整个的课程，起初我们最一开始是学习了mapreduce的学习和实践对吧？那通过一些实践大家想必基本上都掌握一些mapreduce的一些开发流程的技巧，那大家觉得之前mapreduce学的难不难？起码你学完了mapreduce，大概有一种处理大数据思路大概是在脑海里，基本上是有这个大概的印象是吧？

那么接下来呢不管怎么说，你想要实现一个功能我们得去开发一些一系列的代码是吧？去把那个一些要实现的思路去实现对吧？尽管有的时间要解决这些问题并不是很难，但是毕竟你还是要花时间去思考对吧？并且要花大量的时间去调试去实现，有没有一种就是能够又完成想要的功能，又不想在代码上能够少花时间，又不想花太多的调试的一些精力呢？那么这个时候相当于从这一节点我们就抛开了一个新的内容，就是一个hive，hive就是解决以上几个方案的工具，当然我们从客观上来讲并不是说所有的问题都能够给hive，通过就是hive它比较适合胜任一些数据统计，数据分析类的工作对吧？那毕竟我们自己开发的mapreduce你可以实现你的任何的想法，就是你有一些什么天马行空的思路方案等都可以通过mapreduce通过代码的形式很灵活的方式实现，但这个时候不考虑它的成本是吧？但是这个hive它是一款就是解决某一些局部问题的一些工具，所以客观上来讲并不是所有的问题都适合用hive来解决，所以用hive的时候你必须按照它力所能及的那个事情去实现。

那当然有人说我不想用mapreduce是吧？我就想用hive，那hive能不能完全替代mapreduce呢？那如果说非要问这个问题的话呢？那么其实答案是可以的，你可以用hive完全替代mapreduce，但是说实话这个要让hive来完全替代mapreduce这个过程可能要比你单纯去开发mapreduce可能更麻烦，所以每一个工具它的适合的场景不一样，然后每一个工具它在整个的项目开发过程中它所处于的那个角色也是不一样的，不是说所有工具都是很通用的，每个工具都有自己适合的场景也有不适合的场景对吧？

上次我们讲的hbase,，其实也谈不上什么区别，虽然这两者东西都是涉及到一些数据表的一些创建，还有就是往表里面查数据对吧？虽然感觉这个操作起来很类似，但是这个hive和hbase他们两个完全不是一个层面的东西。

Hive就是一个mapreduce的一个封装，所以hive的意义就是在业务分析中能够将用户容易编写sql语言转换成一些复杂难写的mapreduce的一个过程，就是大家不是觉得mapreduce写起来比较麻烦吗？我能不能用hivesql语句，sql语句很短很小然后就可以完成一个很复杂的功能对吧？那能通过简单的方式来替代来完成，其实它的本质还是mapreduce还是集成框架，所以hive你可以当作一个集成框架。

然后另外一个就是从hbase这个就是什么呢？我们学过hbase大家都很熟悉了是吧？Hbase你可以认为是一个HDFS的一个封装，刚才说了hive是mapreduce的一个封装，而hbase是HDFS就是分布式文件系统的一个封装，那它的本质是数据存储用的，是一个nosql数据库，然后hbase它是部署于HDFS之上的，那它的主要的功能就是克服那些HDFS在随机读写方面的那些缺点对吧？如果你非要问Hbase和hive之间到底有什么区别，那就相当于mapreduce和HDFS之间有什么区别。所以HDFS和mapreduce两者之间你要谈区别，你这个问题问的有意义吗？都不是一个层面的东西对不对？

为了学这个hive我们从三个方面来学习，分别是hive基础理论和hive搭建和hive的实践（如下图1所示）

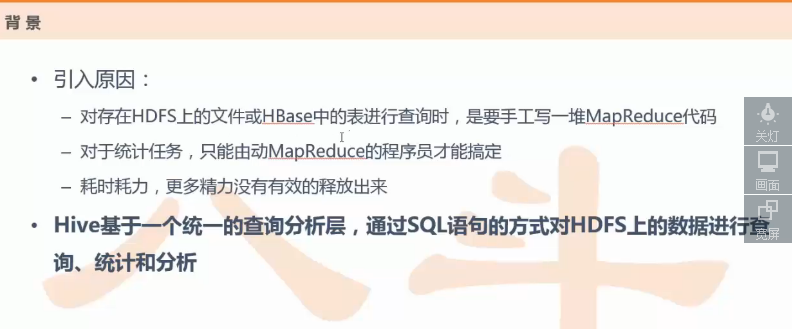
图1：



Hive在一些真正公司就是它的整个架构里面它的角色在工作场景中的一个怎么具体运用都会通过实践案例来说。相当于是做一个总结，因为我们从一开始学完mapreduce，storm等所以到这个时间为止基本上大家相当于可以说做一个大数据来说呢入门是应该问题不大了，所以到这个时间节点呢应该也有必要跟大家通过一些系统讲解把之前学的内容串起来进行巩固。

所以今天还是要知道一下hive是个什么东西对吧？那为什么要引入hive呢？这个原因就是为了替代mapreduce提高效率（如下图2所示）

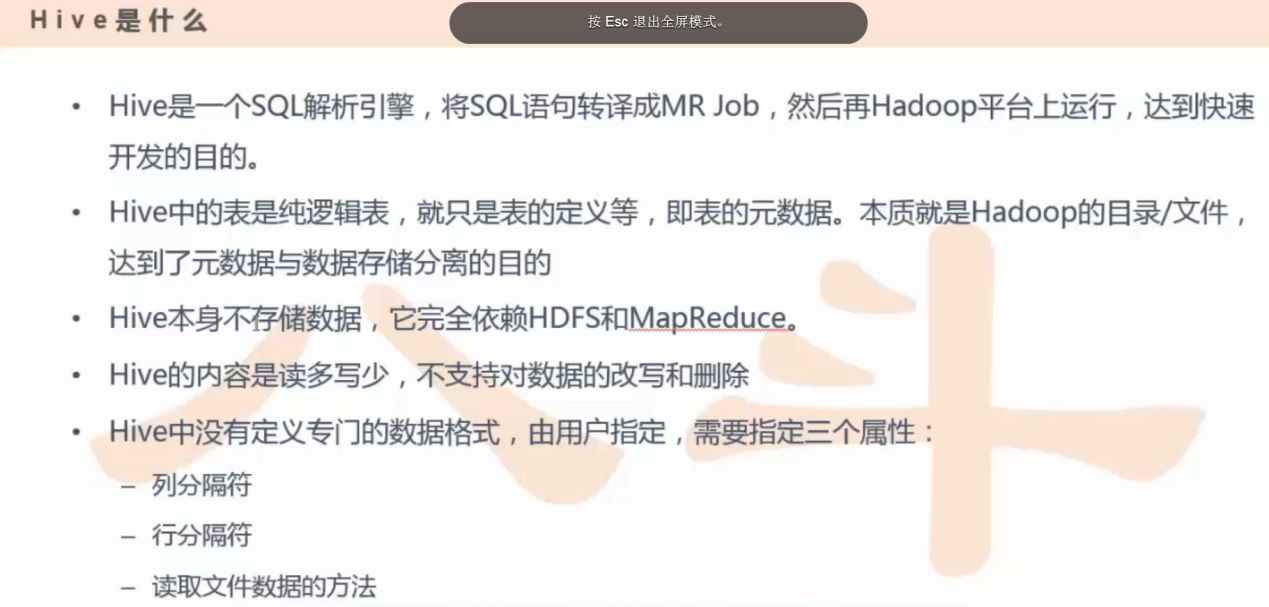
图2：



Hive是基于一个统一的查询分析层，通过sql语句的方式对HDFS上的数据进行查询，统计和分析，这句话看上去很普通，但是这句话隐含了很多的事情，也是出卖了它的一些特点，那怎么看出了的呢？也就是说第一个hive它适合做数据分析适合做数据统计，然后另外一个它使用的是sql语句，但是它这个sql语句跟我们之前的mysql等数据库的sql语句的操作还是不一样的，所以大家之前如果是有过操作mysql等数据库的经验的话，你来学hive的话这个过渡相对容易一些，但是hive的sql并不是一个标准的sql这个大家要清楚，也就是说你不能够保证你拿一个标准的sql语句在hive上就一定能够跑通明白了把？它的sql语句类似是标准sql语句，但是它不完全等价于标准sql。另外对这个HDFS上数据的查询，因为数据的查询因为这个mapreduce操作数据通常是对HDFS的对吧？那我操作数据也通常是要对HDFS做操作，所以对HDFS做操作这个数据查询相当于就出卖了hive本身的并不能存储数据这么一个特点，所以要记住hive它并不能存数据，它就是一个集成框架，它跟mapreduce一样本质就是一个伪装，只不过是它通过一个sql的形式进行一个表达，所以hive在某一种程度上可以当作集成框架或者是代替简单的mapreduce来用。

好了接下来就是介绍一些基本的概念（如下图3所示）

图3：



这基本的概念大家应该也比较好理解，很多人会很好奇那你说这个hive的本质是mapreduce，那hive的mapreduce是怎么生成的呢？就是通过一个sql它有一个解析引擎能够把你的sql进行一些个转义得到一个或者多个mapreduce任务，并且这些任务能在hadoop进行运行，hive的表是一个纯逻辑表，什么意思呢？就是说它只有表的定义也就是说它只有表的元数据，本质上大家如果要看那个hive的话，其实它本质就是一些个目录一些文件明白这意思把？它相当于hive它就帮你表达了一些hadoop的一个路径上的一个数据的一个结构表达，它就存了这么一个结构。

并且hive不存数据，像这个mysql一样可以对某一个字段进行修改，但是hive是不支持数据的修改和删除，但是从0.14版本之后，hive就可以支持修改更新了，但是它这种更新如果你不把这个功能开启，它默认的结果仍然是不支持的，如果你想支持这个功能你还得添加一些额外的配置，所以从这一个角度来讲hive它是不太适合来更新数据的，这个为什么呢？大家应该心里清楚把？这个就跟我们这个hadfs上的数据相关了是不是？我们之前讲hdfs上的数据也是不太适合修改的对吧？你要么把他删了重新建一个对吧？要么就追加一个对吧？为了考虑性能，因为大数据场景都是少写多读，都是这种场景，所以不建议对这个数据进行频繁的做修改和删除。

好了另外一个hive它是没有一个专门的数据格式，需要由我们使用方来指定，我们可以指定一些个列分隔符和行分隔符以及你如何去读这个数据的方法，通常通过这样的方式去读

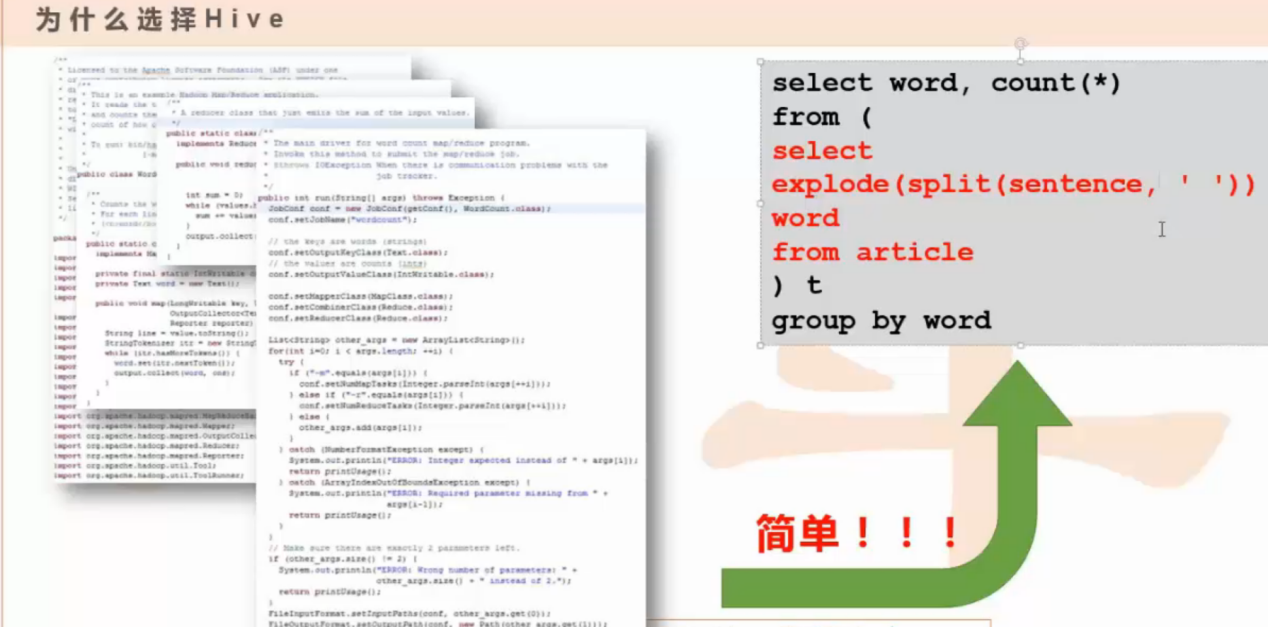
然后那个行分隔符的话通常就是一些空格然后还有\t，\001。然后列分隔符就是\n，还有就是读取数据的方法，有三种数据的格式，第一个就是TextFile,TextFile就是默认的文本，就是可以把这个数据直接可以打开，我们当时学HDFS上不是有这样的一个默认数据格式对吧？而且我们操作的也是比较多的这种格式对吧？但是这种格式有什么优点有什么缺点？优点就是可读性强，缺点就是数据不压缩然后你这个比较占资源空间，就是你的磁盘开销比较大，然后你在去对这个数据进行解析的时候开销也比较大。

还有一个SequenceFile，我们在讲HDFS上的时候，那这个SequenceFile是一个二进制的文件，这个SequenceFile的好处是hadoop提供的一种二进制文件，通常是这种<Key,Value>的形式序列化到文件中，所以这个文件是二进制的，然后这种文件你可以直接通过java接口就是Writeable进行序列化和反序列化，也就是说在java里面通常面对的是这种SequenceFile进行数据操作，通常你要写mapreduce的时候它通常是在那个通过java去开发的时候会常见的这种SequenceFile格式。但是你想通过python的方式去开发的话通常面对的就是TextFile的方式。

那么还有一种格式虽然不常用但是偶尔项目中还是用到一些那就是RCFile, RCFile这种格式是hive专门推出的，是一种面向列的数据格式。所以在hive里面常见的三种格式是就是一个TextFile和SequenceFile还有一个RCFile。

好了那有了这个hive福音呢，那我们看一下这么一个例子（如下图4所示）

图4：

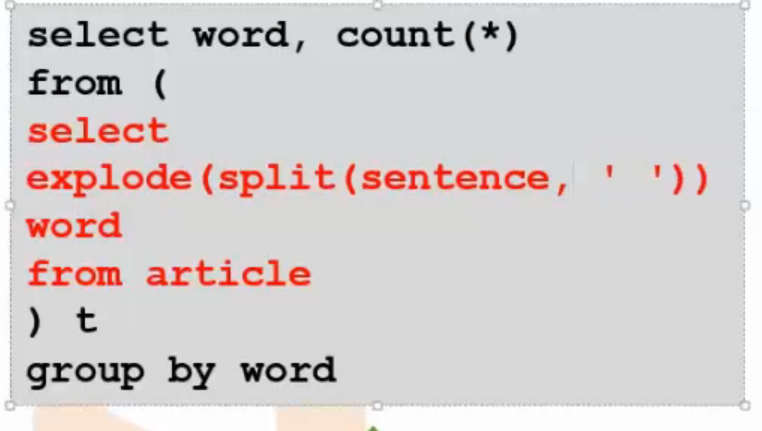


那比如我们为了要实现一个wordcount的话，那我们比如说要用java实现，那左边就有一大堆代码对不对？我们采用java去实现一个java的实现方式，那如果要是说我们要通过sql语言的方式也能实现同样的功能，那我们看看右边的那个代码例子，也就是说我们可以通过右边很短小的sql语句就能代替了左边这一坨很复杂的代码来实现相应的功能，而且你开发java的时候你一不小心里面就会有一些BUG对不对？然后还要调试，很繁琐对不对？而且你通过右边sql语句的方式的话不容易犯错，如果它是能跑起来的话那他就能跑起来，那如果没有跑起来那就是字母写错或者这个字段不存在格式没有搞清楚是不是？这些问题都会比较好查找，你要是写上java源代码的方式你去查问题，相对来说是比较复杂一点。

那好了，既然说能用sql语句来代替复杂的java去开发整个的一个wordcount，那么这个sql语句能看明白吗？怎么看呢？就是我图4中标红的是一部分，然后标为黑色的是一部分，也就是说我红色这部分输出的数据当作一个表供外层的（select word）sql语句来去查询用的对不对？

那这时候我们先去看外层的这个，先不去看内部红色的部分，内部红色的部分可能稍微有点复杂。那么黑色的部分它实现的非常直接，因为你要统计wordcount每一个word它出现的次数，那么这个时候select word，count（\*），也就是这个word到底出现几次对不对？那么后面那个group by就是对word进行一个聚合，把所有这个word的全部聚合在一块，这个怎么开发mapreduce大家应该没有问题对吧？大家之前都通过mapreduce的方式去实现wordcount对吧？好了那么里面的那些标红的是什么意思呢？里面这个红色的部分也是一个select语句，from了一个article，article就是一个文章，你把它当作之前那个英文文章，一个单词一个空格一个单词一个空格对不对？那么这个explode这个函数就可能比较重要，大家就可能比较清楚了，这个是什么意思呢？我们先不看这个explode函数，我们先看里面的那个split这个函数，这个函数split大家一猜就应该能猜出来，那从article它每读一行就是一个sentence，而这个sentence里面包含了很多的空格（如下图5所示）

图5：



然后通过split的方式会对这个空格以对空格的方式把这个sentence变成了一个数组，这时候大家想象一下这个数组可能会包含很多的单词对吧？那也就是说我这sentence它输入的是一个句子，它最后给我输出了多个句子，只不过输出的这个句子每一个句子都是一个单词，所以这个explode函数是干嘛用的呢？就是用来展开用的，就是说它适合于那些一到多的关系明白把？就是说它的输入就是一句话，输出就是多个词，就是一到多的关系，通常我们select的时候通常是一到一对吧？但是这个场景你毕竟要分词，分词的话肯定相对于这个句子分成多个单词就是一到多的关系，那么一到多怎么给它打平给它打开呢？就需要通过explode这么一个函数明白了把？

相当于这个函数explode这里面不是数组吗？如果是数组的话通过这个explode函数select的方式把这个每一个数组里面的元素就当作一个记录是吧？然后供外层去使用，就是一行转成多个行，couut(\*)的意思就是因为你通过这个word去聚合，也就是说我这个word出现多少次对吧？

比如我有一句话就是input : How do you do，那么这句话得到的output的结果是:

How

do

You

do

明白了把？就是我输入了一句话，可是输出却得到四句话了，然后这个时候你要对这个word进行一个聚合，那么这个do出现几次就可以通过count(\*)查出来了。

之前我们说过了hive里面的sql和传统的sql是不一样的，传统的sql你可以认为是一个标准的sql，但是标准的sql你想实现什么功能你完全的拷贝到hive里面去执行那很有可能就是执行失败，所以这两者是直接不能去通用的，那么既然不能通用的话，但是hive里面的sql和传统的sql写起来还是很相似的（如下图6所示）

图6：



那既然相似的话，我们来看这两者的区别，那我们已经知道了HIve的存储是存在了HDFS上或者是Hbase上，因为这个hive这里面的数据是可以和hbase进行一个对接，那这个后面我们会实验一下，这个在工作场景中也是很常用的也比较方便，另外一个比如们拿mysql来举个例子，比如mysql的数据是存在本地的文件系统里面去，那么Hive的数据是存在分布式系统里面去。

那数据格式，我用户自定义，然后标准的sql它的数据格式就是系统来决定。

数据更新，hive是之前不支持，虽然现在支持的话呢，但是也很少那么去用，但是传统的sql就可以数据进行更新操作是没有问题的。

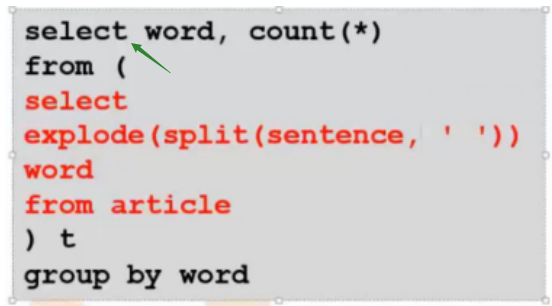
索引这一块hive里面之前是没有的，但是从0.8之后就开始有了，而传统数据sql它一直是有的，那么有了这个索引有什么好处呢？就是它查数据变得快了是吧？所以传统数据库要去查数据的时候它为了加快数据的查询它在很大的一定程度上去建一些索引。

执行这一块，因为hive它本身是通过集成框架去执行，然后如果是想通过传统数据库的话因为每一个不同的数据库它都有自己的一套执行引擎。

执行延迟这一块，那这个好理解，因为hive它本质上是一个mapreduce，你要跑一个mapreduce你不要寄托它能跑的了多块了，毕竟hive适合做一些离线的任务，所以它延迟是很高的，然后相对于传统的sql来说肯定是低的。

可扩展性这一块，hive也是比较高，那为什么说它的扩展性高呢？就是你可以用户自定义的函数，比如说是UDF,UDAF,UDTF，那么这三个函数到底有什么区别呢？UDF就是直接应用于select语句，就是这下图7鼠标箭头这个阶段

图7：



类似这种方式去运用，那么通常我们要搞一个UDF是要面临什么样子的需求呢？就是通常在查询的时候，需要对字段进行一些个格式化处理，举个例子，我们传统sql有没有大小写转换这个功能？但是hive里面可以通过UDF的方式去实现，假设实现一个大小写转换，比如说你的一个表里面的所有的字母都是小写，但是我想查询出来以后就变成大写，那这个hive里面不支持咋办呢？那我就自己定义一个函数，让他支持起来，但是你会发现一些特点，就是一进一出，就是一比一的关系，就是说你要对一些字段进行一些大小写转换你肯定是进来一个字段，然后我输出一个字段对不对？你进来一个小写字段，我输出一个大写字段，这是一个一对一的关系对不对？

如果是想实现一个多对一的方式的话我们就要实现一个UDAF，那么这个多对一的场景通常运用到什么地方呢？通常运用在上图7中的group by这个阶段，就是聚合的时候。

那和多对一相反就是一对多是吧？那就是UDTF。

然后对于扩展性来说，像传统数据库里面相对来说就比较差一点。

然后数据规模这一块，那么这个更好理解了，那hive处理就是一个大数据，所以它处理数据是很大的，然后通常用传统数据库相对来说这个处理数据是比较小的。

数据检查这块很重要，hive里面是读时模式，传统数据库是写时模式，这是什么意思呢？

这个很重要，大家要记住。

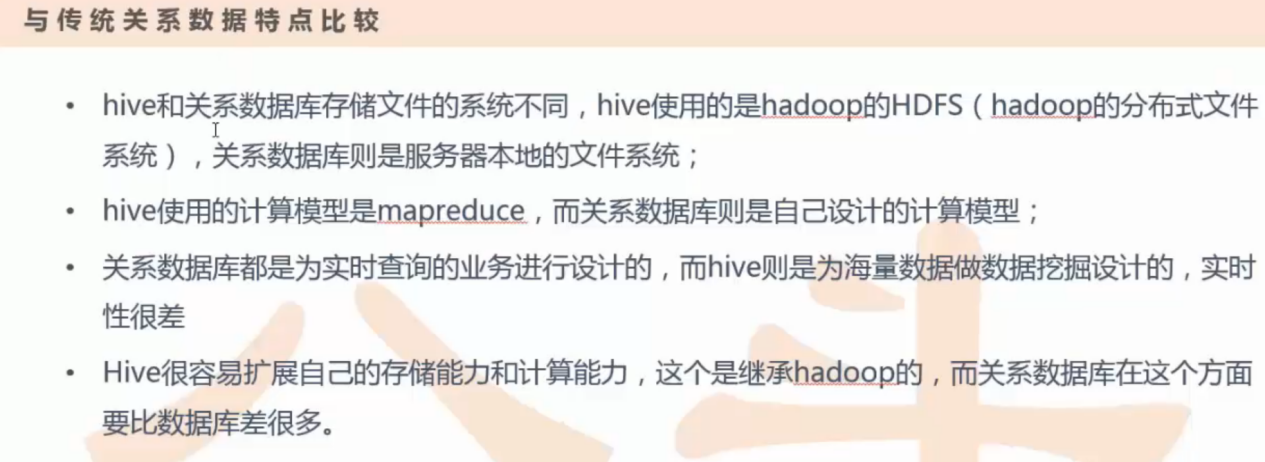
读时模式：只有hive读的时候才会检查，解析字段和schema，schema就是代表你数据的结构的表达，优点就是读的时候才会检查那么言外之意就是说写的时候根本就不检查，那既然它不检查的时候，那这个特点就是说它写的时候数据是非常快的，就像加载它数据的时候非常迅速，因为在写的过程中是不需要解析数据，所以它写的是非常快的。

那写时模式就像我们传统的数据库一样，缺点就是写的比较慢，但是它毕竟还是有优点的，优点就是它为什么会写的比较慢呢？是因为它写的过程中是需要去建立一些索引，然后又压缩并且数据一致性然后字段检查等等，它是要做很多很多的事情，如果它做完这些事情之后，数据真正录入进去之后，那好了你读的时候就是很放心的读了对不对？所以优点就是读的时候会得到优化。

所以在大数据场景来说呢，因为像在这个hadoop上的数据我们之前说过了，那我们面对的这个数据有很多很多的类型，比如有什么视频，图像，文本的各种各样的数据，但是这些数据的话你不能要求每一种数据它都是格式化，结构化非常好的，生活中存在着大量的非结构化数据，你不可能要求这个数据得完全按照特定的格式去存储或者是解析对不对？所以你避免不了在去它数据存储的时候去做一些个包容，然后这个时候你就可以加快它的入读速度，然后这个时候你读的时候去做一些个根据你针对的需要的需求去做一些个过滤。所以这两者差异大家知道一下。

好了接下来我们看一下（如下图8所示）

图8：



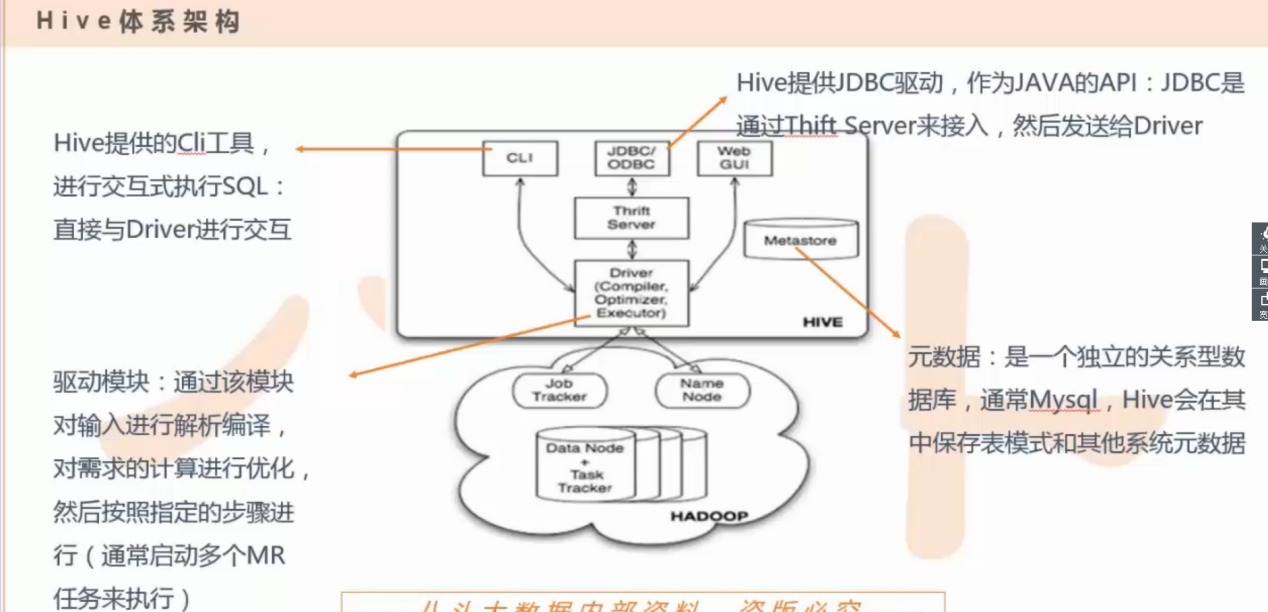
这个上图8就跟刚才我们所说的做一个总结，hive用的是HDFS关系型数据库，然后hive的计算模型是mapreduce，关系型数据库是自己的。

关系型数据库是为了实时查询业务用的，但是hive是为了海量数据，一旦是海量数据的话，那这个实时性就很差的是吧？

然后hive是很容易存储和扩展，因为hive的数据它是存储在了整个的hadoop平台上，hadoop平台好似很容易做一些扩展和一些个容错，所以hive它本身也是有这种易扩展的这么一种特点，它都是继承于hadoop上的，但是关系型数据库就差很多，就是大家这块简单的了解一下。

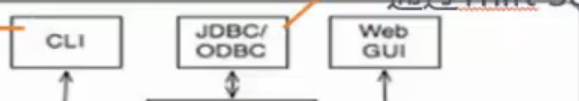
好了接下来我们来看一下整个hive的体系结构（如下图9所示）

图9：



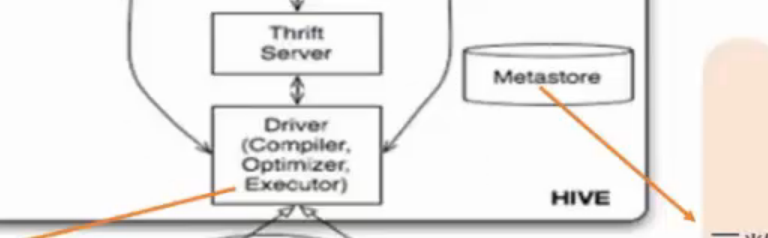
它整体来说其实包含了三个部分，哪三个部分呢？，就是如下图10所示为一个部分

图10：



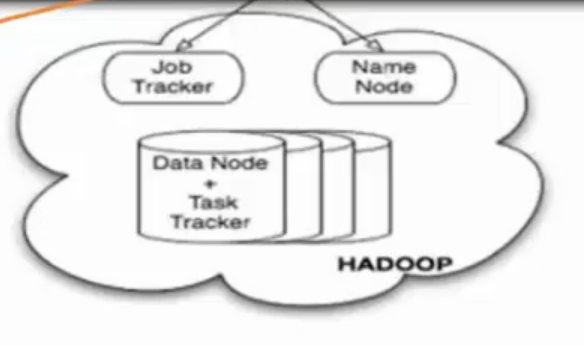
然后中间这一部分为一个部分（如下图11所示）

图11：



然后下面这一部分也为一个部分（如下图12所示）

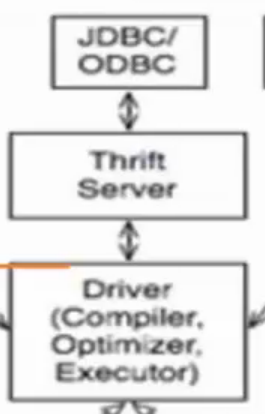
图12：



所以分为了三个部分，那么这三个部分分别代表了什么意思呢？就是上面的这一部分是用户接口，用户接口有CLI就是客户端Client对不对？，你可以理解成一个client终端，这个终端大家没有问题把？比如像我们之前学的那个hbase的时候，我们敲入hbase shell回车，你过一会你就会发现它有一个hbase然后有一个大括号，你就可以在里面做一些个put或者是get等操作，相当于它是命令的一个解释符，就在这个工具它已经给你提供了这么一个终端，类似zookeeper里面不是也有一个client对不对？它也提供了一个什么zookeeper有一个大括号然后你后面就可以做一些个对文件的目录数做一些查询然后数据的读写对不对？

还有一种就是比如通过一些java的方式你可以写一些个代码，然后代码的方式去调用相应的什么JDBC然后做一些类似hive的一些功能，所以通过这种语言的方式或者通过统一这种接口的方式来接入进来的一些请求就走（如下图13所示的这图里面的线）

图13：



但是hive有一部分是通过web页面的方式来去做一些个sql的一些提交一些查询，相当于最上面的那三种是提供给你不同的与客户交互的一个方式，好了那不管是哪种方式的话呢它都是相当于把你的这个任务提交到来了，把你那个hive这个任务就提交到了你整个的一个hive的系统里面来了。

那么这里面还有一个很重要的模块就是驱动模块（Thrift），它会帮你把这些最上面的客户端或者这种用户接口它请求过来的这些命令，通常这个任务的提交都通过命令的方式，你比如说你写了一个sql，相当于这个sql语句也就是一个命令，你回车它这个sql语句就会帮你做一些你想要做的一些事情，就相当于是命令与驱动的，那么通过这个会帮你把这个命令做一些个编译，又帮你做一些个优化，然后生成一些mapreduce，相当于是干嘛的呢？它就是属于一个语句转换，它相当于就是把一个sql语句转换为一个mapreduce或者多个mapreduce，但是这个转换mapreduce的过程中间有编译，有这些优化然后做一些个执行，那所以你会看的出来job它是整个hive里面的核心，扮演了整个的一个用户查询和mapreduce的任务，转换并执行这么一个角色。

还有一个很重要的数据就是元数据，元数据这块其实我们之前接触过很多很多了，这个元数据相当于是在图11中的metastore这一块，也如下图14所示，metastore它就相当于是一个独立的关系型数据库

图14：



一般来说你要是对这个metastore不做任何的修改的话它会默认有一个自带的一个本地的数据库，这个本地的数据库叫什么呢？默认的本地数据库叫：derby，那么这个数据库是什么呢？就是说它会在你本地目录里面，会在你的linux的本地会创建一个目录，然后这个目录里面会以文件的方式存一些你这个数据库或者是表格的一些个结构信息，那通常这种方式单用户模式常用，什么叫单用户模式常用？就是你自己一个人用，别人不跟你抢，就是这个hive就是你自己一个人用，但是你想一想从合理性的角度，你觉得这个线上的一个服务，谁会这么搞？不可能的对吧？线上的服务都是这种多用户模式，什么叫多用户模式？就是说我有一个单机服务器，我在这个单机服务器我多创建几个这样的账户，比如张三一个账户，李四一个账户，然后每一个账户它都会按照自己的名称然后登录是吧？通过跟用户的方式去登录，然后每一个账户之间互相隔离的对吧？

但是hive我是想通用的形式给这两个账户，那这个相当于是由多个用户可以共同来去操作我这一套hive，这就是一个多用户模式，但是这个多用户模式这个还是有点局限，它只局限在一台机器上，只不过是这一台机器上我多建一个账户而已。

真正生产环境中肯定不是这样，生产环境中都是远程服务模式，那么对于这种多用户模式和远程服务模式它们通常会把这个元数据肯定不会存在一个本地数据库里面，一般通常hive的元数据都会存在mysql里面，为什么呢？你要想我mysql是可以单独部署的，我可以部署一个机器，我可以在远程一个节点上面，我也可以在远程搞几个机器变成一个mysql集群对不对？但不管怎么样，对其他人来说它就是一个单独存元数据的这么一个集群，那大家都可以共享大家都可以去用，那这个时候把整个的一个系统就变得更通用了对吧？，好了那这个时候可以通过网络的形式然后大家就可以共享他们之间的一些个元数据。