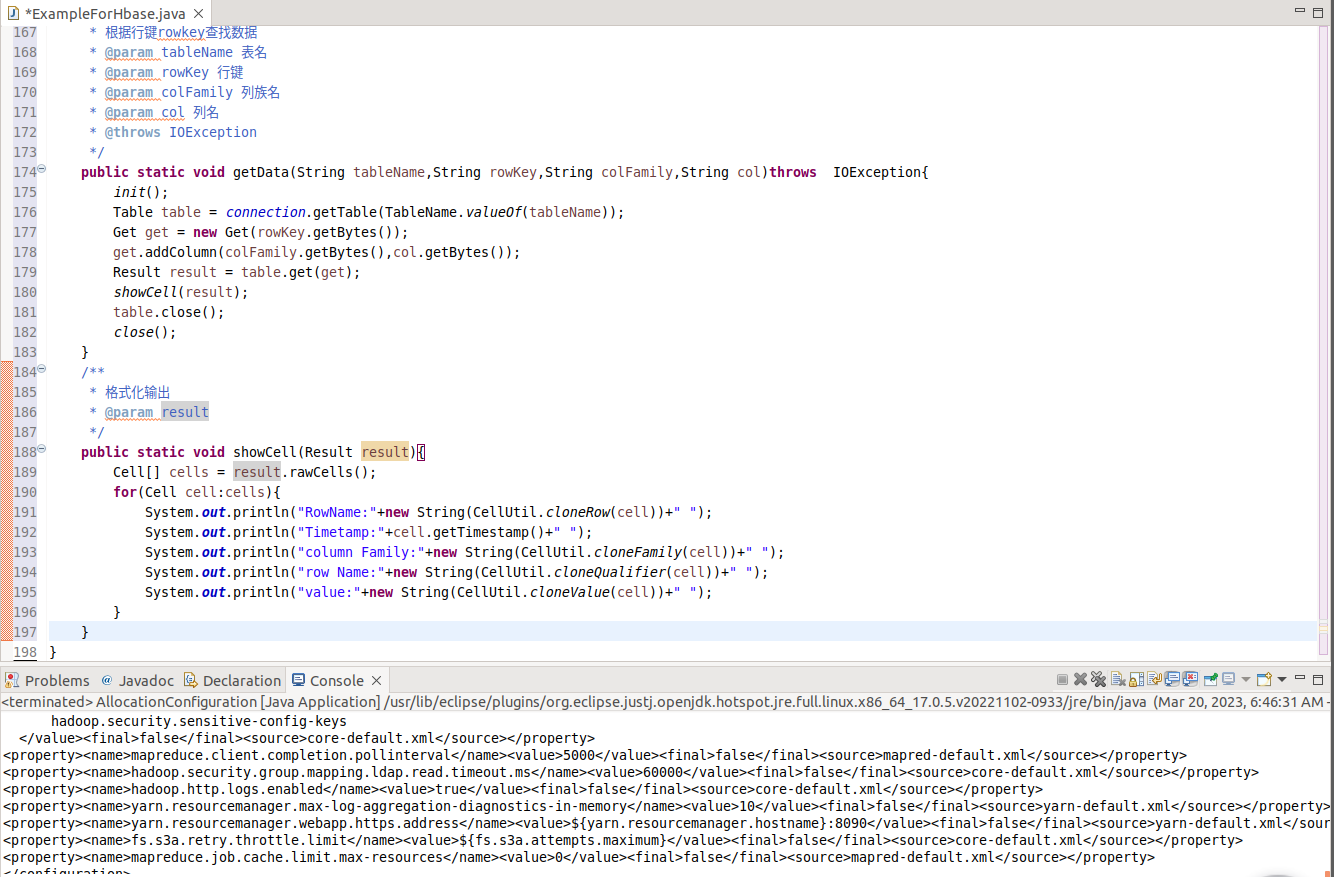




在工程中导入外部jar包



运行编程实例

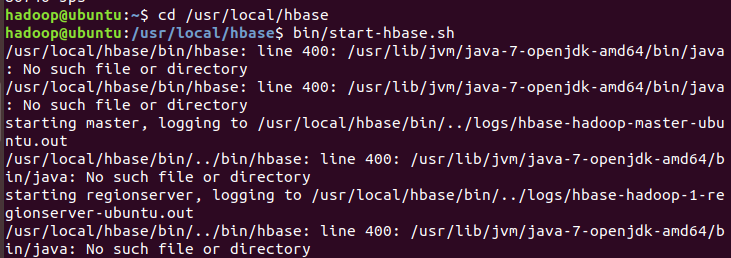


# 二、易错点总结

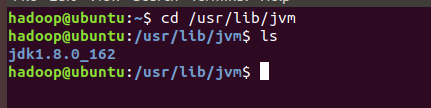
## （一）启动HBase时报错

在启动伪分布式HBase时，出现错误：

/usr/lib/jvm/java-7-openjdk-amd64/bin/java:No such file or directory,导致HBase无法正常启动。这其实就是笔者在（二）HBase配置模块中一开始就声明的问题：即我安装的JDK版本不是java-7-openjkd-amd64,但是在引用路径的步骤中直接复制了指南的代码，因此自然无法启动



**解决方案：**查看自己的JDK版本，在Export PATH步骤输入与本机相对应的版本



# 三、学习小记

## （1）认识Hbase

Hbase是一个分布式的、面向列的开源数据库，具有高可靠、高性能、面向列、可伸缩的特征，其源于Google的一篇论文《BigTable:一个结构化数据的分布式储存系统》。Hbase以表的形式存储非结构化的松散数据数据，目标是处理非常庞大的表，表由行和列组成，划分为若干个列族/列簇(column family)。HBase可以通过水平扩展的方式、利用廉价计算机集群处理由超过10亿行数据和数百万列元素组成的数据表

Hbase有三种运行模式：单机模式、伪分布式模式、分布式模式。单机模式指的是在一台计算机上安装和使用Hbase，不涉及数据的分布式存储；伪分布式模式指的是在一台计算机上模拟一个小的集群。分布式模式指的是使用多台计算机实现物理意义上的分布式存储。

## HBase存在的必要性

·Hadoop可以很好地解决大规模数据的离线批量处理问题，但是，受限于Hadoop MapReduce编程框架的高延迟数据处理机制，使得Hadoop无法满足大规模数据实时处理应用的需求

·HDFS面向批量访问模式，不是随机访问模式

·传统的通用关系型数据库无法应对在数据规模剧增时导致的系统扩展性和性能问题（分库分表也不能很好解决）

·传统关系数据库在数据结构变化时一般需要停机维护；空列浪费存储空间

因此，业界出现了一类面向半结构化数据存储和处理的高可扩展、低写入/查询延迟的系统，例如，键值数据库、文档数据库和列族数据库（如BigTable和HBase等）

·HBase已经成功应用于互联网服务领域和传统行业的众多在线式数据分析处理系统中

## HBase与传统数据库对比

·数据类型：关系数据库采用关系模型，具有丰富的数据类型和存储方式，HBase则采用了更加简单的数据模型，它把数据存储为未经解释的字符串

·数据操作：关系数据库中包含了丰富的操作，其中会涉及复杂的多表连接。HBase操作则不存在复杂的表与表之间的关系，只有简单的插入、查询、删除、清空等，因为HBase在设计上就避免了复杂的表和表之间的关系

·存储模式：关系数据库是基于行模式存储的。HBase是基于列存储的，每个列族都由几个文件保存，不同列族的文件是分离的

·数据索引：关系数据库通常可以针对不同的列构建复杂的多个索引、以提高数据库的访问性能。HBase只有一个索引——行键，通过巧妙的设计，HBase中的所有访问方法，或者通过行键访问，或者通过行键扫描，是的整个系统不会慢下来

·数据维护：在关系数据库中，更新操作会用最新的当前值去替换记录中原有的旧值，旧值被覆盖后就不会存在。而在HBase中执行更新操作时，并不会删除旧版本的数据，而是生成一个新的版本，原有版本仍然保留。

·可伸缩性：关系数据库很难实现横向扩展，纵向扩展的空间也十分有限。相反，Hbase和BigTable这些分布式数据库是为了实现灵活的水平扩展而开发的，能够轻易地通过在集群中增加或减少硬件数量来实现性能的伸缩。

## 数据模型概念

·表：HBase采用表来组织数据，表由行和列组成，列划分为若干个列族

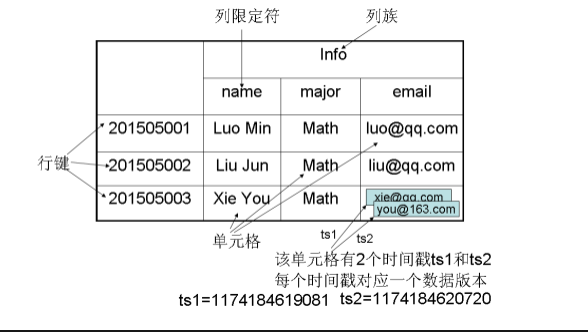
·行：每个HBase表都由若干行组成，每个行由行键（row key）来标识。

·列族：一个HBase表被分组成许多“列族”（Column Family）的集合，它是基本的访问控制单元

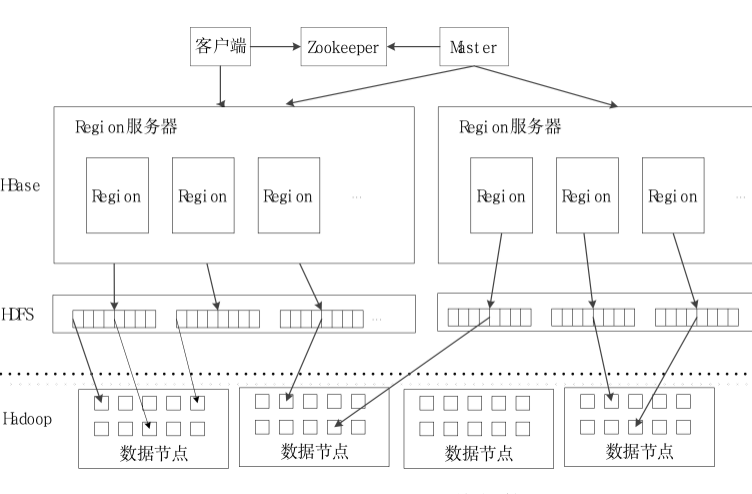
·列限定符：列族里的数据通过列限定符（或列）来定位

·单元格：在HBase表中，通过行、列族和列限定符确定一个“单元格”（cell），单元格中存储的数据没有数据类型，总被视为字节数组byte[]

·时间戳：每个单元格都保存着同一份数据的多个版本，这些版本采用时间戳进行索引

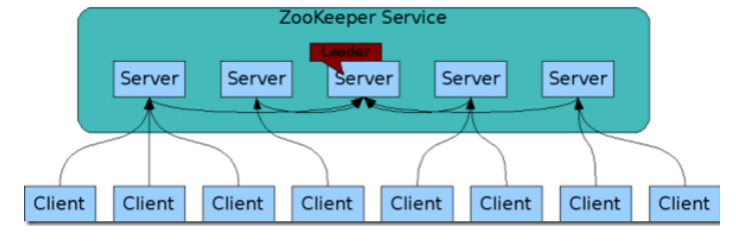


## HBase系统架构



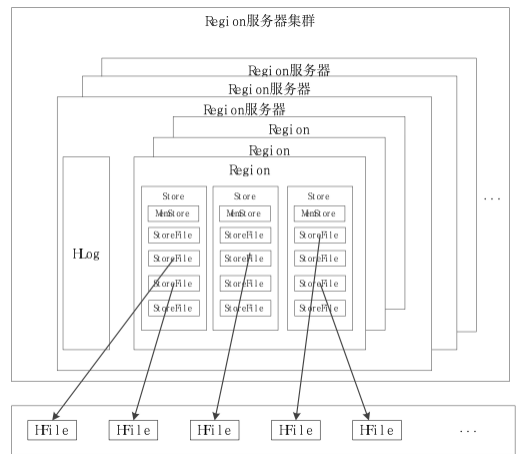
·客户端：客户端包含访问HBase的接口，同时在缓存中维护着已经访问过的Region位置信息，用来加快后续数据访问过程

·Zookeeper服务器：Zookeeper可以帮助选举出一个Master作为集群的总管，并保证在任何时刻总有唯一一个Master在运行，这就避免了Master的“单点失效”问题Zookeeper是一个很好的集群管理工具，被大量用于分布式计算，提供配置维护、域名服务、分布式同步、组服务等。



·Master：主服务器Master主要负责表和Region的管理工作：1.管理用户对表的增加、删除、修改、查询等操作；2.实现不同Region服务器之间的负载均衡；3.在Region分裂或合并后，负责重新调整Region的分布；4.对发生故障失效的Region服务器上的Region进行迁移。

·Region服务器：Region服务器是HBase中最核心的模块，负责维护分配给自己的Region，并响应用户的读写请求



## HBase性能优化方法

·行键（Row Key）：行键是按照字典序存储，因此，设计行键时，要充分利用这个排序特点，将经常一起读取的数据存储到一块，将最近可能会被访问的数据放在一块。举个例子：如果最近写入HBase表中的数据是最可能被访问的，可以考虑将时间戳作为行键的一部分，由于是字典序排序，所以可以使用Long.MAX\_VALUE - timestamp作为行键，这样能保证新写入的数据在读取时可以被快速命中。

·InMemory:创建表的时候，可以通过HColumnDescriptor.setInMemory(true)将表放到Region服务器的缓存中，保证在读取的时候被cache命中

·MaxVersion:创建表的时候，可以通过HColumnDescriptor.setMaxVersion(int maxVersions)设置表中数据的最大版本，如果只需要保存最新的版本数据，可以设置setMaxVersions(1)。

·Time To Live:创建表的时候，可以通过HColumnDescriptor.setTimeToLive(int timeToLive)设置表中数据的存储生命期，过期数据将自动被删除，例如如果只需要存储最近两天的数据，那么可以设置setTimeToLive(2 \* 24 \* 60 \* 60)。