HBase

1、HBase简介

高可靠、高性能、面向列、可伸缩的分布式存储系统，分布式数据库。可以用来存储非结构化和半结构化的松散数据

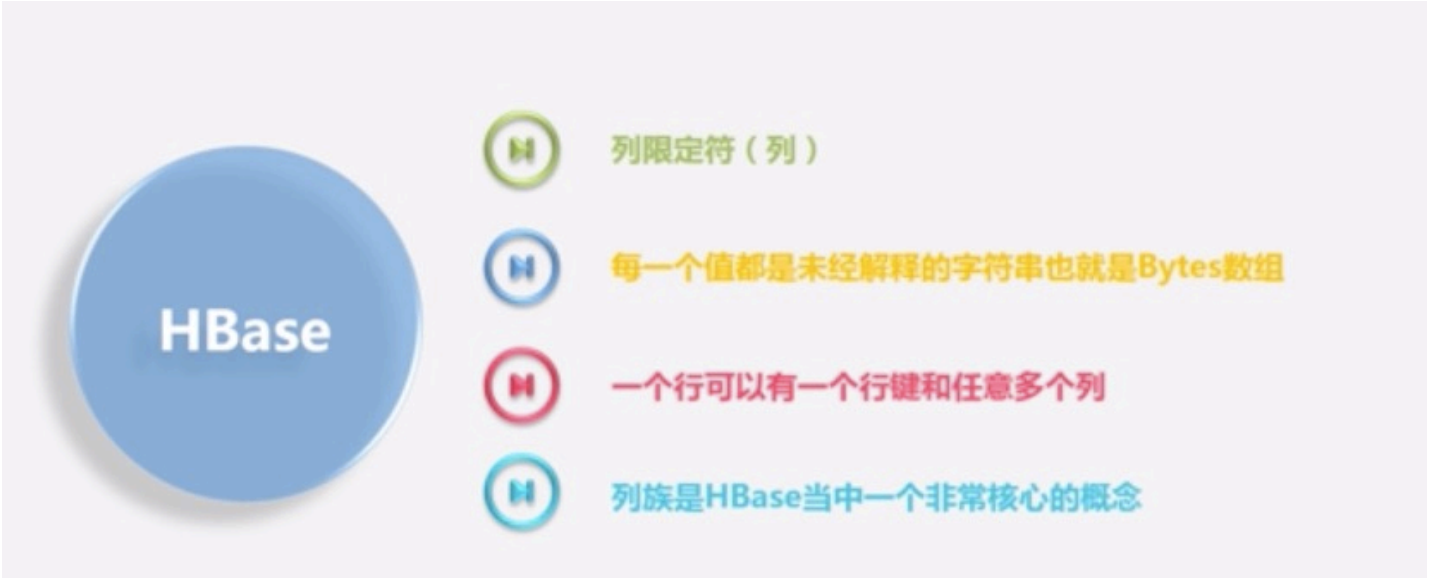
为什么要去设计HBase数据库?

内部切分、水平扩展都是后台全自动实现，具有非常好的水平可扩展性

数据结构多变，HBase能满足不断增长的数据存储需求

2、HBase数据模型

HBase是一个稀疏的多维度的排序映射表

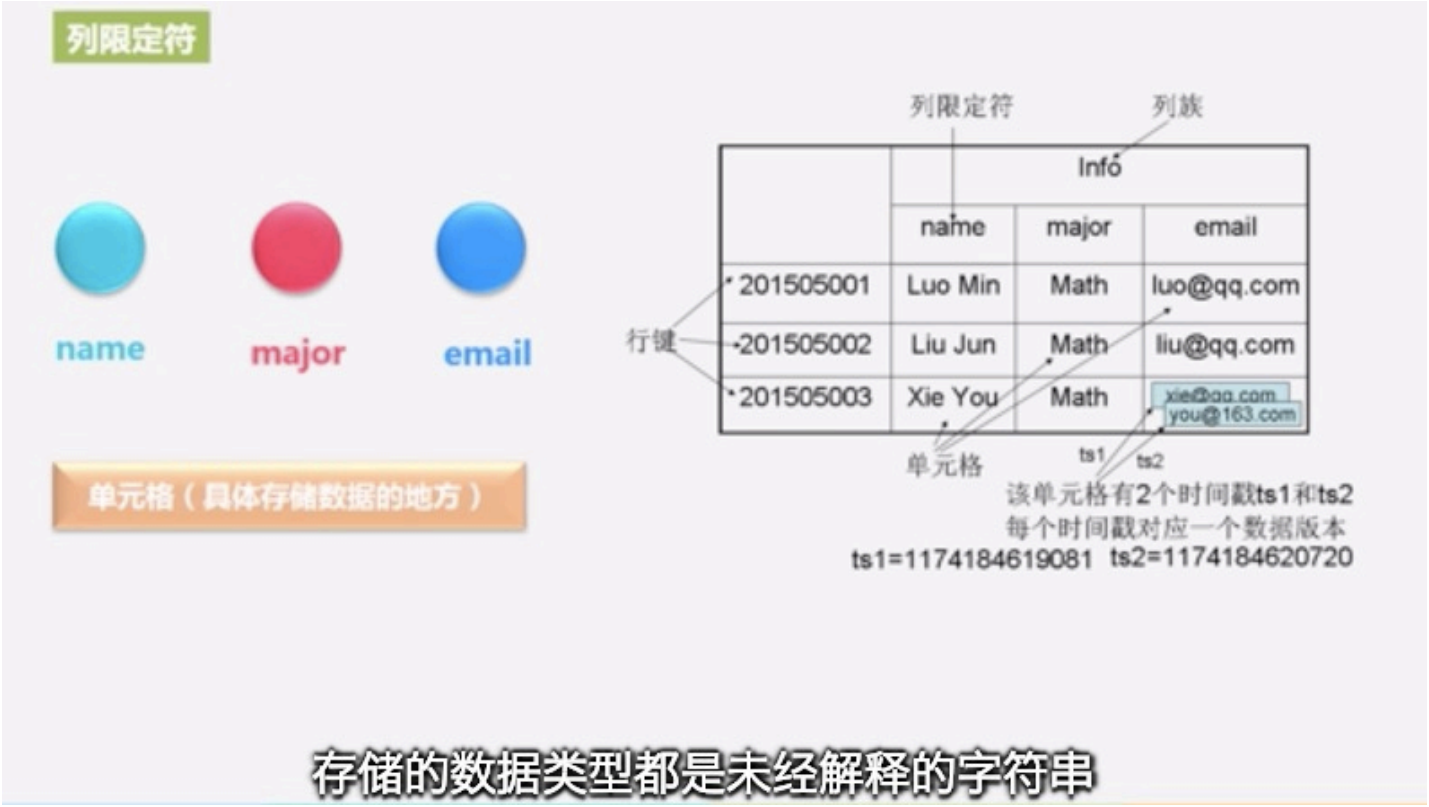


列族

支持动态扩展

保留旧的版本（由于HDFS只支持写入操作，不允许修改）

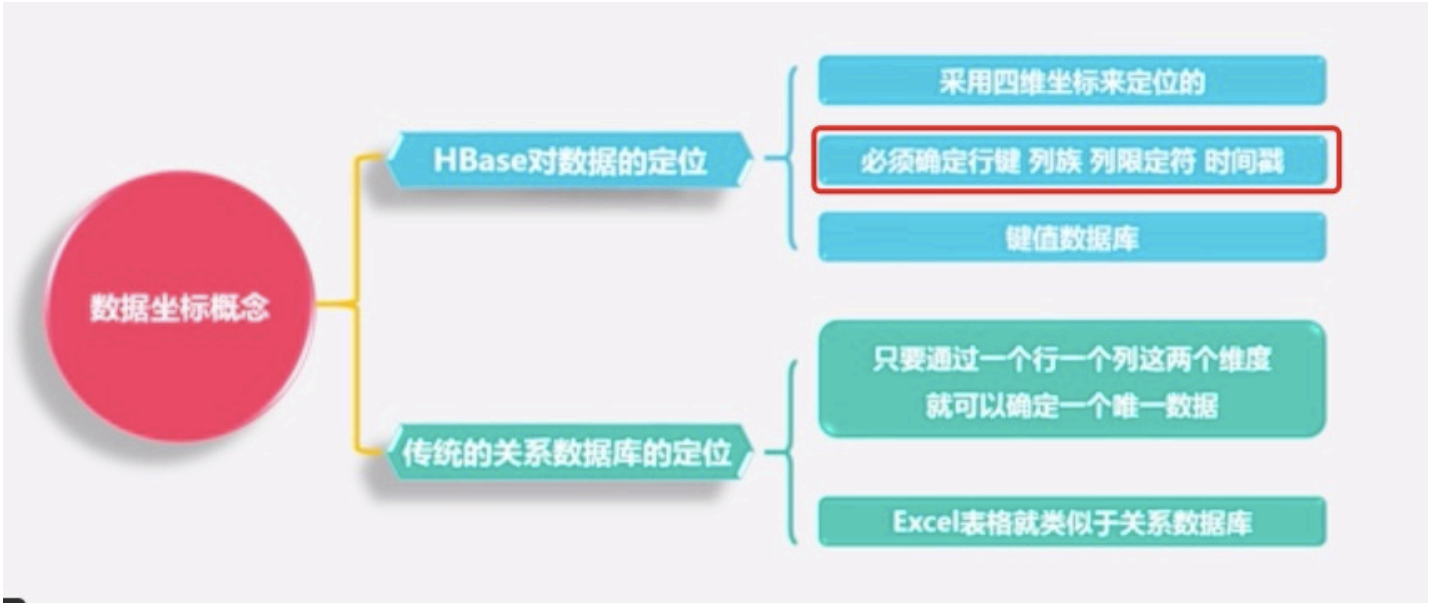
列限定符



时间戳：

在一个单元格里面，旧的版本会被保存，新的版本会通过时间戳来进行区分

数据坐标概念



面向列的存储

便于数据分析

3、HBase实现原理

HBase的功能组件

Master服务器

Region服务器

具体负责数据的存储


Region
维护和管理
Region服务器
数据的存取
客户端并不依赖于Master去获得位置信息
客户端要访问数据

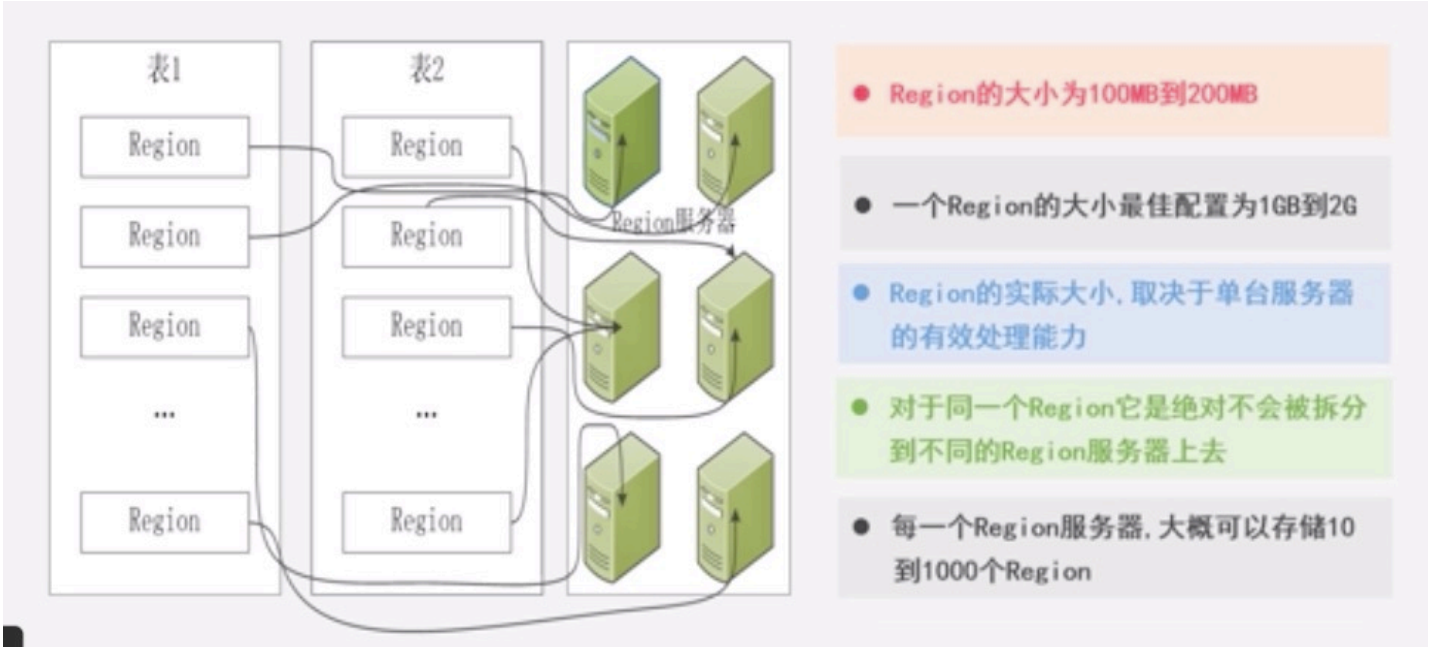
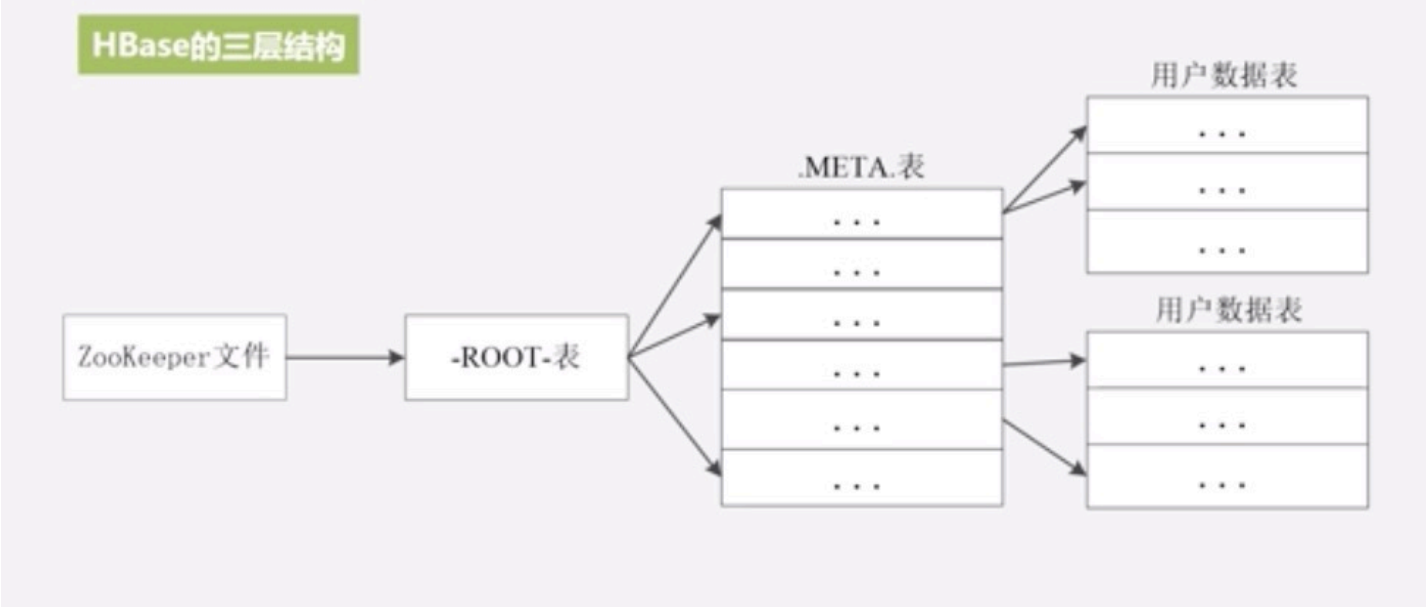


表1
表2
Region
Region服务器
Region的大小为100MB到200MB
一个Region的大小最佳配置为1GB到2G
Region的实际大小，取决于单台服务器的有效处理能力
对于同一个Region它是绝对不会被拆分到不同的Region服务器上去
每一个Region服务器，大概可以存储10到1000个Region

HBase三层结构



-ROOT-表：存取的是元数据表的元数据信息被存放到哪里

元数据的.META.表：负责存储具体数据存取到哪些服务器上

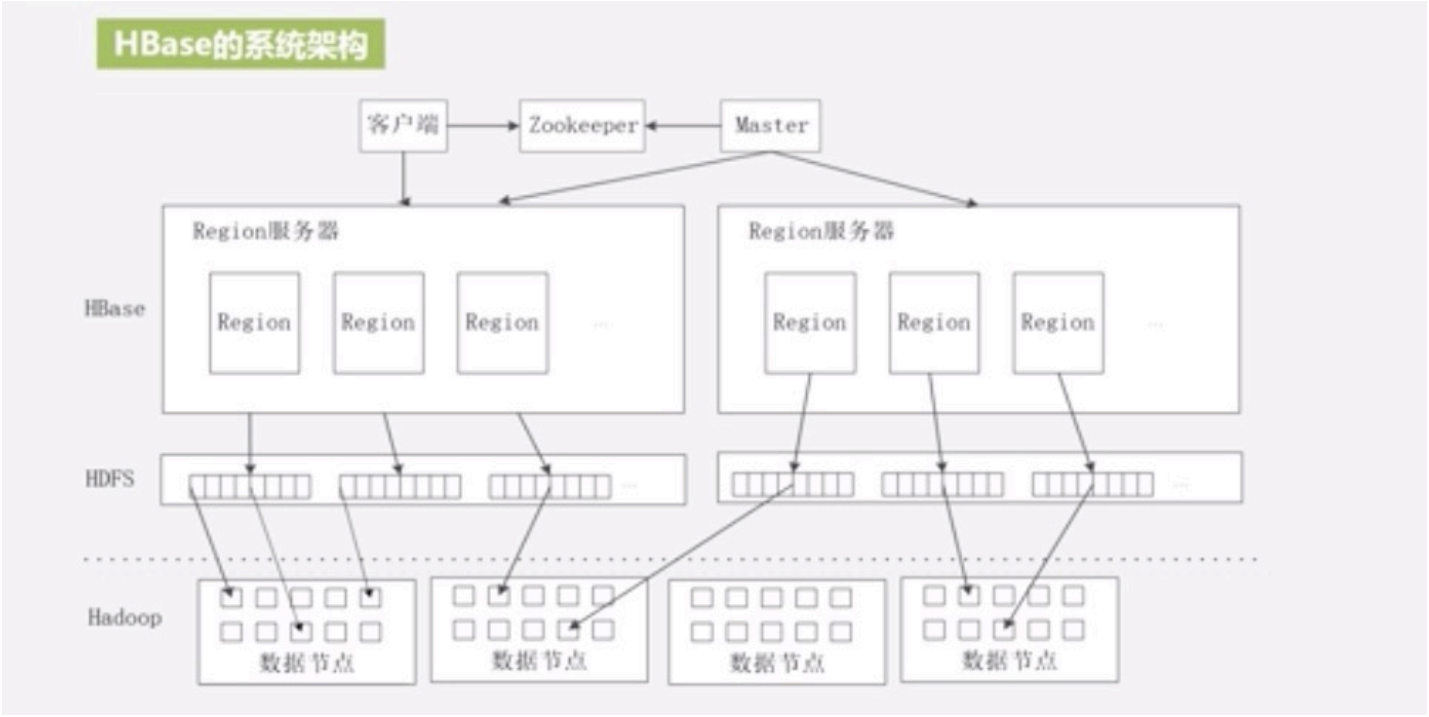
.META.表的全部Region都会被保存在内存中

HBase的三层结构中各层次的名称和作用

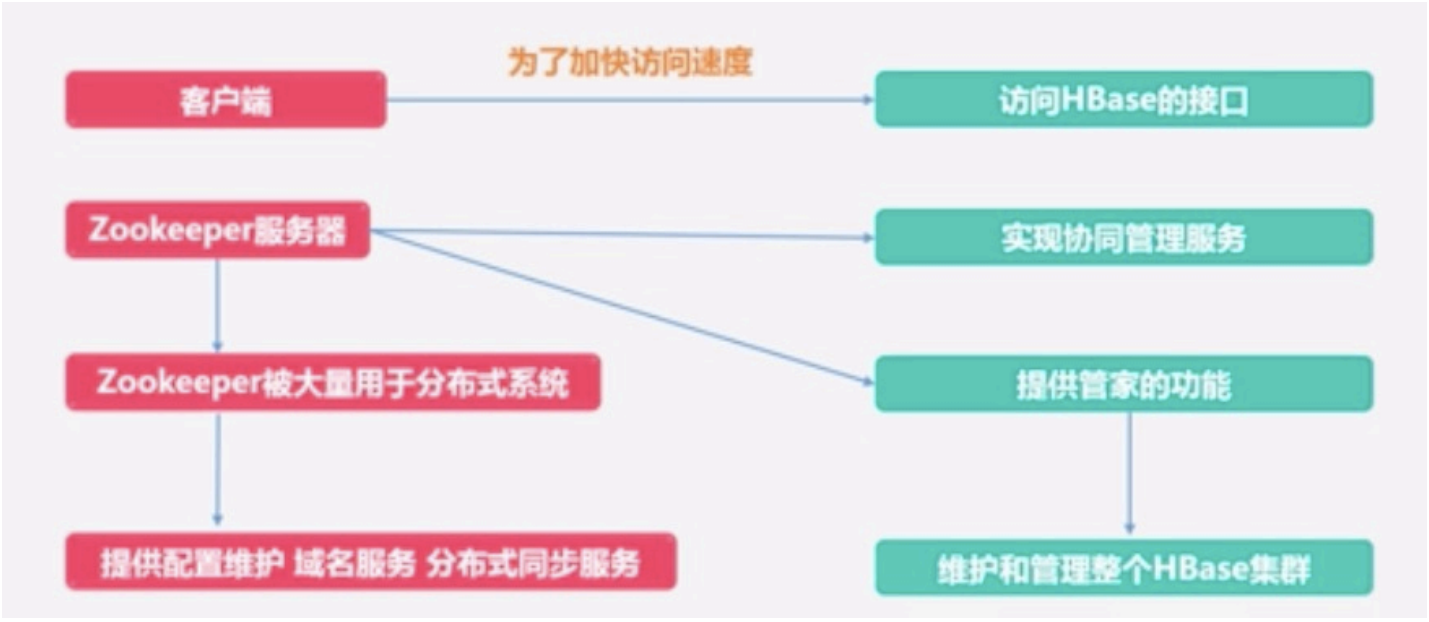
层次	名称	作用
第一层	Zookeeper文件	记录了-ROOT-表的位置信息
第二层	-ROOT-表	记录了.META.表的Region位置信息 -ROOT-表只能有一个Region。通过-ROOT-表， 就可以访问.META.表中的数据
第三层	.META.表	记录了用户数据表的Region位置信息， .META.表可以有多个Region，保存了HBase中所有用户数 据表的Region位置信息

4、HBase运行机制

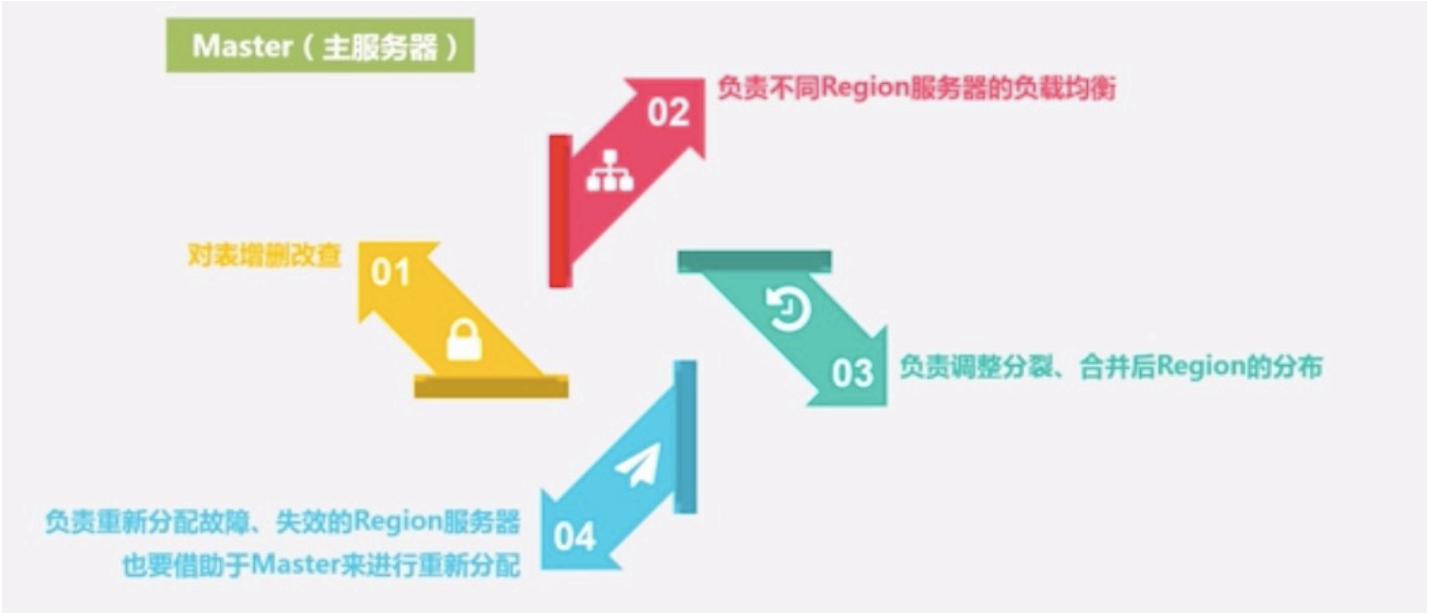
HBase系统架构



客户端和Zookeeper



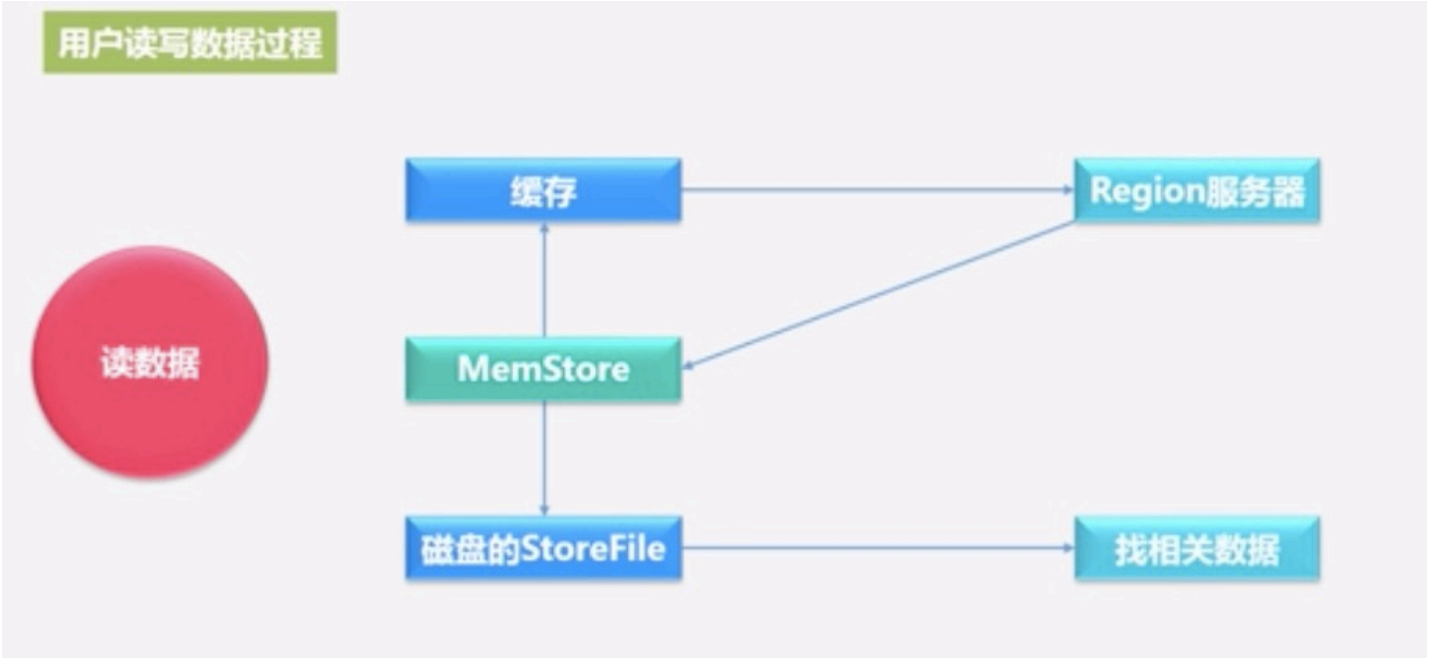
Master（主服务器）



Region服务器

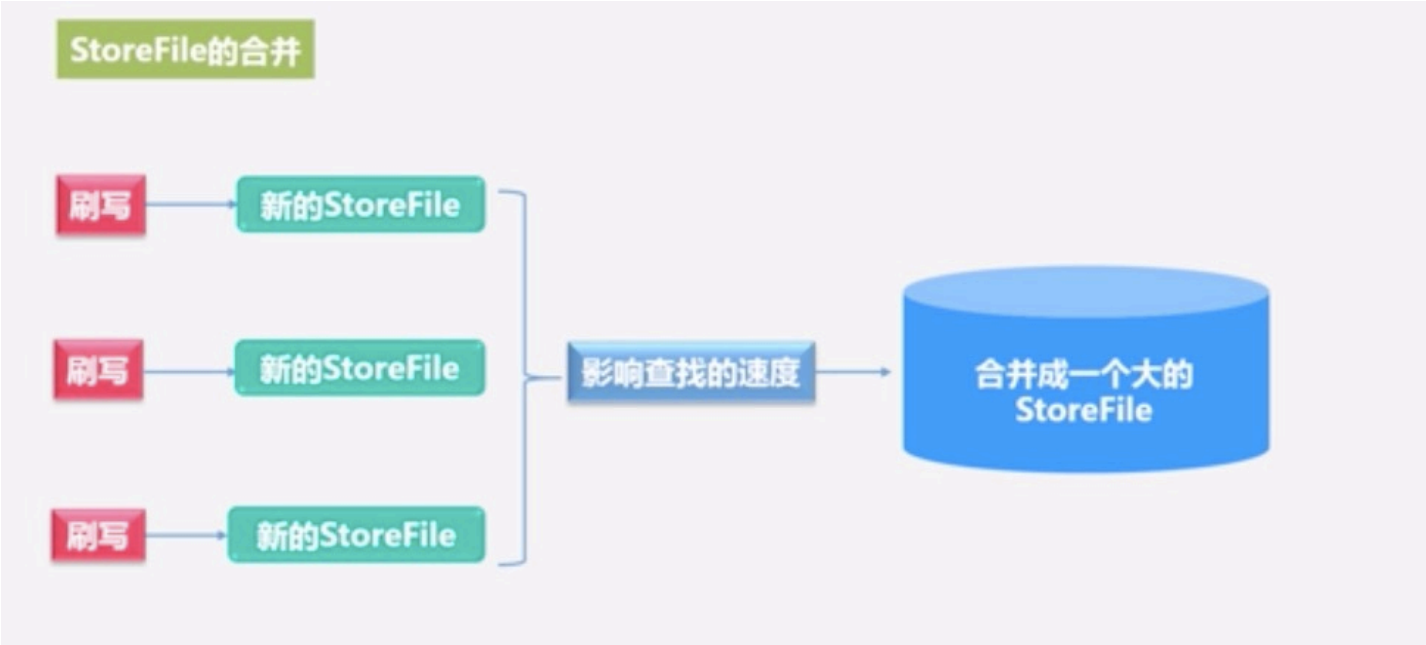
Region服务器向HDFS文件系统中读写数据

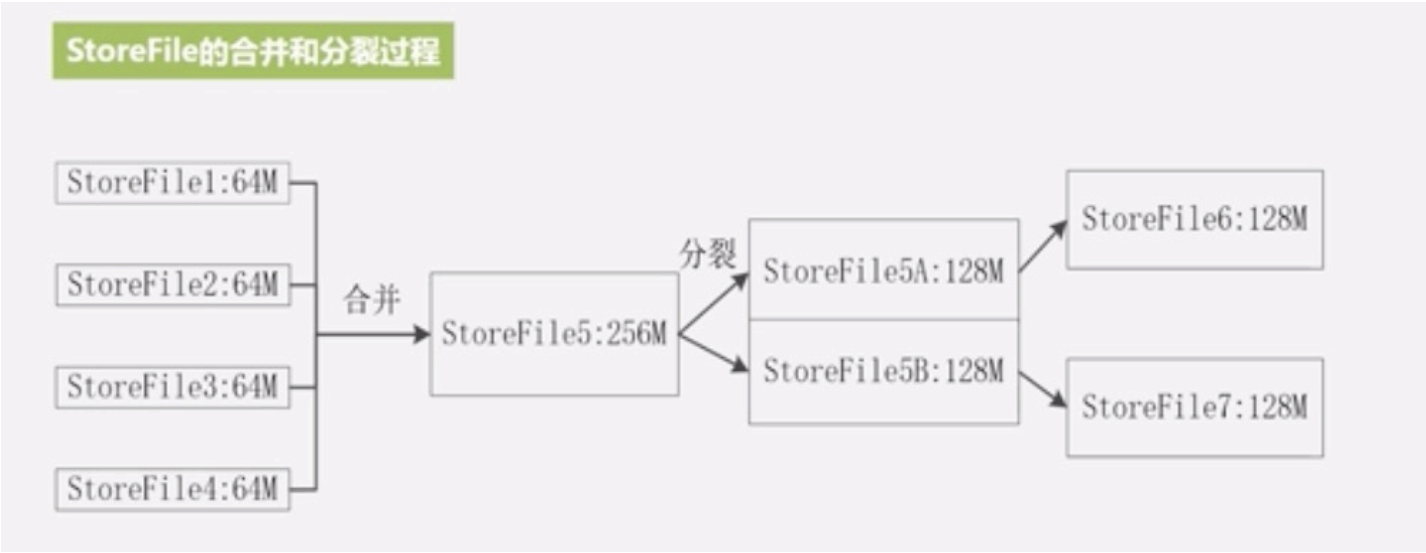
- 用户读写数据过程



- 缓存的刷新

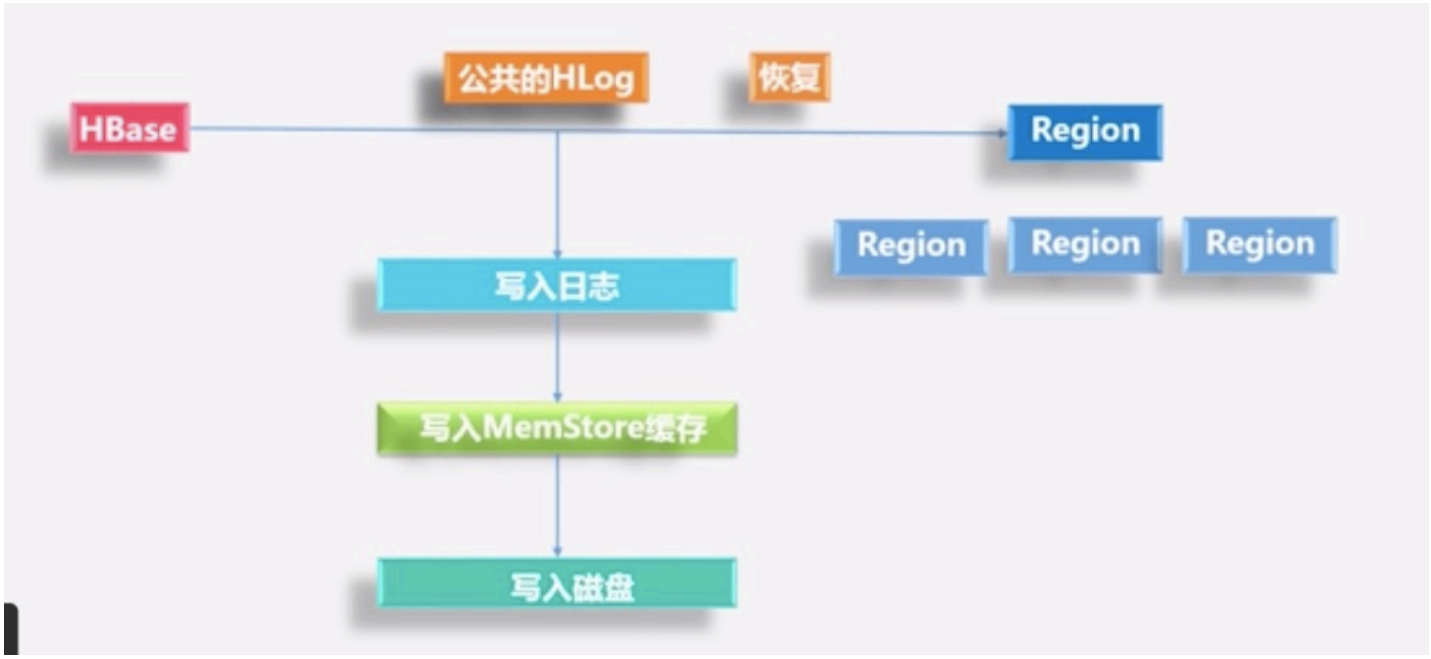
- StoreFile的合并和分裂==>即HBase的分裂操作





- HLog的工作原理

通过日志的方法来恢复数据

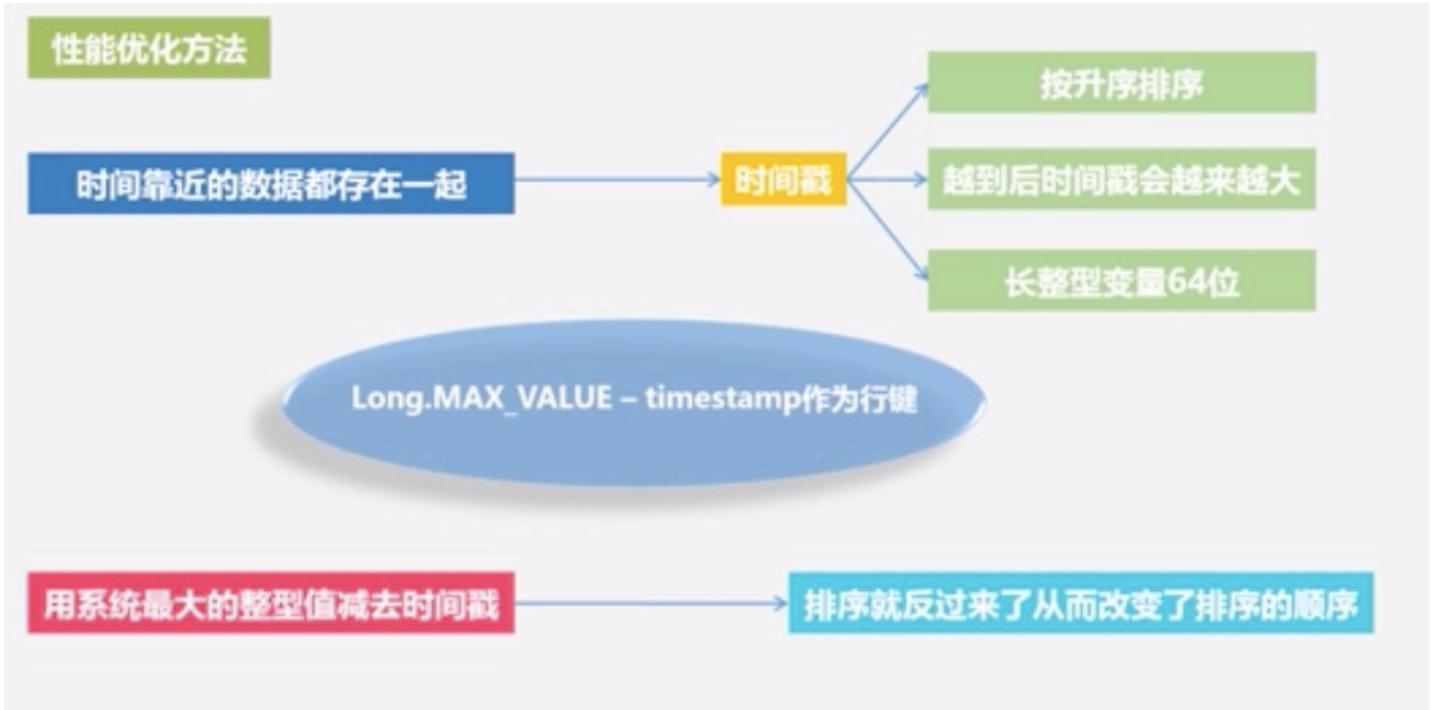


Zookeeper负责监视整个集群，告诉哪个服务器出现问题，并通知给相关的Master。则Master就会迁移故障服务器上的数据

5、HBASE应用方案

性能优化方法

- 时间戳

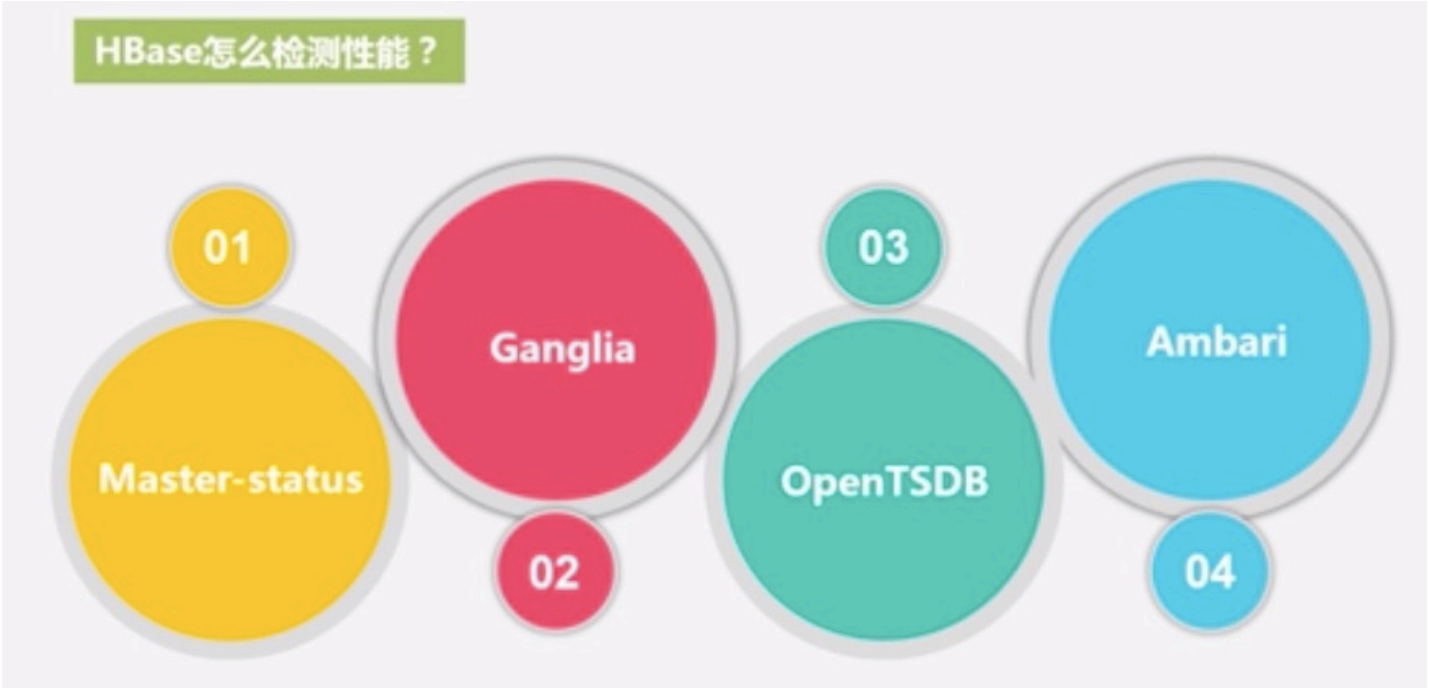


- 提升读写性能，将必要表方案Region服务器的缓存中

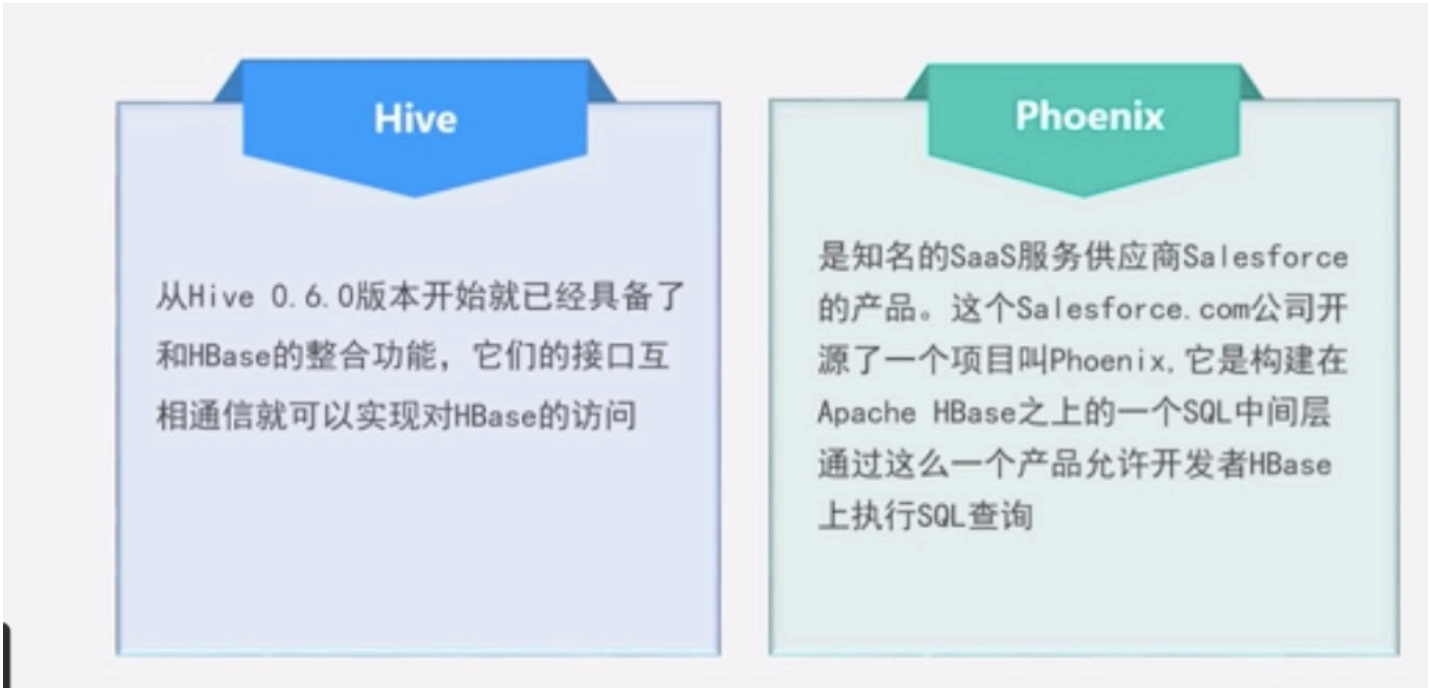
- 自动清除过期数据

HBase怎么检测性能?

性能检测相关工具

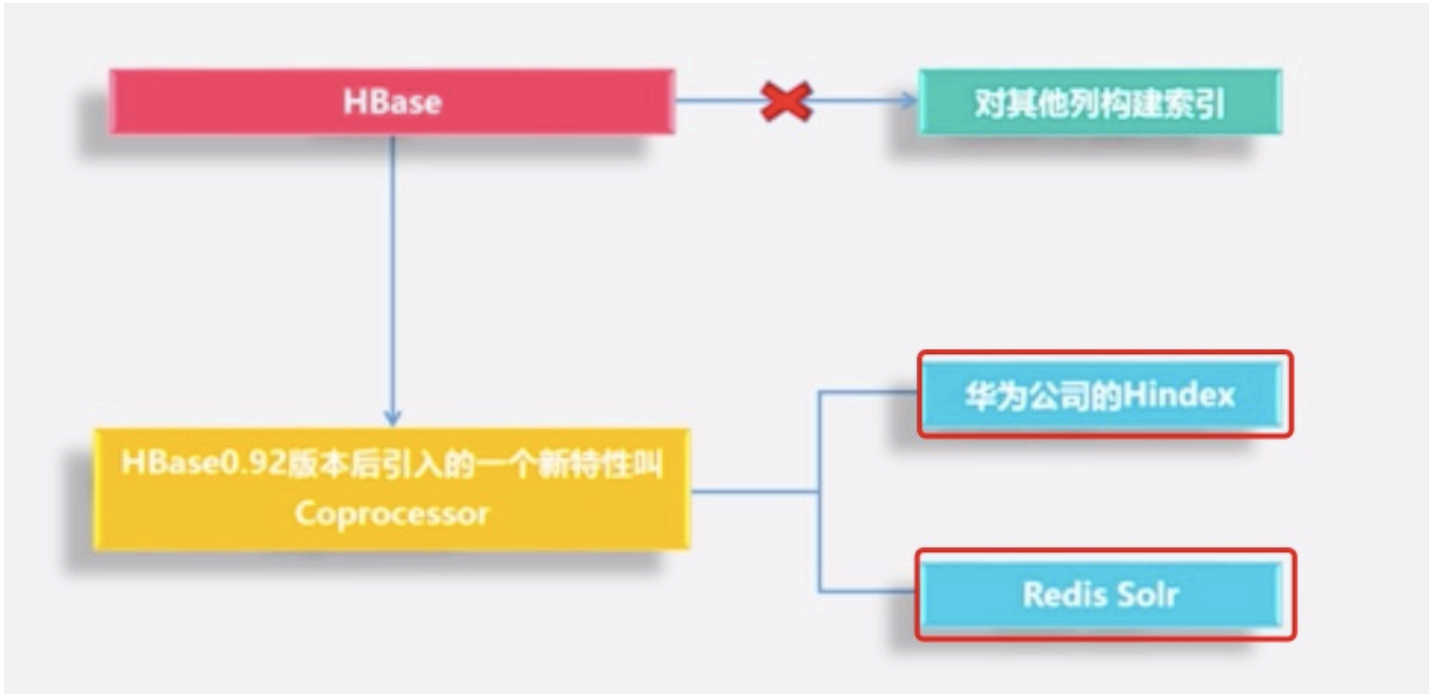


在HBase进行SQL查询工具



构建HBase二级索引

对不同的列进行索引



Hindex

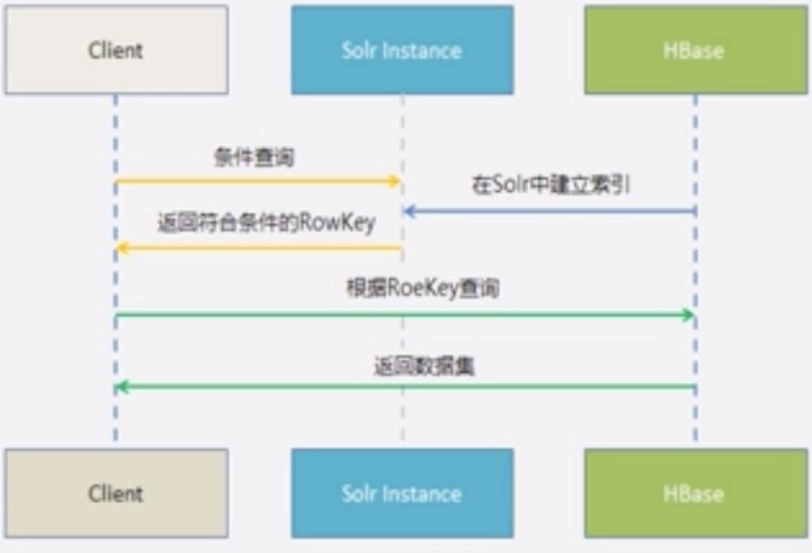
华为公司开发的，专门针对HBase数据库它是用Java开发的支持多个表索引也支持多个列索引，而且也支持基于部分列值的索引

HBase+Redis

Redis是一种键值数据库产品，能高效地管理键值对，由Redis数据库在缓存中管理索引，再定期把索引更新到HBase底层数据中

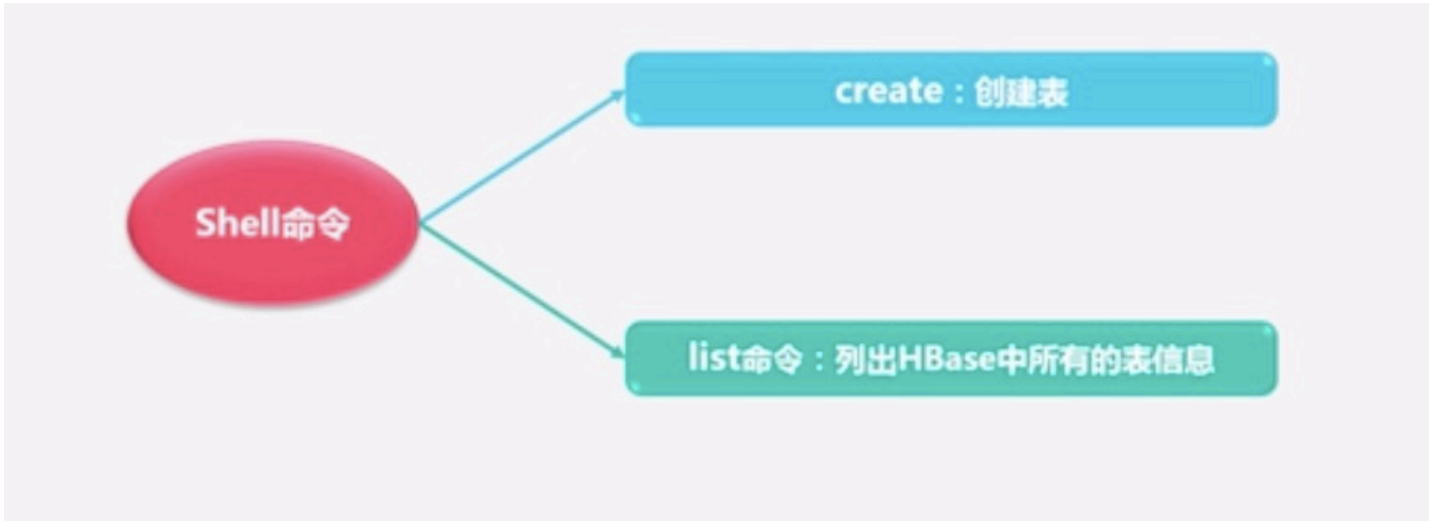
Solr+HBase

Solr是高性能，基于Lucene的全文搜索服务器。Solr构建起的是其他列和行键rowkey之间的对应关系



6、HBase安装与配置

7、Shell命令



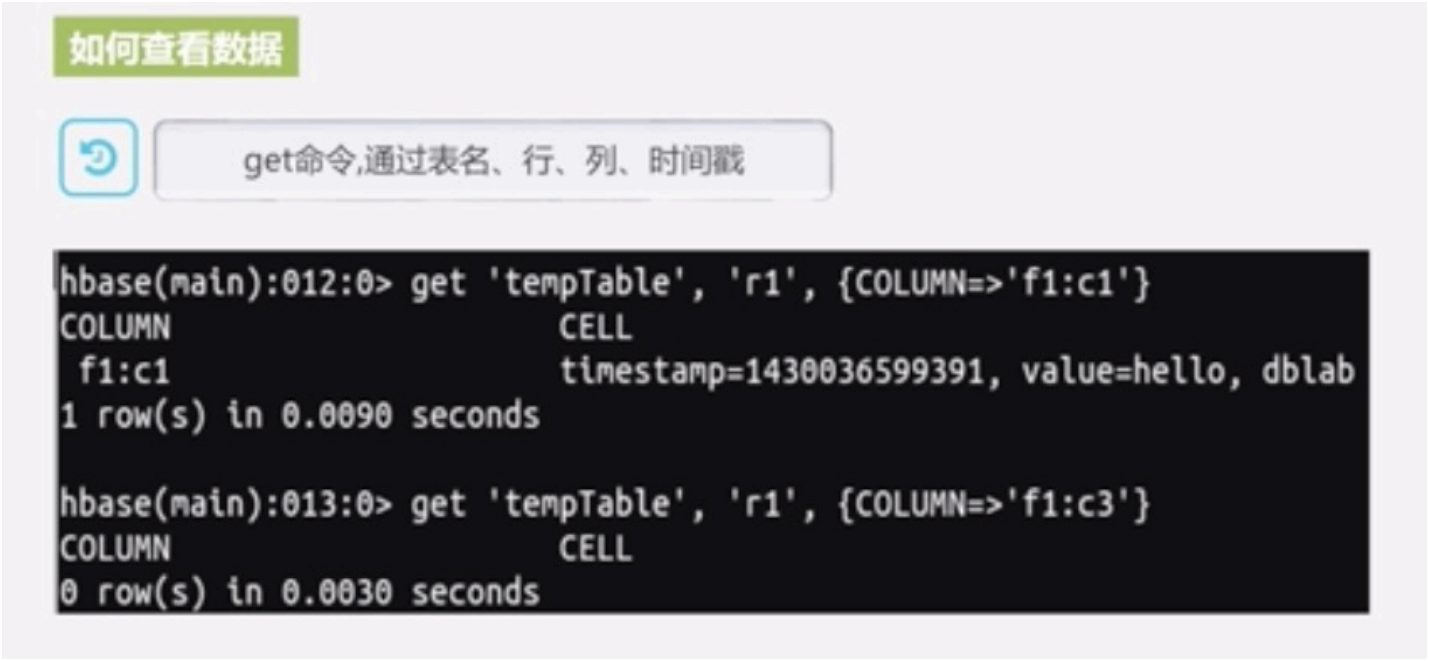
创建表

create '表名' ,'列名','列名','列名'

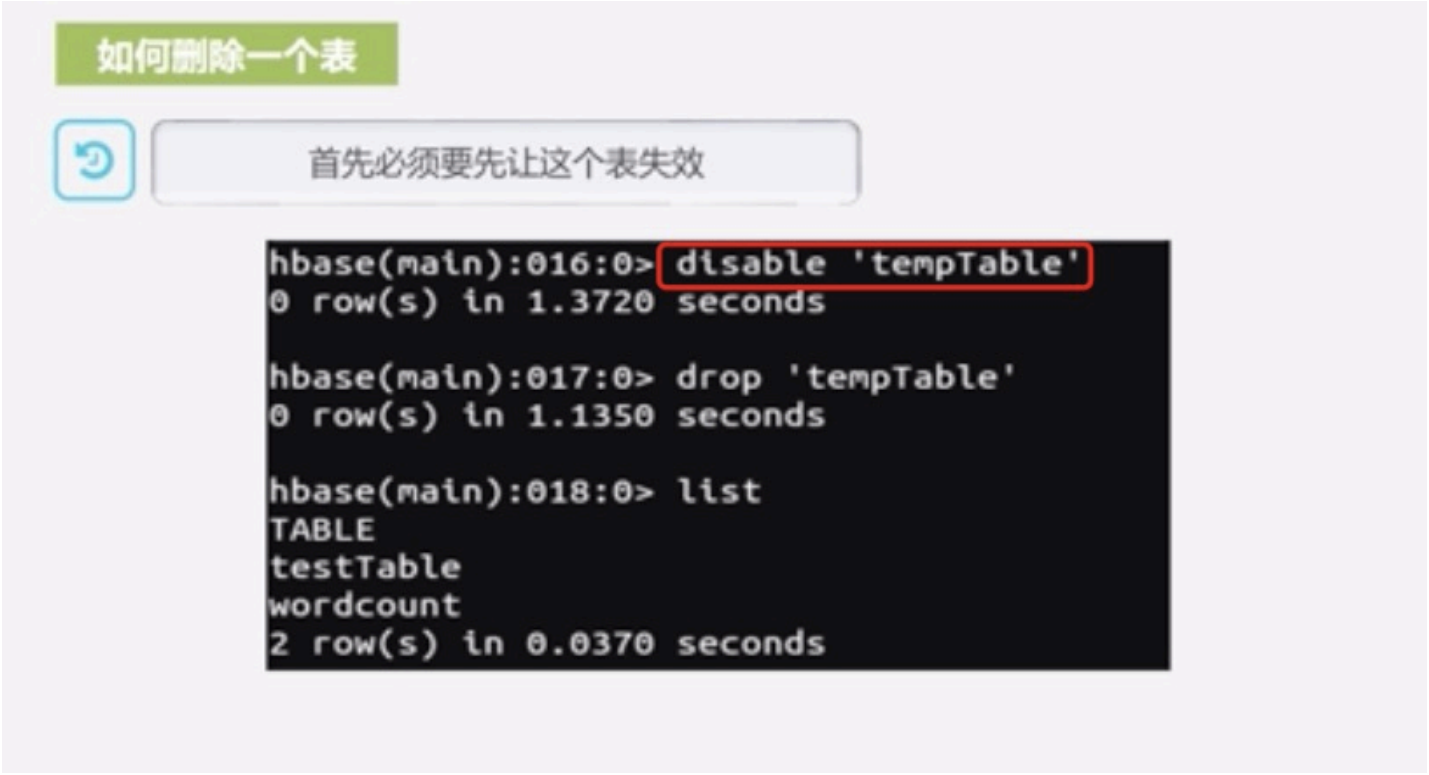
如何添加数据--put

列名称可以添加时自定义

如何查看数据--get

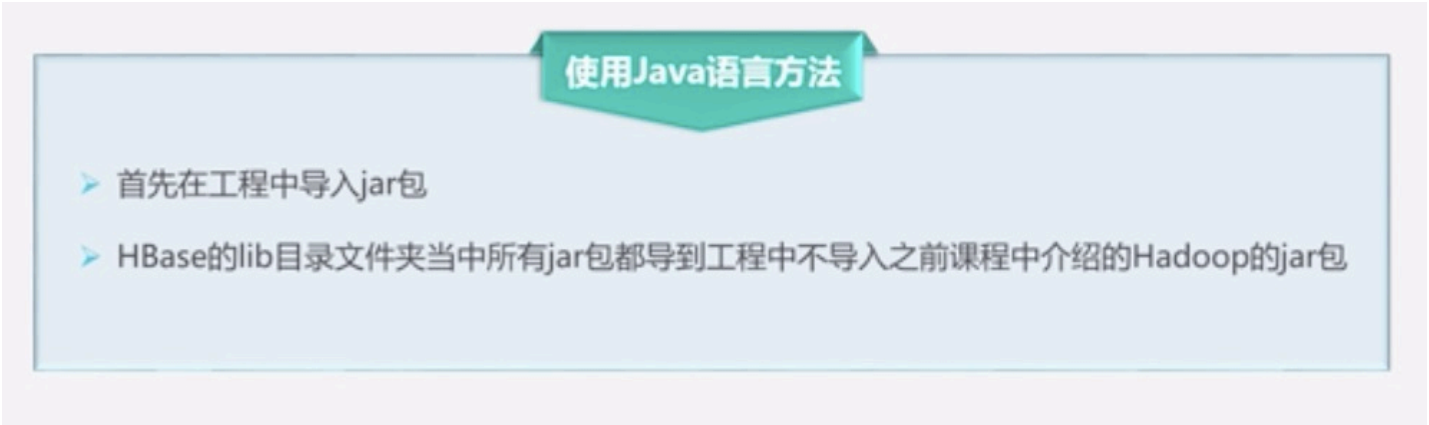


如何删除一个表--disable drop



8、HBase常用Java API及应用实例

HBase使用Java API准备过程



实例

整体过程

建立连接、关闭连接、创建表

- 建立连接

- 关闭连接

```
//关闭连接
    public static void close(){
        try{
            if(admin != null){
                admin.close();
            }
            if(null != connection){
                connection.close();
            }
        }catch (IOException e){
            e.printStackTrace();
        }
    }
```

- 创建表

```
/*创建表*/
 /**
  * @param tableName 表名
  * @param colFamily列族数组
  * @throws Exception
  */
            public static void createTable(String myTableName,String[] colFamily) throws IOException {
    init();
    TableName tableName = TableName.valueOf(myTableName);
    if(admin.tableExists(tableName)){
        System.out.println("table exists!");
    }else {
        HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);
        for(String str: colFamily){
            HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);
            hTableDescriptor.addFamily(hColumnDescriptor);
        }
        admin.createTable(hTableDescriptor);
    }
    close();
 }
```

myTableName

表示要创建表的名称

colFamily

表示要创建的表包含了哪些列族

HTableDescriptor

专门用来对表进行信息理

在运行程序时，需要指定参数myTableName为“student”，colFamily为“{“score”}”
程序的运行效果与如下HBase Shell命令等效：create ‘student’, ‘score’

- 添加数据

```
/*添加数据*/
/**
         * @param tableName 表名
         * @param rowkey 行键
         * @param colFamily 列族
         * @param col 列限定符
         * @param val 数据
         * @throws Exception
         */
    public static void insertData(String tableName, String rowkey, String colFamily, String col, String val) throws IOException {
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Put put = new Put(Bytes.toBytes(rowkey));
        put.addColumn(Bytes.toBytes(colFamily), Bytes.toBytes(col), Bytes.toBytes(val));
        table.put(put);
        table.close();
        close();
    }
```

表名 行键 列族 列名

Val：表示那个单元格的数据

指定参数

- 浏览数据

浏览数据

```
/*获取某单元格数据*/
/**         * @param tableName 表名
            * @param rowkey 行键
            * @param colFamily 列族
            * @param col 列限定符
      * @throws IOException     */
  public static void getData(String tableName,String rowkey,String colFamily,String col)throws  IOException{
        init();
        Table table = connection.getTable(TableName.valueOf(tableName));
        Get get = new Get(Bytes.toBytes(rowkey));
          get.addColumn(Bytes.toBytes(colFamily),Bytes.toBytes(col));
        Result result = table.get(get);
        System.out.println(new String(result.getValue(colFamily.getBytes(),col==null?null:col.getBytes())));
        table.close();
        close();
    }
```

Table	get类	Result类
用来获取相关表信息进行管理	用来查看具体的相关单元格数据	管理输出的结果

指定参数

浏览数据

比如，想看一下张三的English成绩

```
getData("student", "zhangsan", "score", "English");
```

用Shell去读取相关数据

```
get  'student','zhangsan',{COLUMN=>'score:English'}"
```