面试问题总结：

#  自我介绍

我之前公司是**深圳市深讯数据科技股份有限公司**，是一家数据服务外包公司，主要是给银行做项目的，我在里面做BI开发工程师

最近一个项目是广东南粤银行驾驶舱项目项目，是一个面向销售交易、财务、风险监控等分析主题的数据仓储项目。

我们会利用ETL工具把客户业务系统数据抽取到Oracle数据仓库的ods源数据层中,

再利用存储过程把数据同步到DW层，并进行事实表和维度表的区分，

后面根据业务部门提出的需求来建立数据模型，也就是DM表，比如交易相关指标分析，数据模型建好之后再进行可视化报表的制作，

我的主要工作就是写存储过程来完成需求的统计工作，并把数据放到DM的目标表里，把数据提供给前端开发。

# 介绍最近的项目

（主要工作职责sp，用到什么工具plsql/finereport/Oracle/kettle等，怎么用，工作流程：结合工作流程说明工具如何使用）

见剧本话术

项目团队的介绍：需求分析员1位，表结构设计mapping设计人员2名，前台开发2名，数据库开发5名，测试人员2名。

项目流程：

1. 首先是需求阶段，我们组的需求分析会把客户的需求分析好，包括要出什么报表，指标、维度、取数逻辑、范围什么的都会分析好。

客户提一个需求就是衡量资产质量的需求，然后我们就开始对这个需求进行分析，如需要那些指标（不良贷款率，贷款拨备率，拨备覆盖率等），然后就是分析要从那些角度去看这些指标（维度，区域，信用等级，客户性别年龄等），我的取数范围是怎么样的，用到那些表，需要那些字段（信用卡交易明细，信用卡客户信息拓展，信用卡账户信息表等取信用卡号码，账户id，贷款金额类的情况，是否逾期逾期时长等之类的字段）

1. 然后是设计阶段，开发组长会设计需要用到的表，数据仓库表或者模型表，会把数据字典和mapping设计好，然后分配开发任务给我们，比如谁开发哪些表。

需求中涉及到的业务主体概念提炼出来，如在资产质量这个模块中我已提炼出这几个业务主体：客户，公司，机构，产品，贷款，担保物等。可以把这几个主体进行组合从而组建不同的主题域（贷款主题域，借记卡交易主题域，信用等级划分主题域等）

维度表（客户方面，机构，产品，账户之类的），事实表（交易信息表，存款信息表，信用卡开户信息等，贷款信息表等），表与表之间映射关系建好，是1对1，还是1对多，未来根据这个来见数据模型。

这个阶段，我们会从三个过程去考虑，

业务领域过程：用实体建模法来进行制作，抽象出业务中涉及到的实体对象，实体之间发生的业务过程以及对这个业务过程的补充说明，如资产质量这一块，我就可以把几个业务主体提炼出来：个人客户，公司客户，经办机构，产品等

业务过程：个人客户或是公司客户向机构进行借款贷款。以及对业务过程的说明，通过什么渠道来贷款的等。

然后就是到逻辑阶段，主要遵从范式建模理论，把实体的属性与实体与实体之间的关系进行深入剖析理解，方便后面进行维度建模，先规范化在逆规范化

比如个人客户这块有哪些属性（客户id，名称，性别和年龄等）

公司客户这块有哪些属性（公司id，名称，规模，性质等）

经办机构之类的属性，和产品相关属性都细致化。

物理阶段，考虑数据库的特点来实现，进行维度建模，分维度表和事实表这样子。把客户的相关信息合并到一起，把产品类的信息合并在一起。

主要是取数逻辑和表结构的设计，服务于dw与dm层。

3、开发阶段，我们就根据需求文档和设计文档进行开发，我们会在开发环境上进行开发，我开发一般是先把目标表创建好，然后根据mapping来写取数sql，这里面还会用到函数计算某些指标，写好之后确保没有问题了，再编写数据同步用的存储过程，实现数据同步到目标表，写这个存储过程的时候，我们会加上注释,还要加上日志和异常处理，方便后面监控，设置好同步方式，一般是增量，这个写好了，检查没问题了就可以了。我们这里存储过程函数什么的一般会放到包里。

4、测试阶段，我们先自己测试自己写的代码，一般会写一个自测sql脚本，就是统计一下来源的数据量和目标的数据量，在找一个指标统计一下，看看能不能对的上。没问题了就集成测试了，后面是UAT测试。

5、上线阶段，我们开发会把写好的sql脚本提交到SVN上，然后负责上线的人员就会部署到线上了，都是客户那边操作的。后期我们也会跟进一些bug什么的。

基本流程就这些了。

# 3、Hadoop项目

项目介绍（话术）：

我最近也在学习大数据方面的知识，因为之前也参与改造升级了大数据分析系统，

我们是将源数据根据记录时间存入 HDFS 中;然后使用 Hive 创建外部表管理 HDFS 中存放的数据，或者是直接把数据load加载到

hive的ods层，然后根据不同需求创建含有所需字段的表筛选数据，并创建对应的目标表，一般是分区表，每天存一份数据;

我们在hive中创建ODS,DW, DM层，并进行数据同步的处理，最后用shell封装脚本，调试脚本。

# 4、简单说下hadoop：

我说下我的理解，

首先hadoop就是一个大的存储，未来会代替传统数据库存储了，它的优势就是HDFS，分布式文件存储，分就是把数据存储在多台机器上，

文件存储就是数据都是以文件的方式存储的，这样就可以解决数据量大，数据种类复杂的存储了，

然后要想统计计算hadoop里面的数据要用到mapreduce，是个java框架，然后我觉得挺难的，后来就了解到有hive这个组件了，

Hive是基于Hadoop的一个数据仓库工具，可以将HDFS上面结构化的数据文件映射为一张数据库表，并提供类SQL查询功能。

然后这个sql我也了解了一下，

传统数据库上能使用的，基本在hive里都能使用，后面会有一些差异，最近也在看这块，比如如何创建表啊，要用到分隔符啊之类的，

我觉得后面就是在数仓里面把传统数据库换成hive这种，

把存储方式换一下，但是开发流程好像一样，也要在hive里进行分层，然后建设数据仓库这样，进行数据的统计分析。开发的时候要配合shell脚本使用。

# 5、hive基本原理：

首先Hive利用HDFS存储数据，也就是数据是存在hadoop上的，书写的sql语句最终会转化成MapReduce程序执行。

# 6、Hive和数据库有什么区别？

Hive 和数据库除了拥有类似的查询语言，也就是sql，再无类似之处。最重要的就是数据存储位置，

Hive 存储在 HDFS 。

数据的，

Hive 执行延迟较高，也就是查询会很慢。数据库的执行延迟较低。当然，这个是有条件的，即数据规模较小，

当数据规模大库将数据保存在块设备或者本地文件系统中。

然后是对于数据的更新，Hive中不建议对数据的改写。而数据库中的数据通常是需要经常进行修改到超过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。Hive支持很大规模的数据计算；数据库可以支持的数据规模较小，

这个只是相对来说的。

# 7、问业务，问指标如何计算的？要提前了解业务知识和计算公式

广东南粤银行驾驶舱项目：深圳市南山区

说明：贷款分为五类：正常贷款，关注贷款，次级贷款，可疑贷款和损失贷款，后三类组成为不良贷款

表：信用卡交易明细表，信用卡客户信息拓展表，信用卡账户信息拓展表中同步数据

表字段：信用卡号码，客户账户id，贷款日期，贷款总金额，还款日期，已还本金，逾期金额，逾期时长等根据逾期时长判断是否逾期。

指标：质量资产模块，有个不良贷款率指标，不良贷款率=不良贷款/各项贷款\*100%，贷款拨备率/拨备覆盖率

贷款拨备率：呆、坏账准备金的提取比率：贷款减值准备/各项贷款余额\*100%

贷款拨备率：贷款损失准备（一般准备，专项准备，特种准备）/不良贷款 坏账准备金的使用比率

维度有：区域维度，客户维度（年龄，性别），信用等级维度

# 有哪些维度表？客户量多少

客户，账户，产品，币种，渠道和机构，客户量两百万左右

# 最大的数据的表，每日新增数据量多少

记卡交易明细数据，产不多两个亿的数据量，每天还会新增几百万的数据

# 存储过程每个版本多少个，一个存储过程多少行，报表多少张

每个周期的报表有十来张，包括新增和修改的

存储过程一个版本大概2-3个，700多行，最复杂的就是统计卡的交易明细，粒度比较细，涉及的表多，还要进行一些过滤等，1000行左右。

# 11、对于大表过亿，怎么建分区？

对数据先分表，在分区

可以通过存储过程动态sql实现每年创建一张表，然后年表创建好之后，可以按月来对年表进行分区。避免全表扫描，提升查询效率。

往大表中插入数据，怎么提高效率？

索引会影响insert的性能，可以先让索引失效或者删除索引，再来新建索引。

# 12、项目中做过的一些指标：

客户余额，日均余额指标：客户交易明细表和客户信息表，币种，机构和贷款合同等

存款账户交易月汇总指标，即根据存款账户类型不同交易类型来统计交易笔数和交易金额，用到的表有客户交易明细，客户信息，币种，机构等

# 13、SQL相关--性能优化

性能优化的原则就是：1、缩短响应时间；2、提升系统吞吐量；3、提升系统负载能力
再就是看执行计划，看查询的扫描方式，关联机制以及耗费情况，所有**SQL在执行时一定要在第一时间尽量减少扫描的行数，不管是索引扫描还是全表扫描，一定要保证数据库在执行第一个condition的时候能够最大限度的过滤掉不相关的记录，减少后续的扫描和计算消耗。**

SQL优化肯定不仅仅是执行时间降低，应该是资源使用与执行时间降低之间寻求一种平衡，否则，盲目并行，可能提升不了效率，反而让系统资源消耗殆尽。要尽量减少sQL需要访问的资源总量

对于性能优化，我的优化步骤是：首先总逻辑表达式分析，能否将复杂逻辑改成简单的，

然后查看执行计划，从两个方面分析：一是扫表方式：是使用全表扫面还是索引扫面，如果是全表扫面就查看下是没建索引还是索引失效了，如果失效了就考虑使用HINTS，一般情况下不考虑使用强制使用索引，发现这类问题，就去查看索引为什么失效，是不是对索引进行了运算等，如果是这样就索引更改为运算后的索引。然后看关联机制，有三种机制，嵌套循环关联性能不好，扫表次数多，可以添加并行，还有就是把小表作为驱动表；关联字段相同考虑用哈希关联，有索引考虑用排序合并连接。还有就是where后执行顺序是从后往前，所以将筛选条件放在后面，筛选完了之后在关联，能用where筛选完就不用having，还有子查询多的话是不是使用in,可以改为exist性能会好，还有就是考虑union all替代union等等。（接下来是顺着我说的问我，为什么会想union all替换union，union不是数据更少更好么？答：虽然union数据是更少，但是性能效率并没有union all好，解释大概为union是union all 之后还使用去重，所以其实是union all更高效；还有像是除了上面的情况使索引失效，还有什么情况使索引失效？答：像是运用merge into也会使索引失效）

# 14、SQL相关--排序的方法及区别

①ROW\_NUMBER：

ROW\_NUMBER函数返回一个唯一的值，当碰到相同数据时，排名按照记录集中记录的顺序依次递增。

②DENSE\_RANK：

DENSE\_RANK函数返回一个唯一的值，当碰到相同数据时，此时所有相同数据的排名都是一样的。

③RANK：

RANK函数返回一个唯一的值，当碰到相同的数据时，此时所有相同数据的排名是一样的，

同时会在最后一条相同记录和下一条不同记录的排名之间空出排名。

# 15、etl具体使用

我使用的是kettle工具，首先建立自己的kettle资源库，建立好数据库的连接，如源数据库，数据仓库等。

接下来就是通过kettle的输入和输出建立好数据的输入和输出，设置好参数，找到对应的数据来源表和想要获取的字段，再建立好对应数据仓库层的目标表，使用规范的命名方式，完成设置后保存并运行，并到对应的数据仓库层检测数据是否完成同步了。

重复以上操作，加上之前定好的数据模型，就可以把源数据导入到ods，经过过滤筛选和分析计算到dw、dm层。那就可以得到从源数据到ods的转换作业，ods到dw的转换作业以及dw到dm的转换作业。从ods到dw以及dw到dm我会编写好存储过程，然后在kettle中调用实现数据的同步。

最后就是把以上所有的作业整合到一个job中，完成整体数据的同步。

最后可以设定定点跑的时间，也可以采用bat文件实现自动化的调度。

# 16、Fienreport的具体使用

1. 定义数据连接，Oracle数据库jdbc
2. 选定一个合适的报表页面
3. 新建数据库查询，用SQL语句
4. 报表布局好
5. 选择合适的展示方式，如饼图，柱状图，表格等展示方式
6. 对选择的展示工具设置好数据源等参数，标签，标题，保存并预览核对修正
7. 后期的展示优化
8. 报表平台管理--决策系统--创建目录--添加链接
9. 客户端登陆，只有查看功能
10. 空件

设置参数area的控件类型为下拉框，数据字典为数据库表，数据库为FRDemo，

数据表为订单，实际值与显示值的列名为货主地区。

设置参数province的控件类型为下拉框，数据字典为数据查询，

数据集为动态SQL1，实际值与显示值的列名为货主省份。

设置参数city的控件类型为下拉框，数据字典为数据库表，

数据集为动态SQL2，实际值与显示值的列名为货主城市。

# 18、你对大数据的看法

大数据无处不在，大数据应用于各个行业，包括金融、汽车、餐饮、电信、能源、体能和娱乐等在内的社会各行各业都已经融入了大数据的印迹。制造业，利用工业大数据提升制造业水平，包括产品故障诊断与预测、分析工艺流程、改进生产工艺，优化生产过程能耗、工业供应链分析与优化、生产计划与排程。大数据对各行各业的渗透，大大推动了社会生产和生活，未来必将产生重大而深远的影响。

# 19、 之前的业务量有多大，盈利多少？

？

# 20、并行在工作中是怎样用的？根据什么条件加并行？

/\*+ PARALLEL(表名1,并行数)[(表名2,并行数)……] \*/

一般是经过性能调优之后，查询效率依然低下，是由于本身的数据量十分大，会跟项目经理和数据库管理员沟通，再决定是否使用并行。

# 21为什么要这样建维度表？

根据业务需求和数据内容和市场行情综合判断

# 22、看中他们公司哪一点

# 23、hints怎么调优的？

并行、指定索引扫描

# 24、有一张大表怎么建索引的

分表分区建索引

# 25、union和union all的区别，哪个会好点，为什么？

Lunion all不会去重，union会比较需要连接的表看是否有重复数据，如果有就只保留一条数据。好坏要看项目的具体需求，如果项目不在意有重复数据，那么使用union all会好很多，如果项目不允许存在重复数据，那么还是得使用union。

# 26、有什么想问的？

问福利待遇

公司的发展战略等

公司对我的要求是怎么样的？

# 27、数据怎么清洗？清洗机制

首先是根据需求过滤不相关数据，对于一些重复数据进行去重，对于一些空白和缺失数据进行转换保证数据信息的完整性，比如有个客户的性别为null，那么就设为未知或者看能不能通过其他信息来判断他的性别，对于一些客户的隐秘信息进行脱敏处理，单位度量的同一。为方便后期的建模，对于维度表添加代理主键等操作

主要是写SQL来进行过滤清洗，也可以使用kettle的一些功能来进行处理，比如kettle有个可以去重的控件。

维度转代理的，数据整合成维度和事实等

# 去重的方式

具体思路是，首先创建一个临时表，然后将DISTINCT之后的表数据插入到这个临时表中;然后清空原表数据;再讲临时表中的数据插入到原表中;最后删除临时表。--大数据量时效果好！

Rowid Oracle中专用的，具体思路是先通过rowid找出唯一记录，然后删除不在唯一记录中的其他记录删除。

distinct、exists、group by 、rowid，其中distinct性能最差

# 29、聚合函数和分析函数有什么区别？

◎普通的聚合函数用group by分组，每个分组返回一个统计值，

◎而分析函数采用partition by分组，并且每行都可以返回一个统计值，

◎写法上也有区别，分析函数要带 over（）。

over 之后的括号里可以什么都不写

---如果写了partition by意味着分组统计

---如果加了order by 是累计统计

排名类的分析函数：

----over 之后括号里必须要有order by，

----可以没有partition by，可以不分组

# 30、怎么查看报错，还有解决方案？

我一般是先查看日志，看看出错位置和出错类型是什么？

看日志提示错误是什么，如果常见问题如标点符号缺失或是错误，主键冲突，内存溢出等，就可以自己解决掉，其他一些可能少见的异常也可以通过百度出错代码查找相关原因或是咨询同事和上级。

# 遇到的印象深刻Bug

有一个job前一天执行成功了，但是第二天在执行的时候执行失败了

首先我看调度日志是怎么样的情况，看看是否有开始没有结束的情况，来判断是不是可能锁表了，发现不是。然后仔细核查我写的SQL有没有写错，然后一个个去执行查看，发现有个存储过程报错了，报错原因是找不到表，我很奇怪为什么会找不到表？于是我问了前面一层的开发人员才知道他昨天晚上把表名修改了。

# 32、有一张表下午五点才出数据，但是依赖的表早上十点就已经跑完了

我们开始对这个问题进行分析并解决，可能是两方面的问题，一是性能问题，或者是调度问题。先看了调度日志，看是否所有的前置job都跑完了，排除因某个前置job没完成的情况，后面就是会查看一下监控日志，看下跑数的时间，如果确实是目标表更新很慢，再进一步分析慢的原因，比如是否在目标表上建了过多的索引导致更新变慢，或者在查询的时候或者数据量变大的情况导致的性能的问题，可以建索引或者分区表方式进行优化，在分析问题的过程中药关注SQL的执行计划，必要的时候可以使用优化器进行优化。

另外我们也会定期检查跑数的时间，看看跑数的过程是否越来越慢等并进行优化。

# 33、如何写存储过程的

先设定好存储过程要实现的目标

接下来就是建立存储过程的名字与参数，注意命名规范

在就是定义好要用到的变量或者是游标，如果后面需要额外的变量可以后面增加 %rowtype，借用已有表的字段数据类型和长度

后面是写好begin end 一般这种有开始有结尾的语句我都会一开始就成对写，

然后在begin end 中写SQL。

一般还会在SQL中加上日志监控以及异常处理，方便监控

后面就是看看执行情况，是否可以优化等。

将一些经常用到的操作封装起来，以便于后续的多次调用

存储过程主要用做数据同步。

# 34、函数与存储过程的区别？

SP 一般用户数据同步/更新。FUN 一般用于计算指标 。

SP 不需要返回值。 FUN 一定要返回 1 个计算结果。

SP 是通过 BEGIN END /CALL 的方式来调用。FUN 通常用 SELECT 调用

定义 SP 不需要加 RETURN ,只需要在 BEGIN END 之间写存储数据的逻辑即可。

定义 FUN 必须要加 RETURN ,然后在 BEGIN END 之间进行计算，在 END 之前将结果通过 RETURN 返回出来。

# 35、函数与存储过程可不可以互相调用？

从语法的角度来说是可以互相调用的。但是从实际开发的角度，我们通常会用 SP 调用 FUN ，但是很少用 FUN 调用 SP，因为计算需要确保数据稳定。

# 36、分区和索引的区别

索引和分区最大的区别就是索引不分割数据库，分区分割数据库。

索引其实就是拿额外的存储空间换查询时间，但分区已经将整个大数据库按照分区列拆分成多个小数据库了。

# 37、索引失效的情况

隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误

对索引列进行运算导致索引失效,我所指的对索引列进行运算包括(+，-，\*，/，! 等)

使用Oracle内部函数导致索引失效.对于这样情况应当创建基于函数的索引

以下使用会使索引失效，应避免使用；

a. 使用 <> 、not in 、not exist、!=

b. like "%\_" 百分号在前（可采用在建立索引时用reverse(columnName)这种方法处理）

c. 单独引用复合索引里非第一位置的索引列。应总是使用索引的第一个列，如果索引是建立在多个列上, 只有在它的第一个列被where子句引用时，优化器才会选择使用该索引。

d. 字符型字段为数字时在where条件里不添加引号.

e. 当变量采用的是times变量，而表的字段采用的是date变量时.或相反情况。

# 40、建索引：有建本地索引吗

分区索引分为本地(local index)索引和全局索引(global index)。

其中本地索引又可以分为有前缀(prefix)的索引和无前缀(nonprefix)的索引。而全局索引目前只支持有前缀的索引。

Oracle执行基于什么来选择一个最优执行 执行时间cost

# 41、Linux命令都用来干啥

操作虚拟机，load 把Linux文件转移到hdfs中去，执行一些脚本， Hadoop fs- put 路径 到hdfs的文件系统中去。

# 42、隐式游标 和 显式游标的区别

1、隐式游标：SELECT ... INTO ...

 1）不需要声明，直接就可以使用

 2）查询的结果集只能是1行，不能是多行或者没有结果(0行)

2、显示游标

 1）需要声明，然后再拿来使用

 2）查询的结果集可以是多行，也可以是没有结果(0行)

 3）使用方法有2种：

 1、需要手动打开游标，提取数据，关闭游标

2、使用FOR循环依次可以拿到游标指向的结果集种的数据(不需要手动去处理游标)

33、他根据第三个问题又追我高水位线怎么去掉，

我说truncate。然后他停止询问问题，问我有什么要了解的，面试结束。

# 43、什么是表外业务

表内业务是指资产负债表中，资产和负债栏目可以揭示的业务:例如贷款、贸易融资、票据融资、融资租赁、透支、各项垫款等。表外业务是指资产负债表不能揭示的业务，例如保证、银行承兑汇票等。

# truncate， delete，drop的区别；

除几行用什么，删除表用什么，那些会删除表结构，那些不会；那些可以回滚，那些会释放空间

# 45、分析函数怎么行转列

Pivot ,分析函数leg和lead，但是这两个分析函数实现行转列有局限性，需要比较整齐的数据，其他比较常用的方法有用max或者min函数结合case when 实现，或者用关联方式。

# 46、索引的优缺点

创建索引可以大大提高系统的性能。

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

也许会有人要问：增加索引有如此多的优点，为什么不对表中的每一个列创建一个索引呢？因为，增加索引也有许多不利的方面。

第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

 **嵌套循环关联**（**NESTED LOOPS JOIN（NL）**）：是指依次从驱动表中提取一条记录，遍历被探查表，将匹配的记录放入待展示的缓存区中。

优点：适用广，占用内存小，展现快

缺点：需要不停地从硬盘中读取扫描表，性能不好

**注意**：把两张表最终需要关联的数据对比，大表适合做被探查表，因为可以减少从硬盘读取扫描表的次数。

**哈希关联**（**Hash Join（HJ）**）：计算出整张被探查表关联字段的哈希值，这些哈希值和整张被探查表一起放入缓存区，然后从驱动表逐条取记录，计算出关联字段对应的哈希值，再与被探查表的哈希值匹配,匹配上了再精准匹配每一条记录。

优点：性能好，匹配次数大大减少

缺点：只适用于等值关联，占用内存较大

**注意**：把两张表最终需要关联的数据对比，小表适合做被探查表，因为怕缓存不够。如果缓存足够的前提下，大表适合做被探查表。

**排序合并连接**(**Sort Merge Join (SMJ)** )：是指将关联的a表跟b表分别进行排序，生成临时的两张表后，随机取一张表逐条抽取记录与另一张表匹配。

优点：适合有索引的两张表或者不等关联

缺点：排序性能消耗大，占用内存大

# 47、视图是什么

视图（View）实际上是一张或者多张表上的预定义查询，这些表称为基表。从视图中查询信息与从表中查询信息的方法完全相同

可以限制用户只能通过视图检索数据。这样就可以对最终用户屏蔽建表时底层的基表，具有安全性。可以将复杂的查询保存为视图，屏蔽复杂性。

视图没有办法进行修改，只能进行查询。

# 48、缓慢变化维分那几种，各种的作用是什么；

添加维度行：记录所有的历史变化，增加字段开始时间，结束时间和标识字段。用leg和lead函数实现。用到代理主键。Type2

添加维度列：对需要分析历史信息的属性添加一列，来记录属性变化前后的值，可以分析当前以及上一次变化的情况。Type3

直接覆盖原值：type1.最简单的做法，不记录历史信息。

实际开发过程中，三种方式根据业务需求来定。

熟悉数据仓库各类建模理论（维度建模：星型模型和雪花模型）

以及数据仓库层级体系（ods-dw-dm）

销售数据决策BI系统（保密）

事实表：销售订单详情（销售订单详情id，订单id，门店id，商品id，用户id，订单金额等）、库存表（商品id，门店id，商品分类id，商品数量等）

维度：门店（门店id，门店地址，负责人等信息）、商品（商品id，商品分类，商品其他基本信息）、客户（客户id，性别、年龄、地区等），日期表（）

指标：门店销售排名，商品月销售排名，商品分类月销售排名，各区域各商品销售排名，不同性别的客户数量分布，不同年龄的客户分布以及喜爱的商品分类等等

电商公司：深圳市龙岗区坂田街道

银行项目：深圳南山区分行

# 49、维度建模基本概念：

维度建模以分析决策的需求出发构建模型，构建的数据模型为分析需求服务，因 此它重点解决用户如何更快速完成分析需求，同时还有较好的大规模复杂查询的响应性能。

事实表：发生在现实世界中的操作型事件，其所产生的可度量数值，存储在事实表中

表里没有存放实际的内容，他是一堆主键的集合，与各维度表相关联的外键。事实表的度量通常是数值类型。比如订单详情表，库存表等。

维度表示：你去分析事实表的一个角度，比如区域，日期等。

维度表的同步策略：可能会有变化的数据可以存储每日全量或者使用拉链表（缓慢变化维）。

事物型事实表：每日增量，每日分区hive

周期型事实表：拉链表，得到每日新增和变化量。

# 星型模型和雪花模型

Dw层建模：星型模型

星形模式是最常用的维度建模方式。星型模式是以事实表为中心，所有的维度表直接连 接在事实表上，像星星一样。存在冗余，但是性能会好一些，。

星形模式的维度建模由一个事实表和一组维表成，且具有以下特点：

维表只和事实表关联，维表之间没有关联；

每个维表主键为单列，且该主键放置在事实表中，作为两边连接的外键；

以事实表为核心，维表围绕核心呈星形分布；

雪花模式是对星形模式的扩展。雪花模式的维度表可以拥有其他维度表的，虽然

这种模型相比星型更规范一些，但是由于这种模型不太容易理解，维护成本比较高，而且性能方面需要 关联多层维表，性能也比星型模型要低。所以一般不是很常用。

星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而星座模式是基于多张事实表的，而且 共享维度信息，。在业务发展后期，绝大部分维度建模都采用的是星座模式。

优点：维度建模非常直观，紧紧围绕着业务模型，可以直观的反映出业务模型中的业务问题。不需要经过特别的抽象处理，即可以完成维度建模，最大的作用其实是为了解决数据仓库建模中的性能问题

缺点：此会导致大量的数据处理工作，当业务发生变化，需要重新进行维度的定义时，往往需要重新进行维度数据的预处理。而在这些与处理过程中，往往会导致大量的数据冗余。

如果只是依靠单纯的维度建模，不能保证数据来源的一致性和准确性，而且在数据仓库的底层，不是特别适用于维度建模的方法。维度建模的领域主要适用与数据集市层

# 51、主题域的划分

主题域：主题域是指联系较为紧密的数据主题的集合，如订单主题域，可以包含订单详情，商品主题表，用户主题表，日期等。可以根据业务的关注点，将这些数据主题划分到不同的主题域，主题域的确定必须由最终用户和数据仓库的设计人员共同完成。

# 如何确保数据质量？

在dm层如何保证数据质量，首先仔细分析理解需求和指标，有疑问及时沟通处理

然后是进行单元测试时自己比对一下数据结果有没有能对得上

还有一个代码评审环节可以来确认数据同步的方式和需求能不能对得上

进入测试阶段，我们有专门的测试人员来进行测试，对于他们发现的问题，我会分析问题并解决，在看看其他地方是否也有类似的问题，避免漏测。

上线后跟进一些bug和做一些总结。

# 53、为何要分层？

分层的原因是在管理数据，能够对数据有一个更加清洗的掌控

清晰的数据结构，在使用表的时候更方便的定位和理解

数据血缘追踪，能够快速定位到问题且清楚他的危害范围

减少一些通用的中间层数据计算开发

把复杂任务简单化，每一层处理单一步骤，便于维护数据的准确性，当数据出现问题，不用修复所有数据，只需要定位好，从有问题的步骤开始修复即可。

屏蔽业务的影响，不必改一次业务就重新接入数据。

数据分层的思想：

数据分层每一家公司都可以根据自己的业务需求分成不同的层次，但是最基础的分层思想，理论上数据分为三个层次，数据运营层，数据仓库层，数据服务层。基于这个层次之上添加新的层次，来满足不同业务需求。

ods,数据运营层 operate data store

操作数据存储，etl抽取，洗净，从业务系统数据库传输到本层，数据模型与业务系统数据模型保持一致，按主题逻辑划分。

数据仓库层 data warehouse

从ods中获得数据按照主题建立各种数据模型

应用层 ads

主要是提供数据产品和数据分析使用的数据，给报表开发前段开发的

例如：

1\Ods:功能：为DWD层提供基础原始数据，可减少对业务系统的影响

建模方式：从业务系统增量抽取 保留时间由业务需求决定 可分表进行周期存储 数据不做清洗转换与业务系统 数据模型保持一致 按主题逻辑划分 。

2\dwd 数据明细层

功能 ：为dw层提供来源明细数据，提供业务系统细节数据，为未来的分析类需求扩展提供历史数据。

建模方式：数据模型与ODS层一致，不做清洗转换处理 为支持数据重跑可额外增加数据业务日期字段 可按年 月日进行分表 用增量ODS层数据和前一天DWD相关表进行merge处理

3\dwb/s 数据汇总层

功能：为dw和st提供不同粒度的数据

Dwb是根据dwd明细数据进行转换清洗：维度转代理键 身份证清洗 会员注册来源清晰 字段合并 空 值处理 脏数据处理 IP清晰转换 账号余额清洗 资金来源清洗等；

Dws：DWS是根据DWB层数据按各个维度ID进行高粒度汇总聚合

建模方式及原则：

聚合汇总增加派生事实

关联其他主题的事实表

Dwb保持低粒度的汇总加工数据，dws保持高粒度的汇总数据

数据模型可能会采用反范式设计，进行一些信息的合并。

4、Dm数据集市层

功能：可以是一些宽表，是根据DW层数据按照各种维度或多种维度组合把需要查询的一些事实字段进行汇总统计并作为单独的列进行存储； 满足一些特定查询 数据挖掘应用，应用集市数据存储

建模方式：尽量减少数据访问时计算，优化检索；

维度建模，星型模型，星座模型等

事实拉宽，度量预先计算；

分表存储

5、st数据应用层（ads）

功能：面向用户应用和分析需求，包括前端报表 分析图表 KPI 仪表盘 OLAP 专题等分析，面向最终 结果用户；适合作OLAP 报表模型；根据DW层经过聚合汇总统计后的粗粒度事实表

建模方式及原则：

保持数据量小； 维度建模，星形模型； 各位维度代理键+度量；增加数据业务日期字段，支持数据重跑；不分表存储

# 54、数据仓库建模篇：

1. 什么是数据模型？

数据模型（关系型，网状，层次）是抽象描述现实世界的一种工具和方法，是通过抽象实体与实体之间的关系，对实体与实体之间关系的定义与描述，来表达现实实际业务中的业务关系。

1. 为什么需要数据模型？

全面了解该单位的业务架构图和整个业务的运行情况，进行全面的业务梳理，改进业务流程，提高业务效率，指导我们的业务部门的生产

建立全方位的数据视角，消灭信息孤岛和数据差异

能够保证整个企业的数据的一致性，各个部门之间数据的差异将会得到有效解决。

解决业务的变动和数据仓库的灵活性，能够很好的分离出底层技术的实现和上层业务的展现，从而达到整个数据仓库系统的灵活性

1. 如何建设数据模型？

数据仓库模型架构：

系统记录域：数据仓库业务数据存储区，数据模型在这里保证了数据的一致性。ods/dwd

内部管理域：存储数据仓库用于内部管理的元数据，数据模型在这里能够帮助进行统一的元数据的管理。

汇总域：数据来自于系统记录域的汇总，数据模型在这里保证了分析域的主题分析的性能，满足了部分的报表查询。dwb/s

分析域：主要用于各个业务部分的具体的主题业务分析。这部分数据模型可以单独存储在相应的数据集市中。dm/st

反馈域：这部分数据模型主要用于相应前端的反馈数据，数据仓库可以视业务的需要设置这一区域。

 数据模型建设理论：三范式建模、维度建模以及实体建模等

建模理论的介绍：

建模的过程：

业务建模：业务层面的分解和程序化,深入了解各个业务部门的内具体业务流程并将其程序化。提出修改和改进业务部门工作流程的方法并程序化。

业务主线（养老）--业务主题（老人养老保险分析，参保单位构成分析，收费情况分析；中人养老保险分析，单位缴费情况，支出情况分析；新人养老保险分析，欠费单位情况分析，收支平衡预警）

领域模型：对业务模型进行抽象处理。

抽象方法层：实体建模

领域概念层：抽取关键业务概念，并将之抽象化（参与方（个人，公司，经办机构），产品，事件，说明）等

具体业务层：将业务概念分组，按照业务主线聚合。理清分组概念内的业务流程并抽象化，概念模型中实体之间的一些不同组合而已。

个人养老金征缴：个人 公司 一次征缴 经办机构 实体组合

业务主线层：在业务建模阶段即已经完成这方面的划分。我们一般通过这种大的业务主线来划分整个业务模型大的框架。养老部门

逻辑模型：对领域模型的概念实体以及实体之间的关系进行数据库层次的逻辑化

业务概念实体化，并考虑具体属性

事件实体化，并考虑属性内容

说明实体化，并考虑属性内容

实例每一个抽象主题，并考虑其属性，例如对人和公司抽象实体实例化，在业务模块中考虑实体的属性包括哪些，如人包括的属性有年龄，性别，地址等。

找出抽象事件的关系，并对其进行说明。将整个概念模型完整串联成一个有机实体，表达出业务之间的关联性。在这个阶段我一般是采用三范式建模理论来建设，表达出实体的属性以及实体与实体之间的联系。规范化操作。

个人（名称 ，性别 ，年龄 ，地址，籍贯等）

公司（名称，规模，性质等）

机构（名称，功能，地点等）

征缴详情（个人id 公司id 机构id 产品 id征缴额） 事实表等

物理建模：逻辑模型针对不同关系型数据库的物理化以及一些性能等技术问题，是具体的实现。

考虑到数据仓库平台的不同，因此，数据模型得物理建模过程可能会稍微有一些不同，

针对模型的性能考虑，对特定平台作出相应的调整
针对管理的需要，结合特定的平台，做出相应的调整

这个阶段可以采用反范式设计，采取维度建模的星型模型或星座模型，进行一些信息的合并，提高查询性能，方便需求的完成。

在建模的过程中，既需要丰富的业务知识，也需要过硬的技术实力，同时还需要一个适合的方法论来指导我们针对我们的业务来进行抽象，处理，生成各阶段模型。我采用的规范化和你规范化，德国车厂的操作，拧螺丝。

完整的数据仓库的数据模型应该能够相应灵活多变的前端业务的需求，而其本身的模型架构具有很强的灵活性

数据仓库建模方法：

范式建模法：主要解决关系型数据库得数据存储

三范式的定义：所有字段值都是不可分解的原子值

满足第一范式为前提，非主属性必须完全依赖于整个主键，而非主键的一部分

每个非主属性

满足第二范式为前提，每一列数据都和主键直接相关，而不能间接相关

结合了业务系统的数据模型，能够比较方便的实现数据仓库的建模。但其缺点也是明显的，由于建模方法限定在关系型数据库之上，在某些时候反而限制了整个数据仓库模型的灵活性，性能等，特别是考虑到数据仓库的底层数据向数据集市的数据进行汇总时，需要进行一定的变通才能满足相应的需求。

数据仓库的域模型应该包含企业数据模型的域模型之间的关系，以及各主题域定义。数据仓库的域模型的概念应该比业务系统的主题域模型范围更加广。
在数据仓库的逻辑模型需要从业务系统的数据模型中的逻辑模型中抽象实体，实体的属性，实体的子类，以及实体的关系等。

维度建模：按照事实表，维表来构建数据仓库，数据集市

星形模式是最常用的维度建模方式。星型模式是以事实表为中心，所有的维度表直接连 接在事实表上，像星星一样。存在冗余，但是性能会好一些，。

星形模式的维度建模由一个事实表和一组维表成，且具有以下特点：

维表只和事实表关联，维表之间没有关联；

每个维表主键为单列，且该主键放置在事实表中，作为两边连接的外键；

以事实表为核心，维表围绕核心呈星形分布；

雪花模式是对星形模式的扩展。雪花模式的维度表可以拥有其他维度表的，虽然

这种模型相比星型更规范一些，但是由于这种模型不太容易理解，维护成本比较高，而且性能方面需要 关联多层维表，性能也比星型模型要低。所以一般不是很常用。

星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而星座模式是基于多张事实表的，而且 共享维度信息。在业务发展后期，绝大部分维度建模都采用的是星座模式。

优点：维度建模非常直观，紧紧围绕着业务模型，可以直观的反映出业务模型中的业务问题。不需要经过特别的抽象处理，即可以完成维度建模，最大的作用其实是为了解决数据仓库建模中的性能问题

缺点：此会导致大量的数据处理工作，当业务发生变化，需要重新进行维度的定义时，往往需要重新进行维度数据的预处理。而在这些与处理过程中，往往会导致大量的数据冗余。

如果只是依靠单纯的维度建模，不能保证数据来源的一致性和准确性，而且在数据仓库的底层，不是特别适用于维度建模的方法。维度建模的领域主要适用与数据集市层

实体建模：原理：客观世界应该可以分成由一个个实体，以及实体与实体之间的关系组成

将整个业务也可以划分成一个个的实体，而每个实体之间的关系，以及针对这些关系的说明

由于实体建模法，能够很轻松的实现业务模型的划分，因此，在业务建模阶段和领域概念建模阶段，实体建模法有着广泛的应用

再没有现成的行业模型的情况下，我们可以采用实体建模的方法，和客户一起理清整个业务的模型，进行领域概念模型的划分，抽象出具体的业务概念，结合客户的使用特点，完全可以创建出一个符合自己需要的数据仓库模型来。

# 数据敏感心得，怎么体现？

# 56 Hive相关知识

## 1、4个By区别？

1）Sort By：分区内有序；

2）Order By：全局排序，只有一个Reducer；

3）Distrbute By：类似MR中Partition，进行分区，结合sort by使用。

4） Cluster By：当Distribute by和Sorts by字段相同时，可以使用Cluster by方式。Cluster by除了具有Distribute by的功能外还兼具Sort by的功能。但是排序只能是升序排序，不能指定排序规则为ASC或者DESC。

## 2、内部表和外部表区别？

Hive 创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。

在删除表的时候，内部表的元数据和数据会被一起删除，而外部表只删除元数据，不删除数据。

## 3、存储格式

STORED AS

TEXTFILE 使用纯文本存储。

SEQUENCEFILE 对文件进行压缩存储。

**分区**

PARTITIONED ：表示的是分区，不同的分区会以文件夹的形式存在