1你平时使用过哪些关系型数据库?

```
sql \ orcal
```

什么是事务?

```
事务是数据库最小逻辑单位，是一个或者一组sql的集合
```

2、 事务的ACID特性?

```
A=Atomicity 原子性：就是上面说的,事务是数据库最小逻辑单位。

C=Consistency 一致性：系统总是从一个一致性的状态转移到另一个一致性的状态。

I=Isolation 隔离性：通常来说:一个事务在完全提交之前,对其他事务是不可见的.注意前面的通常来说加了红色,意味着有例外情况。

D=Durability 持久性：事务一旦提交，将永久存在。
```

3mysql怎么优化?

```
1 尽量不使用 select *  指明字段查询
2 用in代替or 连续数据用between and
3 查询数据时 避免随机查询
4 查询数据时用 limit 减少查询多余记录
5 insert into 时尽量批量插入
6 group by 如果对group by语句的结果没有排序要求，在语句后面加 order by null 取消排序
```

5shell脚本熟不熟悉，有写过哪些?

```
zookpeer hdfs yarn 等的启动脚本
```

6不规则字符串日期怎么转成想要的日期格式?

```
date_format(date,'%Y-%m-%d') ———->oracle中的to_char();
str_to_date(date,'%Y-%m-%d') ———->oracle中的to_date();
```

8.CDH熟不熟悉?用过哪些?

```
CDH 5       Cloudera Manager CDH的集群管理工具
端口 7180
```

9.oracle和mysql的区别

```
1、Oracle是大型数据库，而MySQL是中小型数据库。但是MySQL是开源的，但是Oracle是收费的，而且比较贵。

2、Oracle的内存占有量非常大，而mysql非常小

3、MySQL支持主键自增长，指定主键为auto increment，插入时会自动增长。Oracle主键一般使用序列。

4、MySQL字符串可以使用双引号包起来，而Oracle只可以单引号

5、MySQL分页用limit关键字，而Oracle使用rownum字段表明位置，而且只能使用小于，不能使用大于。
```

6、Oracle在处理长字符串的时候,长度是小于等于4000个字节,如果要插入更长的字符串,考虑用CLOB类型,插入修改记录前要做进行修改和 长度的判断,如果为空,如果长度超出返回操作处理.（CLOB类型是内置类型，它一般都作为某一行中的一列,有些数据库也有别名）

7、MySQL中0、1判断真假，Oracle中true false

8、MySQL中命令默认commit,但是Oracle需要手动提交

9、MySQL在windows环境下大小写不敏感 在unix,linux环境下区分大小写，Oracle不区分

10. mysql索引 ,你们用过什么?

普通索引：
MySQL中基本索引类型，没有什么限制，允许在定义索引的列中插入重复值和空值，纯粹为了查询数据更快一点。
1.1.2、唯一索引：
索引列中的值必须是唯一的，但是允许为空值，
1.1.3、主键索引：
是一种特殊的唯一索引，不允许有空值。

11.mysql和hivesql的区别?

1、存储位置：Hive在Hadoop上；Mysql将数据存储在设备或本地系统中；

2、数据更新：Hive不支持数据的改写和添加，是在加载的时候就已经确定好了；数据库可以CRUD；

3、索引：Hive无索引，每次扫描所有数据，底层是MR，并行计算，适用于大数据量；MySQL有索引，适合在线查询数据；

4、执行：Hive底层是MarReduce；MySQL底层是执行引擎；

5、可扩展性：Hive：大数据量；MySQL:相对就很少了。

6 hive是OLAP数据仓库工具 mysql是OLTP数据库

13说一说你们常用分析指标?

常见
- UV、PV、IP、Session、平均访问次数、平均访问时长、跳出率、二跳率

常见维度

常见
- 时间维度：年、季度、月、周、天、小时
- 地域维度：国家、省份、城市
- 平台维度
- 用户维度
- 操作系统维度

14业务表关联字段你们是怎么获取的?(有人整理还是自己慢慢整理的)，获取依据是什么?

1.having的作用?

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，并提供简单的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。减少开发成本

2.left join和join的区别?

left join是以左表为准的.右表不符合的以null填充,
join 不以哪边为基准,只查询出符合条件的记录

3.谓词下推?(先过虑,后join)

谓词下推的概念其实出现在sql中，在关联查询时(join，left join ,right join),因为涉及两个大表之间的关联(特别是在hive)造成资源消耗会比较大，因为建议在join之前先将两个表进行过滤(hive 里指的是在map端进行过滤)，系统会进行部分优化，但sql需要遵守PPD规则，所谓下推可以理解成优化（只有满足才能进行优化）。
一句话说完：不影响结果的情况下，尽量将过滤条件提前执行。（记得小表join大表）

4拉链表是什么，怎么构建，项目中有没有用到拉链表?

什么是拉链表
　　当维度数据发生变化时，将旧数据置为失效，将更改后的数据当作新的记录插入到维度表中，并开始生效，这样能够记录数据在某种粒度上的变化历史。

5.linux上如何创建目录?

mkdir

6.行列转换?

union　case when

Hive和hadoop的关系

Hive利用HDFS存储数据，利用MapReduce查询分析数据。

Hive4种排序

order by //可以指定desc 降序 asc 升序
order by会对输入做全局排序，因此只有一个Reducer(多个Reducer无法保证全局有序)，然而只有一个Reducer，会导致当输入规模较大时，消耗较长的计算时间。

sort by 【对分区内的数据进行排序】
sort by不是全局排序，其在数据进入reducer前完成排序，因此，如果用sort by进行排序，并且设置mapred.reduce.tasks>1，则sort by只会保证每个reducer的输出有序，并不保证全局有序。sort by不同于order by，它不受Hive.mapred.mode属性的影响，sort by的数据只能保证在同一个reduce中的数据可以按指定字段排序。使用sort by你可以指定执行的reduce个数(通过set mapred.reduce.tasks=n来指定)，对输出的数据再执行归并排序，即可得到全部结果。

distribute by 【对map输出进行分区】
distribute by是控制在map端如何拆分数据给reduce端的。hive会根据distribute by后面列，对应reduce的个数进行分发，默认是采用hash算法。sort by为每个reduce产生一个排序文件。在有些情况下，你需要控制某个特定行应该到哪个reducer，这通常是为了进行后续的聚集操作。distribute by刚好可以做这件事。因此，distribute by经常和sort by配合使用。

cluster by
cluster by除了具有distribute by的功能外还兼具sort by的功能。当distribute by和sort by 是同一个字段的时候可以使用cluster by替代。但是排序只能是倒叙排序，不能指定排序规则为ASC或者DESC。

9.常见的逻辑模型?

1.层次模型：
优点：
1.结构清晰，便于观看实体间的联系
2.操作简单
3.查询效率高
缺点：
1.结构灵活性低，当需要更新或修改一个实体时，会影响到其他的数据
2.加大了DBMS的管理负担

2.网状模型：
优点：
1.允许单个节点存在多于一个父亲节点
2.可以存在俩个或多个节点没有父节点
3.真实反映现实世界
缺点：
1.联系复杂，难以实现，数据库维护重建难度大

3.关系模型：
优点：
1.结构简单明了
2.独立性比较强
3.操作方便
4.有坚实的数学理论做基础
缺点：
与层次模型比查询效率低，加大了系统的查询负担

2.项目中hive的优化。

1.在 SELECT 中，只拿需要的列，如果有，尽量使用分区过滤，少用 SELECT *

2.多采用分桶技术

3.合并大量小文件

在Map执行前合并小文件，可以减少Map数

4.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；

5.JVM重用：一个job可能有多个map reduce任务，每个任务会开启一个JVM虚拟机，默认情况下一个任务对应一个JVM，任务运行完JVM即销毁，我们可以设置JVM重用参数，一般不超过5个，这样一个JVM内可以连续运行多个任务

4窗口函数有哪些，作用，有用过ntile(自动分组)吗

```
rank  over  denserank rownumber  聚合函数等
```

SecondaryNameNode

```
SecondaryNameNode合并 NameNode 的 edits  到 fsimage 文件中
```

Hadoop读写流程

客户端向namenode发送请求上传文件,namenode对客户端的请求进行权限验证和对客户端请求上传的文件是否存在进行验证,如果没有权限则直接报错,如果文件已存在直接报错,如果符合条件通知客户端可以上传,客户端对需要上传的文件进行切片,客户端重新向namenode发送请求询问文件上传到哪个节点,namenode接收到客户端请求后根据副本机制,负载均衡,机架感知及网络拓扑图找到存储第一个数据块的datanode列表,返回给客户端,客户端根据收到的datanode列表,链接就近的机器,例如node1,然后依次和datanode列表中的机器建立链接形成传输管道,采用数据包的方式传输数据,建立ack确认机制,第一个数据包传输完成后,向namenode请求第二个数据包上传的位置,并接受一个新的datanode列表,然后继续往下执行,直至所有数据块上传完毕.

5.MR的执行流程。

一个文件分成多个split数据片。
每个split由一个map进行处理。
Map处理完一个数据就把处理结果放到一个环形缓冲区内存中。
环形缓冲区满后里面的数据会被溢写到一个个小文件中。
小文件会被合并成一个大文件，大文件会按照分区进行排序。
reduce节点将所有属于自己的数据从分区中拷贝到自己的缓冲区中，并进行合并。
最后合并后的数据交给reduce处理程序进行处理。
处理后的结果存放到HDFS上。

YARN执行流程

client客户端向yarn集群(resourcemanager)提交任务

- resourcemanager选择一个node创建appmaster
- appmaster根据任务向rm申请资源
- rm返回资源申请的结果
- appmaster去对应的node上创建任务需要的资源（container形式，包括内存和CPU）
- appmaster负责与nodemanager进行沟通，监控任务运行
- 最后任务运行成功，汇总结果。

6.shuffle优化如何做的。

对数据进行压缩,减少读取的数据量

减少不必要的排序

尽量将shuffle数据放到内存上

7.Hadoop、Hive和Spark的关系。

hadoop：hdfs,MapReduce,yarn，MapReduce负责计算数据，hdfs负责存储数据。yarn负责资源分配。

hive：使用HiveQL语句,将其转化成MapReduce任务，可以分布式运行。

spark:计算存储在hadoop集群上的数据。计算速度比MapReduce快很多，可以进行实时的应用。

什么是数据倾斜

mapreduce程序执行时，reduce节点大部分执行完毕，但是有一个或者几个reduce节点运行很慢，导致整个程序的处理时间很长，这是因为某一个key的条数比其他key多很多（有时是百倍或者千倍之多），这条key所在的reduce节点所处理的数据量比其他节点就大很多，从而导致某几个节点迟迟运行不完。

10.数据倾斜怎么处理

-增加并行度

-join倾斜
在进行两个表join的过程中，由于hive都是从左向右执行，要注意讲小表在前，大表在后（小表会先进行缓存）

-空值倾斜(丢失的日志)
1 过滤空值,使空值不进行关联
2 给空值做统一的赋值后进行关联

-不同数据类型关联产生数据倾斜
统一数据类型

12.zookeeper如何保证数据一致性

多个操作组成一个事务，要么一起成功，要么一起失败，不会存在中间的状态。如果中间失败了要进行回滚操作。主从架构,只有leader能进行事务性操作,follower和Observer处理非事务性操作　转发事务性操作给leader,当leader挂掉后从follower中选举出新的leader

在工作中主要做集群管理

为什么要分层

1. 清晰数据结构：每一个数据分层都有它的作用域和职责，在使用表的时候能更方便地定位和理解
2. 减少重复开发：规范数据分层，开发一些通用的中间层数据，能够减少极大的重复计算
3. 便于维护：当数据出现问题之后，可以不用修复所有的数据，只需要从有问题的步骤开始修复。
4. 统一数据口径：通过数据分层，提供统一的数据出口，统一对外输出的数据口径
5. 复杂问题简单化：将一个复杂的任务分解成多个步骤来完成，每一层解决特定的问题

分层

源数据层（ODS）

此层数据无任何更改，直接沿用外围系统数据结构和数据，不对外开放；为临时存储层，是接口数据的临时存储区域，为后一步的数据处理做准备。

数据仓库层（DW）

DW 层的数据应该是一致的、准确的、干净的数据，即对源系统数据进行了清洗（去除了杂质）后的数据。

此层可以细分为三层：

明细层DWD（Data Warehouse Detail）：存储明细数据，此数据是最细粒度的事实数据。该层一般保持和ODS层一样的数据粒度，并且提供一定的数据质量保证。同时，为了提高数据明细层的易用性，该层会采用一些维度退化手法，将维度退化至事实表中，减少事实表和维表的关联。

中间层DWM（Data WareHouse Middle）：存储中间数据，为数据统计需要创建的中间表数据，此数据一般是对多个维度的聚合数据，此层数据通常来源于DWD层的数据。

业务层DWS（Data WareHouse Service）：存储宽表数据，此层数据是针对某个业务领域的聚合数据，应用层的数据通常来源与此层，为什么叫宽表，主要是为了应用层的需要在这一层将业务相关的所有数据统一汇集起来进行存储，方便业务层获取。此层数据通常来源与DWD和DWM层的数据。

在实际计算中，如果直接从DWD或者ODS计算出宽表的统计指标，会存在计算量太大并且维度太少的问题，因此一般的做法是，在DWM层先计算出多个小的中间表，然后再拼接成一张DWS的宽表。由于宽和窄的界限不易界定，也可以去掉DWM这一层，只留DWS层，将所有的数据在放在DWS亦可。

数据应用层（DA 或 APP）

前端应用直接读取的数据源；根据报表、专题分析的需求而计算生成的数据。

在线教育数仓建设案例

7. 数据仓库设计案例

这里我们以电商网站的数据仓库为例，针对用户访问日志这一部分数据进行举例说明。

在ODS层中，由于各端的开发团队不同或者各种其它问题，用户的访问日志被分成了好几张表上报到了我们的ODS层。

为了方便大家的使用，我们在DWD层做了一张用户访问行为天表，在这里，我们将PC网页、H5、小程序和原生APP访问日志汇聚到一张表里面，统一字段名，提升数据质量，这样就有了一张可供大家方便使用的明细表了。

在DWM层，我们会从DWD层中选取业务关注的核心维度来做聚合操作，比如只保留人、商品、设备和页面区域维度。类似的，我们这样做了很多个DWM的中间表。

然后在DWS层，我们将一个人在整个网站中的行为数据放到一张表中，这就是我们的宽表了，有了这张表，就可以快速满足大部分的通用型业务需求了。

最后，在APP应用层，根据需求从DWS层的一张或者多张表取出数据拼接成一张应用表即可。

1.1 OLTP和OLAP区别

OLTP 面向事务性操作,注重数据安全\完整\响应效率 这类系统的特点是事务操作频繁，数据量小。RDBMS型数据库 mysql oracle

OLAP支持复杂的分析操作，侧重决策支持，这类系统的特点是没有事务性操作，主要是查询操作，数据量大。数据仓库 hive impala

线上教育技术选型

集群管理工具CDH Cloudera Manager

ETL工具 Sqoop专为关系型数据库和Hadoop之间的ETL而生，支持海量数据，符合项目的需求，且操作简单门槛低，因此选择Sqoop作为ETL工具。

调度工具 Ooize

- **目标**:

- 实现用户的转化运营分析，提高转化率，实现用户的学习管理分析，提高学习效率

项目需求

- 提高报名率
  - **需求1：统计不同维度下的访问转咨询转换率**
    - 为什么这个人访问了，但不咨询?
  - **需求2：统计不同维度下的意向转报名转换率**
    - 为什么明确了意向，但不报名?
- 保证课程质量和学员学习效果
  - 学习效果
  - **需求3：考勤分析：统计不同维度下学员以及班级的出勤情况**
    - 出勤率
    - 迟到率
    - 请假率
    - 旷课率

项目看板模块

- 访问分析主题看板
- 咨询分析主题看板
- 意向分析主题看板
- 报名分析主题看板
- 考勤分析主题看板

数据来源

- **业务数据**：用于支撑业务平台而实现的业务存储
  - 用户信息、订单信息、商品信息
  - 存储：数据库
  - 应用：业务分析
- **用户行为数据**：用于记录用户在平台上所有的操作行为
  - 事件：点击、注册、登陆、浏览、支付、收藏、搜索……
  - 存储：文件
  - 应用：用户画像
- 爬虫数据：爬虫组根据实际的数据需求，爬取其他网络平台的数据

- 运维日志：为了构建自动化运维平台，分析处理所有机器和程序的运营日志，提供给运维做监控

压缩工具使用

系统采用

因为ORCFILE的压缩快、存取快，而且拥有特有的查询优化机制，所以系统采用ORCFILE存储格式（RCFILE升级版），压缩算法采用orc支持的ZLIB和SNAPPY。

在ODS（DWD）数据源层，因为数据量较大，可以采用orcfile+ZLIB的方式，以节省磁盘空间；

而在计算的过程中（DWD、DWM、DWS、APP），为了不影响执行的速度，可以浪费一点磁盘空间，采用orcfile+SNAPPY的方式，提升hive的执行速度。DWD明细层根据业务的需要，灵活选用ZLIB或SNAPPY。