接下来讲一个非常重要的内容，整个hive的话，它的数据我们之前已经说过了，它的数据是存在了你的HDFS上的，那么hive有四种数据模型，一个是Table，一个是External Table，一个是Partition，还有一个是Bucket，但是前两者都是一个Table是吧？那这两个什么区别呢？一个是内部表（Table），一个是外部表（External Table），但是他们都是针对于你数据库表来说的，内部表和外部表之间的差异也是很大的，然后hive支持了四个数据模型，但是暂时来看先把Table和External Table这两个表格当作一个表来看，把这两个表合二为一，那么此时大概就分成了三个数据模型，所以如下图1中的表格写了三个数据模型（如下图1）

图1：



我们首先看这个Table，hive中的这个table和你传统数据库里面的table，这个概念上基本上一致的，也都是类似的都是一个数据表，那但是每一个Table在hive里面它都会有相对应的一个目录来存储数据，什么是目录呢？就是你在hdfs上会有一个路径，一个path目录来存储你的数据的。那假如说我这边有一个例子，我有一个表，这个表的名字叫做moblle\_user，你既然建了这张表，HDFS上找到相应的路径，因为你的hive是不存数据的，你既然有了数据的话那这个数据肯定是在HDFS上去找到的对吧？好了你如果有这个moblle\_use的话肯定会在某一个路径上找到，那这个路径比方说找到了/lbs/moblle\_user这个路径，那其中lbs这个是什么意思呢？这个是可以通过你的配置文件来修改的，这个就是代表的是你的数据仓库这么一个名字，但是在这个hive里面，如果你不修改这个它会有一个默认的，就叫数据仓库的名字，但是这里修改了，就叫做lbs。

重点是这个表格tablename的名字就是moblle\_user，你就可以在相应的目录路径下找到相应的table，这个table它不是个文件，它是个目录，那这个是我们的数据表。

那接下来有一个分区表（Partition），一个班级有男同学有女同学，那我按照性别去分区，那么男同学可能是分成一个区域，女同学分成一个区，这是你可以通过一些属性，可以把整个的数据分成一个个桶，就是类似于reducede的概念。

那么这个有什么好处呢？假设说通常我们在去hive里面查询数据的时候会扫描你整个表的内容，你会发现你会消耗很多的时间，而且做了很多没有必要的工作，比如通常在一个表里面查询数据的时候会带相应的条件判断，你要去查询的时候它要去扫描整个表的内容然后对每一条记录针对增加的相应的条件判断去做相应逐步的过滤，你会发现其实这是很耗性能的对不对？因为你的数据里面包含各种各种的信息，那么这一些信息对我们来说完全没有意义，你只需要获取到你条件判断里面的数据。

那么如果说我们要引入了这个分区表（Partition）这个概念的时候，就大大优化了性能，为什么呢？你的分区表在你创建表的时候，你需要指定相应的一个（Partition）一个分区的空间，就一个表它可以拥有一个或者多个分区，然后每一个分区单独存在你这个表的文件夹目录下面，你的表的文件夹是哪里，就是上图1的那个table，你这个table下面是可以存分区表的，例如我这里面分了两个区，一个区是action一个区是day，这个day就是一个日期，那么我们看一下这个分区表哪些字段是适合来做Partition的，就是说你这个表格这个记录里面有很多的字段，不是说所有的字段你都拿来做分区的，有的字段拿来做分区的话会大大帮你优化整个的查询效率，如果有的字段你拿来当分区的话，你反而额外增加了一些集群的压力，那么我们看下哪些字段是适合当做分区的，首先第一个就是这个字段通常鉴于你的条件判断，如果要是说你的这个条件，比如说你通常写一个sql语句，那通常这个字段是要求经常要在你的where判断里面要存在的，如果你这个字段通常不需要拿来做条件判断，那么这个字段是完全不需要做分区的。

比如某一天对数据进行统计，那么我就需要对这个一天进行过滤，那我就需要把这个条件要放到我的where里面去，那如果在大量的数据里面过滤，那还是很耗时的时候，那这个时候就提前建个表目录然后对时间做一个分区，比如说创建一个表目录名称为2017年11月11日做一个目录。2017年11月12日做一个目录，好了这个时候写数据的时候，我就要提前判断这个数据里面它是不是有什么时间字段，并且我按照这个时间字段去归到相应的时间目录里面去明白了把？

好了这个时候如果要是说大的数据被分散到存储与每一个以时间为单位的目录桶里面的时候，那接下来的优化就好做了，当我有一个查询语句，那我就要对这个条件判断里面的数据做一个查询，那么这个时候我直接定位到相应的时间目录就可以了，那么其他的不在条件判断下的数据都统统过滤掉，都不需要去读它，直接把这个相应的时间目录读出来即可，这样就可以大大加快了我的查询速度，这就是分区表的概念。

并且你要做分区的时候不仅仅是可以对一个字段做分区，你也可以对多个字段做分区，一个分区表它对应一个目录，如果两个字段同时做分区的话，那比如说我要查询day=20131020数据，那么我直接定位到是以day=20131020为目录的就可以了，这样的话它会在你建数据的时候就提前帮你把这些条件给你提前判断好，所以这是非常加快你的查询速度。

但是这里面大家想一个问题，如果说一个字段它也经常存在于where循环里面，但是这个字段它的取值是非常非常多的，比如说这个字段还有一个and的，你的用户名user\_name等于一个zhangsan，那么你觉得这个字段适合拿来做分区吗？答案是肯定不能拿它去分区，因为如果拿它去做分区，相当于你为每一个用户都要创建一个目录对不对？那么这样目录就会很多，这样会导致特别占用空间，因为每一个小文件会占用你的一个nodenema的一个内存区域对吧？所以为什么你的分区表这个字段只适合那些它的取值范围有限集合的那些字段，明白了把？

那这一块是非常重要的一个内容，这工作当中这是绝对会用到的一个优化，这个优化也效果非常好。

还有一个优化，工作中常见的分区就是拿时间，这个时间是绝对是非常非常常见的，一个时间还有一个就是比如说流量，pc还是平板，你这个手机的类型是ios的还是Android的，就是这些它是可以枚举出来的，它是可以内容是可以有限集合，而且这个集合也不要放的太大。

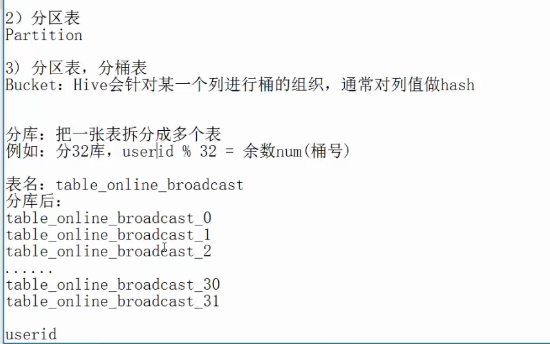
这是一个优化，这个优化相当于是我们从刚才的table，在这个table下我们可以去做一些个Partitiondeep优化，那继续，我们来看看在Partition的基础之上再做一个优化，这个优化就叫Bucket，或者这个table直接到Bucket，好了我们接下来看看这个Bucket是什么意思。

Bucket它其实也是一个类似于这么一个分区表，但是它的分区跟你的Partition分区不太一样，那么它的分区是怎么分的呢？它就是按桶分的，那怎么按桶分呢？我们先说一个场景，在mysql当中，如果你mysql里面这个表格它存储的数据肯定是有限的，你不可能存太多的记录，一般记录好像存到上亿以上，性能就不太行了，然后假设说你这个表里面有一个字段叫做userid，因为你一张表太大了你要是再往里面追加数据的话，这个表很有可能达到了你的mysql的一个上限了，那有什么办法可以存完整的数据又不想突破一张表的一个能力范围，那么想办法把一张表拆分成多个表，这个通常叫分库。

那么怎么分库呢？比如说我们要分32个库，因为通常这个数据是以userid为粒度的，都是每一个用户它的一些信息，这个userid是作为它的主key这个很重要的一个信息，那通常我们会对这个用户进行一个分桶，这个分桶很好理解就是一个哈希，userid取模32等于什么样的数字，然后这个数字代表它的一个桶号。

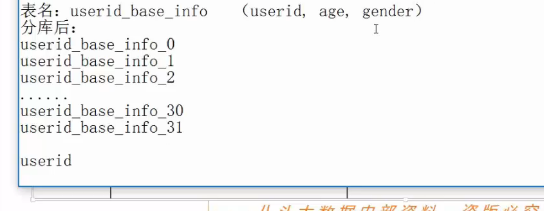
假设说原来这个桶叫做table\_online\_broadcast，分库后这个表名table\_online\_broadcast变成table\_online\_broadcast\_0和table\_online\_broadcast\_1这样一直到31张表，由原来的一张表存的数据就平担了32张表共同的存储，而且每张表存储的数据都是不会重复的，因为我会按照userid去做分桶这个大家理解了把？（如下图2所示）

图2：



好了这个时候这个Bucket就派上用场了，这个Bucket是hive会针对某一个列进行桶的组织，通常对列值做哈希（hash），那么你这个userid的余数是存在哪个桶上去了，好了大家想一想我难道就是为了解决这个问题我就单独搞了一个Bucket出来吗？比如像在hadoop平台上我有必要这么做吗？我有必要把它分成多个桶吗？本来这个数据就在HDFS上它可以存海量的信息，我还要再去搞一个什么分桶有必要吗？其实它本来不分桶，其实也可以存那么大的信息的对不对？如果单独从这一个角度来说你说你用Bucket我可以优化我的性能，这个显然是站不住脚的，那想一想那为什么还要搞一个Bucket出来呢？其实也是为了优化查询，它怎么优化查询的呢？比如说我有一张表，这个表是代表了一个举个例子这个表里面有useriid和它看了哪个music\_id，然后它这个账户这个userid它一共听了几次（num次数），好了那我还有一个表名，这个表名叫做userid\_bash\_info，关于这个用户的基本信息，因为它这个用户也是很多的，所以我也进行了分库（如下图3所示）

图3：

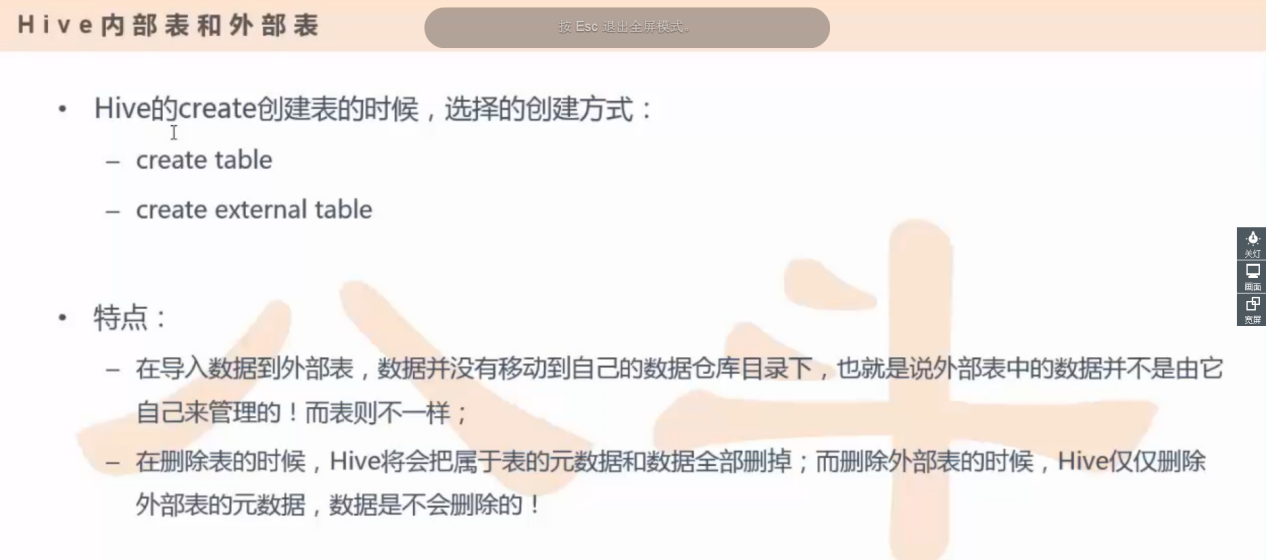


大家想一想如果想把这两张表做一个join的话呢，如果我不分区相当于我是把所有的数据直接做了个join对吧？如果分区之后是不是比如说我这个上图2括号中的userid我命中了table\_online\_broadcast\_1这个桶，那么上图3的括号中的userid我肯定也命中了userid\_bash\_info\_1这个桶对不对？那我是不是可以把这个表数据就变小了，本来是两个大表之间join，就变成两个小表之间的join了是不是？然后相当于这些没有关系的信息完全是不用考虑了对不对？这个就是方便了你的join，所以这是Bucket的一个非常重要的意义是在这个地方，其实这两个表都做了分区之后，如果是这两个分区表join它会自动激活，因为这个hive它本质不是mapreduce吗？它就会自动激活你的map端的有一个map-side join，就刚才我们所说的那种方式，自动激活两个小表之间的一个join，这就是优化查询的一个优点，第二个优点就是方便采样，就是说在处理大规模的数据的时候你在开发和查询的时候如果能在数据集里面我取很小很小的一部分样本我先试探一下，我先拿过来快速的调研一下，然后大概看一下这个数据的分布，我可以相当于通过这么一个点，我可以大概能够摸得清整个的一个数据的一个情况是不是？相当于就是一个采样。

那把这里面真正的Bucket它在那个HDFS上怎么存的呢？比如说它前面有一个Partition的目录下面会有很多的part0000，其实你会发现这个东西就是类似我们reduce的输出的一个路径对不对？

好了接下来我们看一下创建内部表和外部表很容易，有两个命令（如下图4所示）

图4：



create table和create external table，如果不加external 这个单词的话那么它默认创建的是create table内部表，如果加了external 这个单词的话就默认帮你创建了外部表，这两个表有什么差别呢？其实它最大最大的差别就是说我这个表一旦创建之后，并且我也往里写了一些数据，我也可以从这个表里面去查询一些数据出来，说明这个表已经建好了并且里面有数据，如果说一不小心把这个数据删除了，就是把这个表删了，就是通过hive终端的方式我突然把这个表删掉了，这个时候如果你创建了这个表你join这个表是内部表的话，那么不仅你的这个表结构丢失了，你这个数据也会丢失明白了把？如果你创建外部表的时候，你不小心把这个表删了没关系，你只是删了你这个表结构，但是你的这个数据没有删，你是可以做恢复回来的，大家理解这个区别了吗？就是建议大家创建表的时候尽量用外部表，因为能够避免一些你的误操作把你的表删除了，导致你的数据一起丢失这么一个风险，所以尽量创建外部表。

那么如果你不小心把这个表删除了，那么怎么恢复呢？这个恢复是很简单的，你先把这个表重新建一下，那么数据就会自动恢复，这个在实践的时候我们会说的哈。

好了这个时候我们再看一个Partition（如下图5所示）

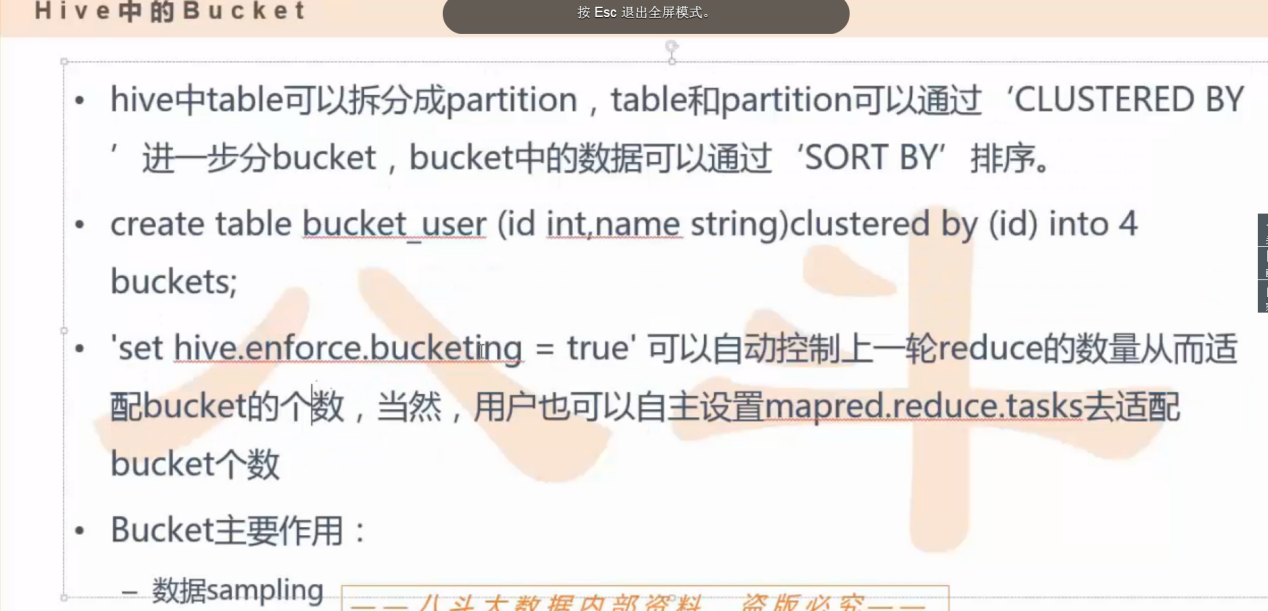
图5：



那么Partition这个我们就不说了，刚刚说过了，上图5中是一个例子。

那么Bucket我们也说过了（如下图6所示）

图6：



怎么开启Bucket呢？你需要通过一个图6中的一个set hive.enforce.bucketing=true这个配置

去配一下就可以了，那么它相当于是自动适配。

然后还有一个就是采样（如下图7所示）

图7：



采样这个上图7有一个例子，大家看一下这个例子就能明白。

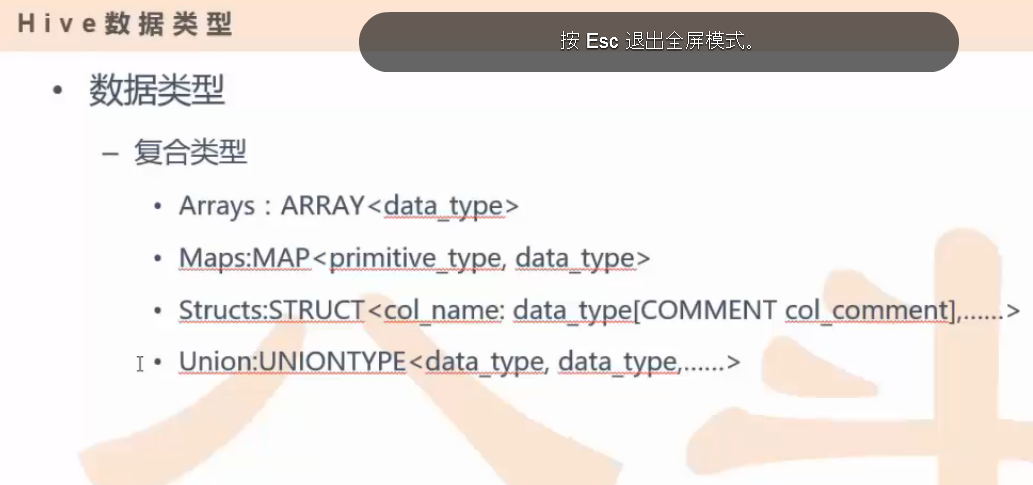
然后你去写hive的时候你这个每一个字段其实是有一个数据类型的，那hive的支持的数据类型也是非常丰富的（如下图8所示）

图8：



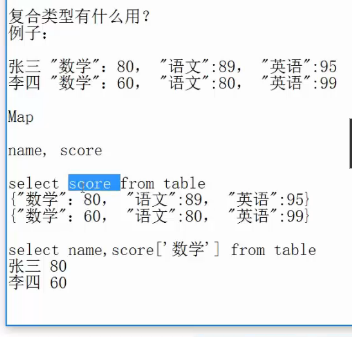
那它支持的类型有原生类型或者叫基础类型，还有一种叫复合类型（如下图9所示）

图9：



基础类型估计大家都明白对吧？都是int，float，double等等这些很基础的一些类型，那么复合类型是什么意思呢？就是一些个数组，map集合，结构体和联合体之类的，那主要的复合类型就包含那四种，那么大家肯定会有疑问了，我既然有原生类型了，我为什么还要复合类型呢？复合类型到底有什么用？假设我们举个例子，比如说张三这个人可能学习比较好，数学考了80分，语文考了89分，英文考了95分，然后李四就数学考了60分，语文考了80分，英语考了99分，好了这个时候你会发现不管你是数学语文还是英语其实他们都属于一个类型，那我是不是可以通过某一个类型把这里面的信息能够统一起来？所以我们看看能不能通过一个map集合的方式去封装起来，好了，如果说你这个字段已经封装起来了之后，比如这个字段包含了一个name和score这两个字段，其中score字段不再是我刚才讲的这些很基础的类型，这些一些复合类型，假设说我这个score已经是复合类型了，那你用的时候就很简单，比如你去select score from table 这个时候你其实查询就相当于把张三的语数英成绩和李四的语数英成绩给查出来了，但是我只想考虑每一个同学的数学考多少分，这个时候它的结果就变成什么呢？就变成张三 80，李四60，就变成这样的结果（如下图10所示）

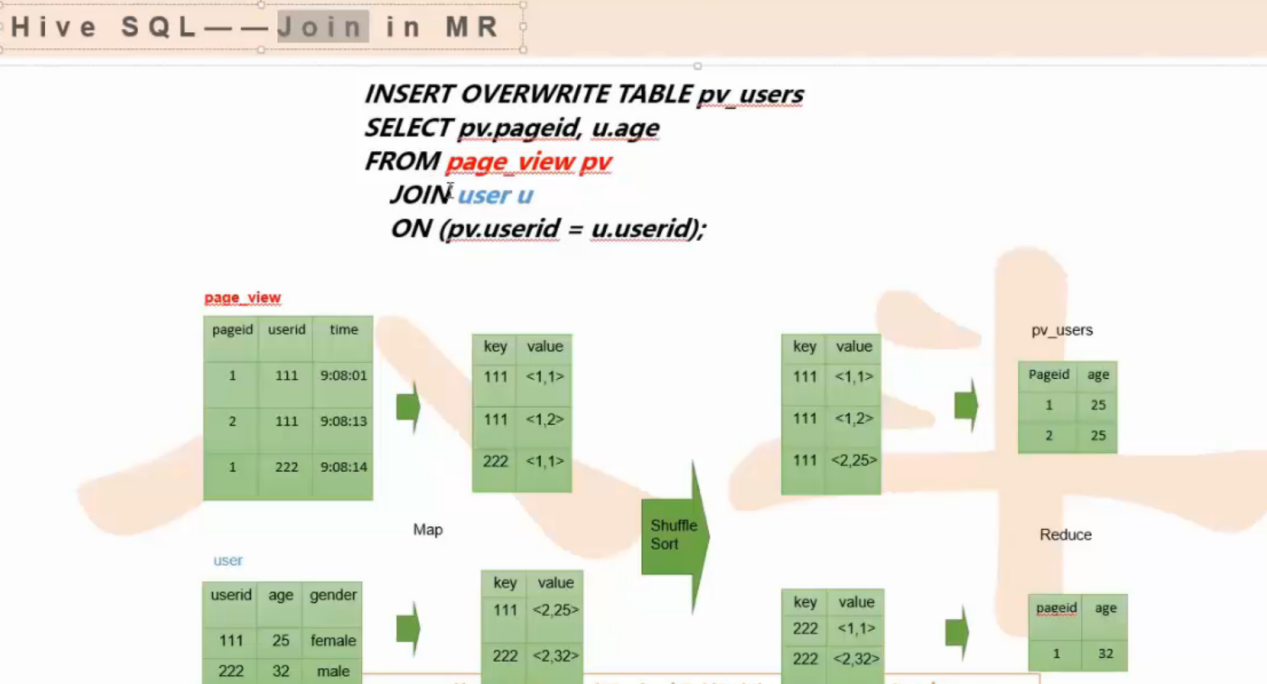
图10：



那么有什么好处呢？比如说我对语文和英语这两门课的成绩不关心，我只关心某一门课的一个成绩，我只关心把所有学员的名字打印出来并且每一个学员的数学分数是多少我可以通过select name ,score[‘数学’]这样的语句就很容易检索出来我需要的相应的数据了，这个就复合类型，当然我只通过了一个简单的map类型来表达，其他的类型大家可以自己研究一下。

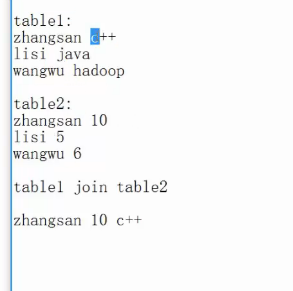
好了我们看一个实际的例子把（如下图11所示）

图11：



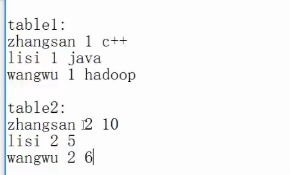
这个在我们之前写mapreduce的时候写过类似的例子对吧？其实这个例子就是一个join，就一个mapreduce的一个join，当时我们是怎么做join呢？还记得吗？就比如zhangsan是搞c++的，lisi是搞java的，wangwu是做hadoop的，那么zhangsan搞c++搞了十年，，lisi搞java搞了五年，wangwu搞hadoop搞了六年，现在我希望想得到什么样子的数据？就是把用户和工种这一个表和用户的工作年限的表，这两张表，我想把这两个表做一个join（如下图12所示）

图12：



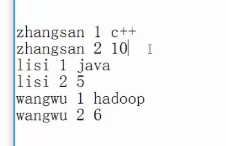
那其实我最终的目的就是举个例子，相当于我是想把这两张表做完join之后，我想把后面的字段本来是不在一个表里面的，现在可以变成一张表或者是一条记录进行输出，这个是怎么做的呢？当时我们是这样做的，在table1这张表里面打一个标记1并且在table2这张表里面打一个标记2（如下图13所示）

图13：



然后把这两个数据表作为数据统一输入到一个map里面去，然后通过它mapreduce一个内部的排序机制，然后这个时候你能保证你这个zhangsan这个排序完了以后会变成下图14所示

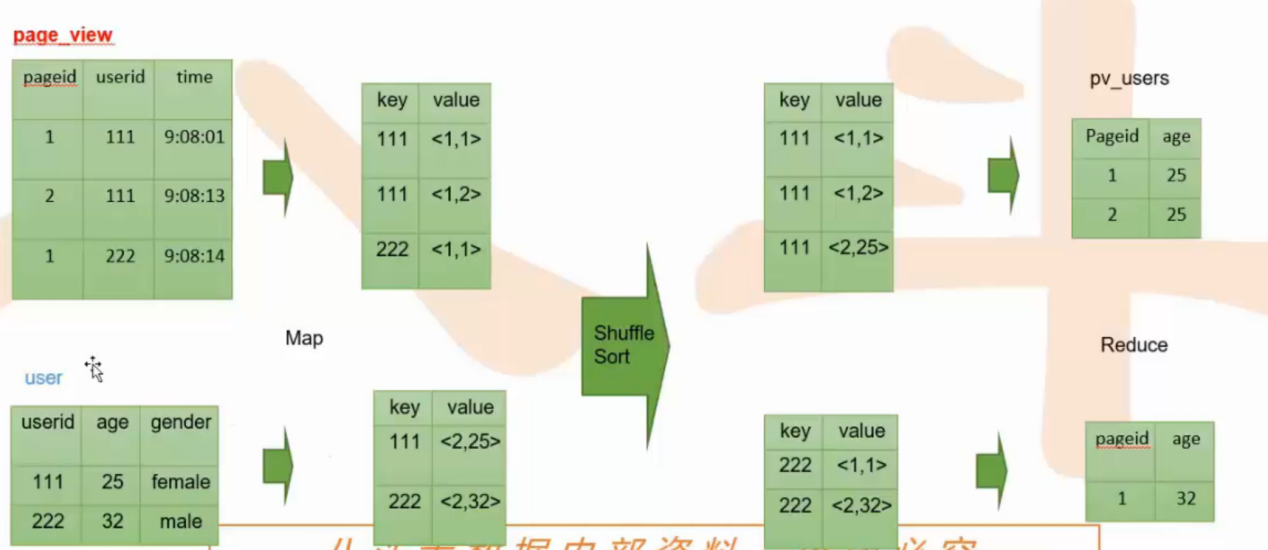
图14：



然后这个时候你就可以把它进行一个join进行处理了，然后把这个同样是zhangsan key的1和2后面的数据就可以拼到一块了，这个之前我们讲mapjon的逻辑，大家还记得把？相当于刚刚我领着大家复习了一遍。

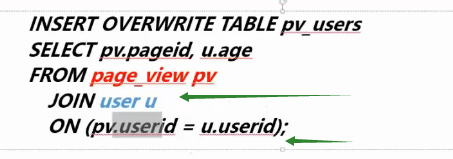
好了我们继续回头来看这张图（如下图15所示）

图15：



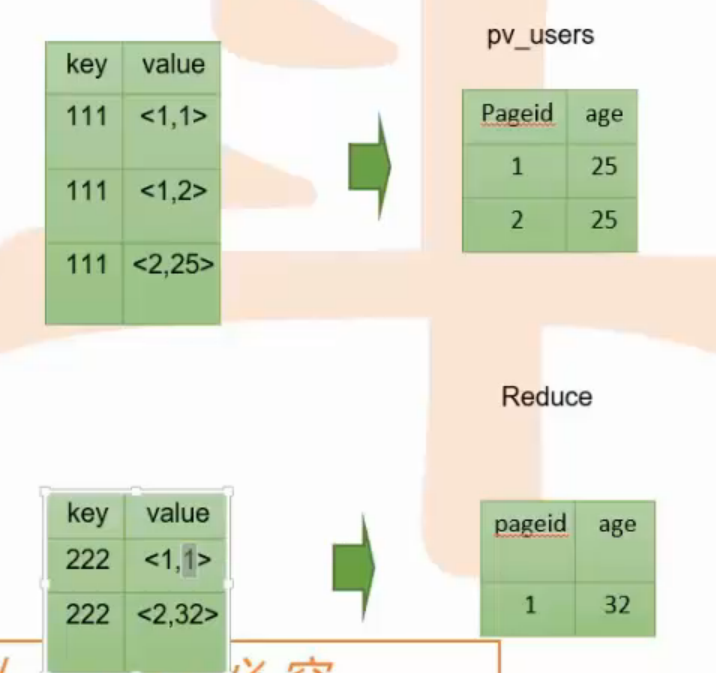
上图15有一张用户基础信息表，还有一张表是用户的行为表，用户基础信息表是有哪些信息呢？就代表这个用户userid，年龄age和性别gender，那么用户行为表是有用户userid它点击了哪一个网页，然后tiem就是在几点钟点击的，那么这个就是它的行为表，那我现在想把这两个表按照user为key进行一个join，那么怎么join呢？（如下图16所示）

图16：



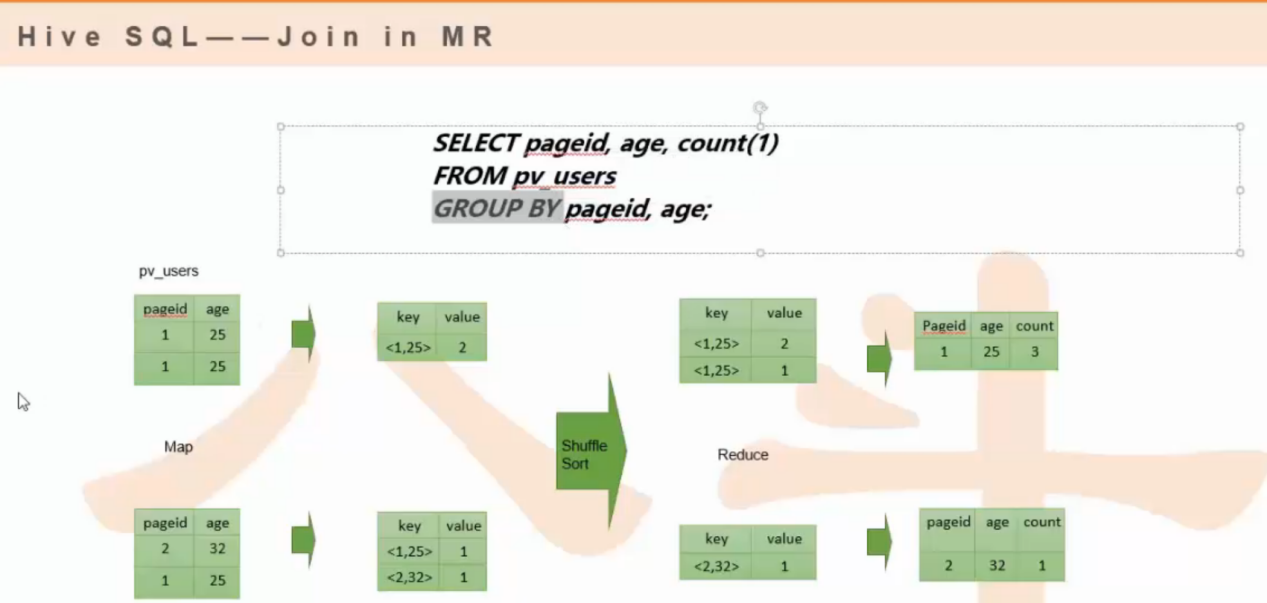
看上图16下面那里不是开始join吗？并且要求这个表的userid等于下面这个表的userid对不对？然后select它的pageid和它的age对不对？怎么做呢？相当于是把用户行为里面user用户变为key，那么value里面前面就是1，这个1就是我们刚刚打的标签嘛对不对？然后下面就是标记2，然后这两张表因为key都属于userid，然后把它作为整体的一个数据输入到你的mapreduce里面来，通过内部的一个shuffler和sort就可以把这两个key进行分开了对不对？分开完之后就从这个表里面分成了一个前面是1后面是25和2后面是25扩展出来，然后下面是1后面是32对不对？（如下图17所示）

图17：



所以这个流程大家有没有理解？这个之前有了mapreduce的join的基础，这个应该没有什么问题对吧？那么这个例子代表什么问题呢？说明是我们写一个hive这么一个语句其实内部我们完全可以通过mapreduce的方式来实现，我们再看下面一个例子（如下图18所示）

图18：



这个例子是做一个聚合，这怎么聚合的呢？这个例子也是两张表，两张表的属性都是pageid和age，然后我想对它做一个group by，好了这个时候把这个表按照这个<1.25>这个key去做一个转换，你看上面的group by后面有两个字段，一个pageid和一个age，就说明它把这两个字段作为一个key去做一个count，那么在这个表里面这个1，25这个为key，然后它value就是2，就出现两次一模一样的嘛对不对？

然后下面那张表就是个各出现一次，然后它排的这个顺序先把相同的key排到一块，然后这里面的2加1变成3，然后下面的那张就出现1次对不对？那么group by这个例子大家看懂了把？这个其实也是通过sql的语句来表达了一些实现。