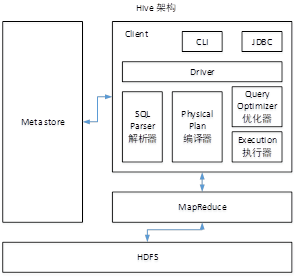
# Hive面试题

## 一、简述Hive主要架构及解析成MR的过程

Hive元数据默认存储在derby数据库，不支持多客户端访问，所以需要将元数据存储在MySQL中，才支持多客户端访问。主要架构如下：



Hive解析成MR的过程：

Hive通过给用户提供一系列交互接口，接收到用户的指令(sql语句)，结合元数据(metastore)，经过Driver内的解析器，编译器，优化器，执行器转换成mapreduce（将sql转换成抽象语法树AST的解析器，将AST编译成逻辑执行计划的编译器，在对逻辑执行计划进行优化的优化器，最后将逻辑执行计划转换成mapreduce），提交给hadoop中执行，最后将执行返回的结果输出到用户交互接口。

## 二、Hive与传统数据库的区别

Hive和数据库除了拥有类型的查询语言外，无其他相似

存储位置：Hive数据存储在HDFS上。数据库保存在块设备或本地文件系统

数据更新：Hive不建议对数据改写。数据库通常需要经常修改

执行引擎：Hive通过MapReduce来实现。数据库用自己的执行引擎

执行速度：Hive执行延迟高，但它数据规模远超过数据库处理能力时，Hive的并行计算能力就体现优势了。数据库执行延迟较低

数据规模：hive大规模的数据计算。数据库能支持的数据规模较小

扩展性：Hive建立在Hadoop上，随Hadoop的扩展性。数据库由于ACID语义的严格限制，扩展有限

## 三、Hive内部表和外部表的区别

存储：外部表数据由HDFS管理；内部表数据由hive自身管理

存储：外部表数据存储位置由自己指定（没有指定location则在默认地址下新建）；内部表数据存储在hive.metastore.warehouse.dir（默认在/uer/hive/warehouse）

创建：被external修饰的就是外部表；没被修饰是内部表

删除：删除外部表仅仅删除元数据；删除内部表会删除元数据和存储数据

## 四、Hive中order by，sort by，distribute by和cluster by的区别

order by：对数据进行全局排序，只有一个reduce工作。

sort by：每个mapreduce中进行排序，一般和distribute by使用，且distribute by写在sort by前面。当mapred.reduce.tasks=1时，效果和order by一样。

distribute by：类似MR的Partition，对key进行分区，结合sort by实现分区排序。

cluster by：当distribute by和sort by的字段相同时，可以使用cluster by代替，但cluster by只能是升序，不能指定排序规则。

在生产环境中order by使用的少，容易造成内存溢出(OOM)。

生产环境中distribute by和sort by用的多。

## 五、row\_number()，rank()和dense\_rank()的区别

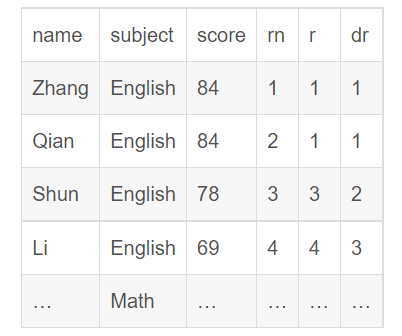
都有对数据进行排序的功能

row\_number()：根据查询结果的顺序计算排序，多用于分页查询

rank()：排序相同时序号重复，总序数不变

dense\_rank()：排序相同时序号重复时，总序数减少

|  |
| --- |
| select name,subject,score  row\_number() over(partition by subject order by score desc) rn,  rank() over(partition by subject order by score desc) r,  dense\_rank() over(partition by subject order by score desc) dr  from student\_score; |



## 六、Hive中常用的系统函数有哪些

date\_add(str,n)、date\_sub(str,n) 加减时间

next\_day(to\_date(str),’MO’) 周指标相关,获取str下周一日期

date\_format(str,’yyyy’) 根据格式整理日期

last\_day(to\_date(str)) 求当月最后一天日期

collect\_set(col) 收集数据返回一个以逗号分割的字符串数组

get\_json\_object(jsondata,’$.object’) 解析json，使用'$. object’获取对象值

NVL(str,replace) 空字段赋值，str为空返回replace值；两个都为空则返回null

## 七、Hive如何实现分区

建表：create table tablename(col1 string) partitioned by(col2 string);

添加分区：alter table tablename add partition(col2=’202101’);

删除分区：alter table tablename drop partition(col2=’202101’);

## 八、Hive导入数据的五种方式

1. Load方式，可以从本地或HDFS上导入，本地是copy，HDFS是移动

本地：load data local inpath ‘/root/student.txt’ into table student;

HDFS：load data inpath ‘/user/hive/data/student.txt’ into table student;

2. Insert方式，往表里插入

insert into table student values(1,’zhanshan’);

3. As select方式，根据查询结果创建表并插入数据

create table if not exists stu1 as select id,name from student;

4. Location方式，创建表并指定数据的路径

create external if not exists stu2 like student location '/user/hive/warehouse/student/student.txt';

5. Import方式，先从hive上使用export导出在导入

import table stu3 from ‘/user/export/student’;

## 九、Hive导出数据的五种方式

1. Insert方式，查询结果导出到本地或HDFS

Insert overwrite local directory ‘/root/insert/student’ select id,name from student;

Insert overwrite directory ‘/user/ insert /student’ select id,name from student;

2. Hadoop命令导出本地

hive>dfs -get /user/hive/warehouse/student/ 000000\_0 /root/hadoop/student.txt

3. hive Shell命令导出

]$ bin/hive -e ‘select id,name from student;’ > /root/hadoop/student.txt

4. Export导出到HDFS

hive> export table student to ‘/user/export/student’;

5. Sqoop导出

## 十、Hive的窗口函数有哪些

在SQL处理中，窗口函数都是最后一步执行，仅位于order by之前

over()：指定分析函数工作的数据窗口大小随行变化（跟在聚合函数 [wh2]后面，只对聚合函数有效）

current row 当前行

n preceding 往前n行数据

n following 往后n行数据

unbounded：

unbounded preceding 从前面开始 |————>

unbounded following 直到终点 ————>|

lag(col,n) 往前第n行数据

lead(col,n) 往后第n行数据

ntile(n) 把有序分区中的行分化到n个数据组中，各组的编号从1开始，ntile会返回每行所属的组编号（n为int类型）

举例说明：

|  |
| --- |
| select name,orderdate,cost,  sum(cost) over() as sp1, --所有行相加  sum(cost) over(partition by name) as sp2, --按名字分组，组内相加  sum(cost) over(partition by name order by orderdate) as sp3, --按名字分组并按时间排序  sum(cost) over(partition by name order by orderdate  rows between unbounded preceding and current row)as sp4,--由起点到当前行的聚合  sum(cost) over(partition by name order by orderdate  rows between 1 preceding and current row)as sp5, --由当前一行到当前行的聚合  sum(cost) over(partition by name order by orderdate  rows between 1 preceding and 1 following)as sp6, --由当前行到前后一行的聚合  sum(cost) over(partition by name order by orderdate  rows between current row and unbouded following)as sp7--由当前行到后面所有行的聚合  from business; |

\*注：sp3和sp4结果一样，因为根据order by升序逐渐依次增加

查看顾客上次和下次的购买时间

|  |
| --- |
| select name,orderdate,cost,  lag(orderdate,1) over(distribute by name sort by orderdate)as sp1,--上次购买时间  lead(orderdate,1) over(distribute by name sort by orderdate)as sp2--下次购买时间  from business; |

查询前20%时间的订单信息

|  |
| --- |
| select \* from (  select name,orderdate,cost, ntile(5) over(order by orderdate) sorted  from business  ) t  where sorted = 1; |

## 十一、怎么自定义UDF、UDAF、UDTF函数

### 1.自定义UDF函数

继承org.apache.hadoop.hive.ql.UDF函数；

重写evaluate方法，evaluate方法支持重载。

### 2.自定义UDAF函数

必须继承org.apache.hadoop.hive.ql.exec.UDAF(函数类继承)和org.apache.hadoop.hive.ql.exec.UDAFEvaluator(内部类Evaluator实现UDAFEvaluator接口)；

重写Evaluator方法时需要实现 init、iterate、terminatePartial、merge、terminate这几个函数：

init():类似于构造函数，用于UDAF的初始化

iterate():接收传入的参数，并进行内部的轮转，返回boolean

terminatePartial():无参数，其为iterate函数轮转结束后，返回轮转数据，类似于hadoop的Combiner

merge():接收terminatePartial的返回结果，进行数据merge操作，其返回类型为boolean

terminate():返回最终的聚集函数结果

### 3.自定义UDTF函数

继承org.apache.hadoop.hive.ql.udf.generic.GenericUDTF函数；

重写实现initialize, process, close三个方法。

UDTF首先会调用initialize方法，此方法返回UDTF的返回行的信息（返回个数，类型）。

初始化完成后，会调用process方法,真正的处理过程在process函数中，在process中，每一次forward()调用产生一行；如果产生多列可以将多个列的值放在一个数组中，然后将该数组传入到forward()函数。

最后close()方法调用，对需要清理的方法进行清理。

## 十二、ACID含义

A(atomicity)原子性：事务操作要么全部做完，要么不做

C(consistency)一致性：事务运行期间，数据对象依然完整性

I(isolation)独立性：并发事务之间不会相互影响

D(durability)持久性：一旦事务提交后，修改将永久保存在数据库上

## 十三、Hive表关联查询，如何解决数据倾斜的问题

### 1）倾斜原因：

map输出数据按key Hash的分配到reduce中，由于key分布不均匀、业务数据本身的特、建表时考虑不周、等原因造成的reduce 上的数据量差异过大。

（1）key分布不均匀;

（2）业务数据本身的特性;

（3）建表时考虑不周;

（4）某些SQL语句本身就有数据倾斜;

如何避免：对于key为空产生的数据倾斜，可以对其赋予一个随机值。

### 2）解决方案

（1）参数调节：

hive.map.aggr = true

hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定位true,生成的查询计划会有两个MR Job。第一个MR Job中，Map的输出结果集合会随机分布到Reduce中，每个Reduce做部分聚合操作，并输出结果，这样处理的结果是相同的Group By Key有可能被分发到不同的Reduce中，从而达到负载均衡的目的；第二个MR Job再根据预处理的数据结果按照Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个Reduce中），最后完成最终的聚合操作。

（2）SQL 语句调节：

① 选用join key分布最均匀的表作为驱动表。做好列裁剪和filter操作，以达到两表做join 的时候，数据量相对变小的效果。

② 大小表Join：

使用map join让小的维度表（1000 条以下的记录条数）先进内存（小表在左，大表在右）。在map端完成reduce.

③ 大表Join大表：

把空值的key变成一个字符串加上随机数，把倾斜的数据分到不同的reduce上，由于null 值关联不上，处理后并不影响最终结果。

④ count distinct大量相同特殊值:

count distinct 时，将值为空的情况单独处理，如果是计算count distinct，可以不用处理，直接过滤，在最后结果中加1。如果还有其他计算，需要进行group by，可以先将值为空的记录单独处理，再和其他计算结果进行union。

## 十四、谈一下hive的特点，以及hive和RDBMS有什么异同

hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供完整的sql查询功能，可以将sql语句转换成mapreduce任务进行运行。它的优点就是学习成本低，可以通过类sql语句快速实现简单的mapreduce统计，不必开发专门的mapreduce应用，十分适合数据仓库的统计分析，缺点：不支持实时数据查询。

Hive与关系型数据库的区别：



## 十五、简要描述数据库中的 null，说出null在hive底层如何存储，并解释select a.\* from t1 a left outer join t2 b on a.id=b.id where b.id is null; 语句的含义

null与任何值运算的结果都是null，可以使用is null、is not null函数指定其值为null情况下的取值。

null在hive底层默认是用'\N'来存储的，可以通过alter table test SET SERDEPROPERTIES('serialization.null.format' = 'a');来修改。

查询出t1表中与t2表中id相等的所有信息。

## 十六、写出hive中split、coalesce及collect\_list函数的用法（可举例）？

split将字符串转化为数组，即：split('a,b,c,d' , ',') ==> ["a","b","c","d"]。

coalesce(T v1, T v2, …) 返回参数中的第一个非空值；如果所有值都为NULL，那么返回NULL。

collect\_list列出该字段所有的值，不去重 select collect\_list(id) from table。

## 十七、hive有哪些方式保存元数据，各有哪些特点？

Hive支持三种不同的元存储服务器，分别为：内嵌式元存储服务器、本地元存储服务器、远程元存储服务器，每种存储方式使用不同的配置参数。

* 内嵌式元存储主要用于单元测试，在该模式下每次只有一个进程可以连接到元存储，Derby是内嵌式元存储的默认数据库。
* 在本地模式下，每个Hive客户端都会打开到数据存储的连接并在该连接上请求SQL查询。
* mysql在远程模式下，所有的Hive客户端都将打开一个到元数据服务器的连接，该服务器依次查询元数据，元数据服务器和客户端之间使用Thrift协议通信。

## 十八、写出将text.txt文件放入hive中test表‘2016-10-10’分区的语句，test的分区字段是l\_date

|  |
| --- |
| load data local inpath '/your/path/test.txt' overwrite into table test partition(1\_date='2016-10-10') |

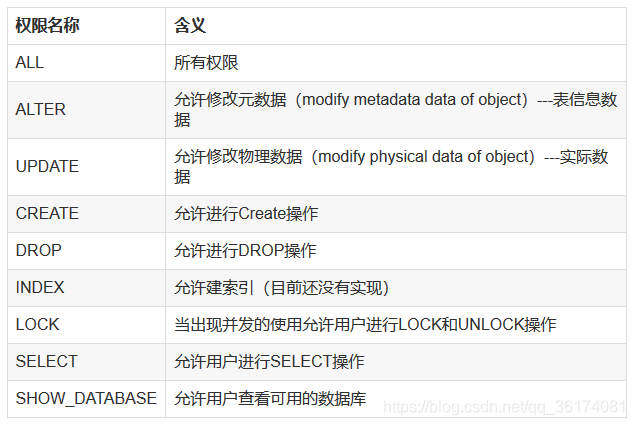
## 十九、Hive如何进行权限控制

目前hive支持简单的权限管理，默认情况下是不开启，这样所有的用户都具有相同的权限，同时也是超级管理员，也就对hive中的所有表都有查看和改动的权利，这样是不符合一般数据仓库的安全原则的。Hive可以是基于元数据的权限管理，也可以基于文件存储级别的权限管理。

为了使用Hive的授权机制，有两个参数必须在hive-site.xml中设置：

|  |
| --- |
| <property>  <name>hive.security.authorization.enabled</name>  <value>true</value>  </property>  <property>  <name>hive.security.authorization.createtable.owner.grants</name>  <value>ALL</value>  </property> |
|  |

hive支持的权限



## 二十、Hive 中的压缩格式TextFile、SequenceFile、RCfile 、ORCfile各有什么区别？

* **TextFile**：默认格式，存储方式为行存储，数据不做压缩，磁盘开销大，数据解析开销大。可结合Gzip、Bzip2使用（系统自动检查，执行查询时自动解压），但使用这种方式，压缩后的文件不支持split，Hive不会对数据进行切分，从而无法对数据进行并行操作。并且在反序列化过程中，必须逐个字符判断是不是分隔符和行结束符，因此反序列化开销会比SequenceFile高几十倍。
* **SequenceFile**：SequenceFile是Hadoop API提供的一种二进制文件支持，存储方式为行存储，其具有使用方便、可分割、可压缩的特点。SequenceFile支持三种压缩选择：NONE，RECORD，BLOCK。Record压缩率低，一般建议使用BLOCK压缩。优势是文件和hadoop api中的MapFile是相互兼容的。
* **RCfile**：存储方式：数据按行分块，每块按列存储。结合了行存储和列存储的优点：首先，RCFile 保证同一行的数据位于同一节点，因此元组重构的开销很低；其次，像列存储一样，RCFile能够利用列维度的数据压缩，并且能跳过不必要的列读取；
* **ORCfile**：存储方式：数据按行分块 每块按照列存储。压缩快 快速列存取。效率比rcfile高,是rcfile的改良版本。

总结：相比TEXTFILE和SEQUENCEFILE，RCFILE由于列式存储方式，数据加载时性能消耗较大，但是具有较好的压缩比和查询响应。数据仓库的特点是一次写入、多次读取，因此，整体来看，RCFILE相比其余两种格式具有较明显的优势 。

## 二十一、Hive join过程中大表小表的放置顺序

在编写带有join操作的代码语句时，应该将条目少的表/子查询放在join操作符的左边。因为在reduce阶段，位于join操作符左边的表的内容会被加载进内存，载入条目较少的表可以有效减少oom（out of memory）即内存溢出。所以对于同一个key来说，对应的value值小的放前，大的放后。------小表放前（小表join大表）。

## 二十二、Hive的两张表关联，使用mapreduce怎么实现？

1）如果其中有一张表为小表，直接使用map端join的方式（map端加载小表）进行聚合。

2）如果两张都是大表，那么采用联合key，联合key的第一个组成部分是join on中的公共字段，第二部分是一个flag，0代表表A，1代表表B，由此让Reduce区分客户信息和订单信息；在Mapper中同时处理两张表的信息，将join on公共字段相同的数据划分到同一个分区中，进而传递到一个Reduce中，然后在Reduce中实现聚合。

## 二十三、介绍一下有哪些常用函数？

### 1．行转列函数

CONCAT(string A/col, string B/col…)：返回输入字符串连接后的结果，支持任意个输入字符串。

例如： concat( aa, ‘:’, bb) 就相当于把aa列和bb列用冒号连接起来了，aa:bb。

CONCAT\_WS(separator, str1, str2,…)：CONCAT\_WS() 代表 CONCAT With Separator ，是CONCAT()的特殊形式。第一个参数是其它参数的分隔符。分隔符的位置放在要连接的两个字符串之间。分隔符可以是一个字符串，也可以是其它参数。如果分隔符为 NULL，则结果为 NULL。函数会忽略任何分隔符参数后的 NULL 值。但是CONCAT\_WS()不会忽略任何空字符串。 (然而会忽略所有的 NULL）。

COLLECT\_SET(col)：函数只接受基本数据类型，它的主要作用是将某字段的值进行去重汇总，产生array类型字段。

### 2．列转行函数

EXPLODE(col)：将hive某列中复杂的array或者map结构拆分成多行。

LATERAL VIEW：常和UDTF函数一起使用。

用法：LATERAL VIEW udtf(expression) tableAlias AS columnAlias

解释：用于和split, explode等UDTF一起使用，它能够将一列数据拆成多行数据，在此基础上可以对拆分后的数据进行聚合。

### 3．Rank排名函数

RANK() 排序相同时会重复，总数不会变；

DENSE\_RANK() 排序相同时会重复，总数会减少；

ROW\_NUMBER() 根据顺序计算排名。

在实际开发中，以上三个rank函数通常是和开窗函数一起使用的。

### 4．窗口函数（开窗函数）

OVER()：用于指定分析函数工作时的数据窗口大小，这个数据窗口大小可能会随着行的变而变化；

CURRENT ROW：当前行；

n PRECEDING：往前n行数据；

n FOLLOWING：往后n行数据；

UNBOUNDED：起点，UNBOUNDED PRECEDING 表示从前面的起点， UNBOUNDED FOLLOWING表示到后面的终点；

LAG(col,n,default\_val)：往前第n行数据；

LEAD(col,n, default\_val)：往后第n行数据；

NTILE(n)：把有序分区中的行分发到指定数据的组中，各个组有编号，编号从1开始，对于每一行，NTILE返回此行所属的组的编号。这个函数需要注意：n必须为int类型。

## 二十四、Hive优化相关面试题

### 1.MapJoin

如果不指定 MapJoin 或者不符合 MapJoin 的条件，那么 Hive 解析器会将 Join 操作转换 成 Common Join，即：在 Reduce 阶段完成 join。容易发生数据倾斜。可以用MapJoin把小 表全部加载到内存在 map 端进行 join，避免 reducer 处理。

### 2.行列过滤

列处理：在 SELECT 中，只拿需要的列，如果有，尽量使用分区过滤，少用 SELECT \*。

行处理：在分区剪裁中，当使用外关联时，如果将副表的过滤条件写在 Where 后面，那 么就会先全表关联，之后再过滤。

### 3.多采用分桶技术

### 4.结合实际环境合理设置 Map 数

通常情况下，作业会通过 input的目录产生一个或者多个map任务。 主要的决定因素有：input的文件总个数，input的文件大小，集群设置的文件块大小；

map数不是越多越好；如果一个任务有很多小文件（远远小于块大小 128m），则每个小文件 也会被当做一个块，用一个 map 任务来完成，而一个 map 任务启动和初始化的时间远远大 于逻辑处理的时间，就会造成很大的资源浪费。而且，同时可执行的 map 数是受限的。解决这个问题需要减少map数。

并不是每个map处理接近128m的文件块就是完美的；比如有一个 127m 的文件，正常会用一个 map 去完成，但这个文件只 有一个或者两个小字段，却有几千万的记录，如果 map 处理的逻辑比较复杂，用一个 map 任务去做，肯定也比较耗时。解决这个问题需要增加map数。

### 5.合并大量小文件

在Map执行前合并小文件，可以减少Map数：CombineHiveInputFormat 具有对小文件进行合并的功能（系统默认的格式）。HiveInputFormat 没有对小文件合并功能。

### 6.设置合理的Reduce数

Reduce 个数也并不是越多越好

过多的启动和初始化 Reduce 也会消耗时间和资源；

有多少个 Reduce，就会有多少个输出文件，如果生成了很多个小文件，那么如果这些小文件作为下一个任务的输入，则也会出现小文件过多的问题；

在设置Reduce个数的时候也需要考虑这两个原则：处理大数据量利用合适的 Reduce 数；使单个 Reduce 任务处理数据量大小要合适；

### 7.输出合并小文件常用参数

|  |
| --- |
| SET hive.merge.mapfiles = true; -- 默认 true，在 map-only 任务结束时合并小文件  SET hive.merge.mapredfiles = true; -- 默认 false，在 map-reduce 任务结束时合并小文件  SET hive.merge.size.per.task = 268435456; -- 默认 256M  SET hive.merge.smallfiles.avgsize = 16777216; -- 当输出文件的平均大小小于 16m 该值时，启动一个独立的 map-reduce 任务进行文件 merge |

### 8.开启 map 端 combiner（不影响最终业务逻辑）

开启命令：

|  |
| --- |
| set hive.map.aggr=true； |

### 9.中间结果压缩

设置 map 端输出、中间结果压缩。（不完全是解决数据倾斜的问题，但是减少了 IO 读写和网络传输，能提高很多效率）

## 二十五、扩展面试题

<https://www.jianshu.com/p/b0b77b045fab>

## 二十六、Hive静态分区与动态分区的区别

hive中支持两种类型的分区：

静态分区SP（static partition）

动态分区DP（dynamic partition）

静态分区与动态分区的主要区别在于静态分区是手动指定，而动态分区是通过数据来进行判断。详细来说，静态分区的列实在编译时期，通过用户传递来决定的；动态分区只有在SQL执行时才能决定。

## 二十七、海量数据分布在100台电脑中，想个办法高效统计出这批数据的TOP10

方案1:

a) 在每台电脑上求出TOP10，可以采用包含10个元素的堆完成(TOP10小，用最大堆，TOP10大，用最小堆)。

b) 比如求TOP10大，我们首先取前10个元素调整成最小堆，如果发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆顶，然后再调整为最小堆。

c) 最后堆中的元素就是TOP10大

方案2

a) 求出每台电脑上的TOP10后，然后把这100台电脑上的TOP10组合起来，共1000个数据

b) 再利用上面类似的方法求出TOP10就可以了。

## 二十八、Hive的特点

本题主要为了考察对hive的整体使用场景的掌握程度，毕竟只有知道了hive的特点，才能有针对性的在实际项目中的合适场景下使用hive。可以从下面四个角度去分析：

### 1.数据存储位置

Hive的数据存储在hdfs上，元数据可以存储在指定的地方比如mysql，PostgreSQL等。

### 2.数据更新

Hive处理数据时一般不对数据进行改写，因为它不支持行级别的增删操作，如果要进行更新数据，一般可以通过分区或者表直接覆盖。

### 3.执行效率

Hive 执行延迟较高。虽然在小数据量时传统数据库延迟更低，但是当数据规模大到超过传统数据库的处理能力的时候，Hive 的并行计算显然能体现出优势。

### 4.数据规模

Hive 支持大规模的数据计算，通常是PB级别的数据。