**1.有一张表下午五点才出数据，依赖的表早上十点就已经跑完了，是什么原因导致表这么慢出数据？**

**类似的问题**

**1.1从数仓同步数据本来是早上9点出结果的，结果一直到下午4点才出结果，怎么找问题的原因**

**回答思路：**

 第一，跑数的时间大大超过预计的时间 极大可能是性能问题 。

 第二，除了性能问题之外可能是调度问题。

 第三，定位问题不能靠主观想象，一定要检查日志，通过日志分析原因 。

**参考回答：**

 1 前置依赖的表上午10点就跑完了，但是目标表的JOB一直到下午5点才出结果 。首先我会看一下日志。

 先看调度入职，再次检查一遍是否是所有的前置job都跑完了，排除因某一个前置 job 没跑完，之前同事检查的时候没发现，导致目标表的job一直等待。

 排除上面的可能以后 ，再分析是否是性能问题，检查监控日志，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 检查是否是在目标表上建了过多的索引导致更新变慢。

 另外我们平时也要定期（每周或者每2周对之前的跑数job进行检查）检查job是不是越来越慢，可能是因为数据量过大导致的。

 数据量变大导致的性能问题可以通过 创建索引 或者 通过分区表的方式进行优化。

 在分析问题的过程中我们要关注SQL的执行计划，必要的时候也可以使用优化器进行SQL优化。

 1.1 回答思路同上，因为这个问题中面试官没有说前置job是否跑完了，所以我们必须去检查一下前置依赖的job是否都跑完了 ，什么时候跑完的，然后再进一步分析是否是更新目标表的性能很慢。

**2.数据仓库是用什么方式来保存历史数据的？**

**类似的问题**

**2.1 数据仓库是通过什么方式来反应历史变化的？**

**2.2 要如何设计一张表来反应历史变化？**

**回答思路：**

 只要涉及到 保存历史数据 ，拉链表，缓慢变化维 这样的关键词，那么问的都是如何保留历史数据 或者如何通过拉链表来保存缓慢变化的的历史数据。

**参考回答：**

 2 数仓保留历史数据通常是在 ODS层创建拉链表来保存历史数据的。

 2.1 数仓中看数据的历史变化，是通过缓慢变化维的方式来记录数据的变化过程 ，每次更新数据都会通过update进行逻辑删除，然后将最新的数据插入到拉链表中，并将历史数据更新成无效状态，将最新数据更新成有效状态。

 2.2 首先看需求，要保留多少历史数据，如果是只保留最近的数据，那么在目标表中需要创建一个标记字段记录和区分最新的数据和最近的一批数据。

 如果要保留所有的历史数据，那么我们会使用缓慢变化维的技术 通过拉链表来实现。

缓慢变化维：

维度建模的数据仓库中，有一个概念叫Slowly Changing Dimensions，中文一般翻译成“缓慢变化维”，经常被简写为SCD。缓慢变化维的提出是因为在现实世界中，维度的属性并不是静态的，它会随着时间的流失发生缓慢的变化。这种随时间发生变化的维度我们一般称之为缓慢变化维，并且把处理维度表的历史变化信息的问题称为处理缓慢变化维的问题，有时也简称为处理SCD的问题。

处理缓慢变化维的方法通常分为三种方式：

第一种方式是直接覆盖原值。这样处理，最容易实现，但是没有保留历史数据，无法分析历史变化信息。第一种方式通常简称为“TYPE 1”。

第二种方式是添加维度行。这样处理，需要代理键的支持。实现方式是当有维度属性发生变化时，生成一条新的维度记录，主键是新分配的代理键，通过自然键可以和原维度记录保持关联。第二种方式通常简称为“TYPE 2”。

第三种方式是添加属性列。这种处理的实现方式是对于需要分析历史信息的属性添加一列，来记录该属性变化前的值，而本属性字段使用TYPE 1来直接覆盖。这种方式的优点是可以同时分析当前及前一次变化的属性值，缺点是只保留了最后一次变化信息。第三种方式通常简称为“TYPE 3”。

在实际建模中，我们可以联合使用三种方式，也可以对一个维度表中的不同属性使用不同的方式，这些，都需要根据实际情况来决定，但目的都是一样的，就是能够支持方便的分析历史变化情况。

**3. A表是主表，80万数据，B表是扩展表，100万数据，这两张表相同主键是学生编号，根据这个场景，判断是否存在异常？什么原因造成数据重复？**

**参考回答：**

 面试官您刚才说了 A 表作为主表 数据量是80万，我的理解是这样的，通常外关联才有主表和从表之分，如果是外关联，那么 A 表作为主表的数据肯定是全部取的，但是关联以后的结果集比如 B 的数据量变成了 100 万 ，很明显数据量多了20万，这种明显是数据发散了，

 通常数据发散的原因可能是关联条件有问题 ，比如少些了某个关联字段导致查询出来的结果集变大了，所以我认为应该重新检查和分析一下 A 表和其他表的关联条件是否遗留了某个关联字段 。

 此外还有一种可能，当然这种可能性很小 也不能排除就是 A 表的数据是不是刚开始就有一部分重复了 ，比如原来是 80 万 ，但是因为没有按照3NF原则在表中创建主键 ，结果更新的时候插入了部分重复记录导致结果集发散，这种情况可以先对主表数据按照主键字段去重，然后再表中创建主键 或者 对关联字段创建唯一索引，通过唯一约束来避免此类问题。

**4 有一个job 前一天执行成了，第二天再执行失败了，请你分析失败的原因？**

**参考回答：**

 如果第一天成功，第二天失败，有2种可能，一种是报错，如果报错的话 我们需要先看日志，如果报主键冲突，说明更新的SQL逻辑有问题，我们可以看下代码，更改原来的同步逻辑即可。

 第二种如果不报错，发现脚本一直处于执行状态，可能是锁表了，我会查询一下 锁表的进程，然后杀死该进程，问题就解决了 。

**alter system kill session** 'sid,serial#';

数仓和数据库

1 数仓存储的是汇总的数据，高延时的。（相对稳定）

2 数据库是业务系统的后台，要求实时性。（前台的操作后台立马要刷新，同时后台的数据刷新了，前台其他页面也要同步刷新）

3 数据库使用的OLTP（联机事务处理）

4 数仓使用的OLAP（联机数据分析）

事实和维度

事实表就是你要关注的内容；

维度表就是你观察该事务的角度，是从哪个角度去观察这个内容的。

主题：事实表和维度表的汇总结果 ，数仓中的面向主题，不同的业务主题 ，对应M层输出的各种可视化的报表。

三范式

1NF:字段不可分;

2NF:有主键，非主键字段依赖主键;

3NF:非主键字段不能相互依赖;

解释:

1NF:原子性 字段不可再分,否则就不是关系数据库;

2NF:唯一性 一个表只说明一个事物;

3NF:每列都与主键有直接关系，不存在传递依赖;

三种依赖关系

部分函数依赖：举个例子：通过AB能得出C，通过A也能得出C，通过B也能得出C，那么说C部分依赖于AB。

完全函数依赖：举个例子：通过AB能得出C，但是AB单独得不出C，那么说C完全依赖于AB.

传递函数依赖：举个例子：通过A得到B，通过B得到C，但是C得不到B，B得不到A，那么成C传递依赖于A

建表的规范：

建表时需要在表中创建 主键 ，主键列要和其他的列是相关的（不相关的数据不要存在一张表中），其他的列必须跟主键的全部字段关联而不能只跟主键部分相关。

主键可以由多列组成，多列才能确保数据的唯一性，其中的任何一列都无法确定数据的唯一性。

主键的特点，非空且唯一 ：

主键列的数据有非空约束。

唯一包含2层意思 一是表中的主键只允许创建一个 ，二是主键列（单列，组合主键）的数据必须唯一，不允许重复。

星型模型与雪花模型

事实表和维度表的关联方式。

如果事实表和维度表之间可以直接关联，那么这样的数据模型叫做 星型数据模型。

只要有任何一个维度表跟事实表之间是间接关联的，那么这种数据模型就是 雪花模型。

2种模型的区别：

1 性能不同，星型性能更好一些 。

2 存储空间不同，雪花模型更节省空间。

在项目组不同的业务 可能会用不同的数据模型。

未来有一天，我们开发人员要会设计数据模型，初期面试的时候统一说 数据的逻辑模型都是老大（项目经理）设计的，我们只是负责物理模型（物理表)的设计。通过数据字典维护。

数据字典 是我们开发人员负责设计和维护的，一直到上线之前。

----------------------------------------------------------------------------------------------------------------------------------------------------

保存历史数据的方法

对历史数据的处理方式有三种 ：

1 不保存历史数据，直接更新

2 保存所有的历史数据 （拉链表）， 通常也是分区表 。

3 保存当前或者上一次的历史数据

1. table、extent、segment、block之间的关系；

答：这些都是Oracle存储的逻辑结构，table创建时,默认创建了一个data segment,每个data segment含有min extents 指定的extents 数,每个extent 据据表空间的存储参数分配一定数量的blocks

tablespace: 一个数据库划分为一个或多个逻辑单位，该逻辑单位成为表空间;每一个表空间可能包含一个或多个Segment;

Segments: Segment指在tablespace中为特定逻辑存储结构分配的空间。每一个段是由一个或多个extent组成。包括数据段、索引段、回滚段和临时段。

Extents: 一个extent 由一系列连续的blocks 组成.ORACLE 为通过extent 来给segment分配空间。

Blocks：Oracle 数据库最小的I/O存储单位，通常是8K。

1. tablespace、datafile的关系

答：一个tablespace可以有一个或多个datafile,每个datafile只能在一个tablespace内,

table中的数据,通过hash算法分布在tablespace中的各个datafile中,

tablespace是逻辑上的概念,datafile则在物理上储存了数据库的种种对象。

1. 日志的作用

答：恢复 、事物的快速提交、写缓存。

1. 表的连接方式（不是内外关联）；

答：（1）HASH JOIN 散列连接：优化器使用两个表中较小的表（通常是小一点的那个表或数据源）利用连接键（JOIN KEY）在内存中建立散列表，将列数据存储到hash列表中，然后扫描较大的表，同样对JOIN KEY进行HASH后探测散列表，找出与散列表匹配的行

1. SORT MERGE JOIN 排序合并连接：是先将关联表的关联列各自做排序，然后从各自的排序表中抽取数据，到另一个排序表中做匹配。
2. NESTED LOOP 嵌套循环连接：Nested loops 工作方式是循环从一张表中读取数据(驱动表outer table)，然后访问另一张表（被查找表 inner table,通常有索引）。驱动表中的每一行与inner表中的相应记录JOIN。类似一个嵌套的循环。
3. 说说你对索引的认识

答：通常来说，在表上建立恰当的索引，查询时会改进查询性能。但在进行插入、删除、修改时，同时会进行索引的修改，在性能上有一定的影响。有索引且查询条件能使用索引时，数据库会先度取索引，根据索引内容和查询条件，查询出ROWID，再根据ROWID取出需要的数据。由于索引内容通常比全表内容要少很多，因此通过先读索引，能减少I/O，提高查询性能。

1. 绑定变量是什么?绑定变量有什么优缺点?

答：绑定变量是指在SQL语句中使用变量，改变变量的值来改变SQL语句的执行结果。

优点：使用绑定变量，可以减少SQL语句的解析，能减少数据库引擎消耗在SQL语句解析上的资源。提高了编程效率和可靠性。减少访问数据库的次数, 就能实际上减少ORACLE的工作量。

缺点：经常需要使用动态SQL的写法，由于参数的不同，可能SQL的执行效率不同；

1. 比较truncate和delete

答：truncate清空表内所有数据，保留表的结构，并且不能回滚

delete能用where指定要删的数据，并且可以回滚

1. fact table上要建什么索引；

答：位图索引（bitmap index）

1. 什么是scd，种类和区别

答：SLOWLY CHANGING DIMENSIONS，中文叫渐变维度，缓慢变化维，现实世界里维度的属性不是静态的，而是会缓慢变化的。

TYPE1：直接覆盖原值

TYPE2：添加新的维度行

TYPE3：添加属性列

OLTP与OLAP

答：OLTP：联机事务处理，用于传统关系型的数据库，需要实时处理大量请求，每次请求数据量都很小，用于基本的、日常的处理。

OLAP：联机分析处理，数据仓库的主要应用，在海量数据中得出统计和综合信息。

主题域

主题域通常是联系较为紧密的数据主题的集合。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。主题域的确定是由最终用户和数仓的设计人员共同完成的。

一、什么是水线(High Water Mark)?

所有的oracle段segments 都有一个在段内容纳数据的上限，我们把这个上限称为"high water mark"或HWM。这个HWM是一个标记，用来说明已经有多少没有使用的数据块分配给这个segment。HWM通常增长的幅度为一次5个数据块，原则上HWM只会增大，不会缩小，即使将表中的数据全部删除，HWM还是为原值，由于这个特点，使HWM很象一个水库的历史最高水位，这也就是HWM的原始含义，当然不能说一个水库没水了，就说该水库的历史最高水位为0。但是如果我们在表上使用了truncate命令，则该表的HWM会被重新置为0。

修正ORACLE表的高水位线

在ORACLE中，执行对表的删除操作不会降低该表的高水位线。而全表扫描将始终读取一个段(extent)中所有低于高水位线标记的块。如果在执行删除操作后不降低高水位线标记，则将导致查询语句的性能低下。下面的方法都可以降低高水位线标记。

1.执行表重建指令 alter table table\_name move;

(在线转移表空间ALTER TABLE 。。。 MOVE TABLESPACE 。。。ALTER TABLE 。。。 MOVE 后面不跟参数也行，不跟参数表还是在原来的表空间，move后记住重建索引。如果以后还要继续向这个表增加数据，没有必要move，只是释放出来的空间，只能这个表用，其他的表或者segment无法使用该空间)

2.执行alter table table\_name shrink space; 注意，此命令为Oracle 10g新增功能，再执行该指令之前必须允许行移动alter table table\_name enable row movement;

3.复制要保留的数据到临时表t，drop原表，然后rename临时表t为原表

4.emp/imp

5.alter table table\_name deallocate unused

6.尽量truncate吧

触发器

数据库触发器是一个与表相关联的，存储的PL/SQL 语句。

每当一个特定的数据操作语句（insert update delete）在指定的表上发出时，Oracle自动执行触发器中定义的语句序列。

其中：

BEFORE 和AFTER指出触发器的触发时序分别为前触发和后触发方式，前触发是在执行触发事件之前触发当前所创建的触发器，后触发是在执行触发事件之后触发当前所创建的触发器。

FOR EACH ROW选项说明触发器为行触发器。

行触发器和语句触发器的区别表现在：行触发器要求当一个DML语句操走影响数据库中的多行数据时，对于其中的每个数据行，只要它们符合触发约束条件，均激活一次触发器；而语句触发器将整个语句操作作为触发事件，当它符合约束条件时，激活一次触发器。

当省略FOR EACH ROW 选项时，BEFORE 和AFTER 触发器为语句触发器，而INSTEAD OF 触发器则只能为行触发器。

REFERENCING 子句说明相关名称，在行触发器的PL/SQL块和WHEN 子句中可以使用相关名称参照当前的新、旧列值，默认的相关名称分别为OLD和NEW。触发器的PL/SQL块中应用相关名称时，必须在它们之前加冒号(:)，但在WHEN子句中则不能加冒号。

WHEN 子句说明触发约束条件。Condition 为一个逻辑表达时，其中必须包含相关名称，而不能包含查询语句，也不能调用PL/SQL 函数。WHEN 子句指定的触发约束条件只能用在BEFORE 和AFTER 行触发器中，不能用在INSTEAD OF 行触发器和其它类型的触发器中。

当一个基表被修改( INSERT, UPDATE, DELETE)时要执行的存储过程，执行时根据其所依附的基表改动而自动触发，因此与应用程序无关，用数据库触发器可以保证数据的一致性和完整性。

每张表最多可建立12 种类型的触发器：

元数据

是关于数仓中数据的数据，类似与数据库管理系统的数据字典，保存了逻辑数据结构、文件、地址、索引等信息。

广义上，在数据仓库中，元数据描述了数仓内的数据的结构和建立方法的数据。

去重的方式有几种：

 1 DISTINCT 对查询结果去重

 2 GROUP BY 对分组字段去重 ，

 3 ROWID 伪列去重

 4 ROW\_NUMBER()OVER(ORDER BY XX) 分析函数去重

 5 UNION 集合运算去重

HADOOP三层架构和概念

1. 分布式存储系统HDFS（Hadoop Distributed File System）

2. 资源管理系统YARN

3. 分布式计算框架MapReduce

普通视图和物化视图的区别

物化视图是一种特殊的物理表，“物化”(Materialized)视图是相对普通视图而言的。

普通视图是虚拟表，应用的局限性大，任何对视图的查询，Oracle都实际上转换为视图SQL语句的查询。这样对整体查询性能的提高，并没有实质上的好处。

普通视图是不存储任何数据的，他只有定义，在查询中是转换为对应的定义SQL去查询，

而物化视图是将数据转换为一个表，实际存储着数据，这样查询数据，就不用关联一大堆表，如果表很大的话，会在临时表空间内做大量的操作。

Oracle分区表分为四类：

范围分区表；列表分区表；哈希分区表；组合分区表

Oracle有哪几种索引?

1.单列索引与复合索引

单列索引是基于单列所创建的索引，复合索引是基于两列或者多列所创建的索引

2.唯一索引与非唯一索引

唯一索引是索引列值不能重复的索引，非唯一索引是索引列可以重复的索引。都允许取 NULL 值，默认 Oracle 创建的索引是不唯一索引

1. B 树索引

B 树索引是按 B 树算法组织并存放索引数据的，B 树索引主要依赖其组织并存放索引数据的算法来实现快速检索功能

4.位图索引

它采用位图偏移方式来与表的行 ROWID 号对应，通过位图索引中的映射函数完成位到行的ROWID 的转换,主要用于节省空间，减少oracle对数据块的访问，采用位图索引一般是重复值太多、只有几个枚举值的表字段。

5.函数索引

Oracle 对包含列的函数或表达式创建的索引

你们公司如何管理存储过程？

答：按照包（自己包或者提交代码让项目经理来包）来管理。（1）

我们公司存储过程按照版本管理，以月为单位。（2）

我们公司存储过程按照分支，主干管理，我完成需求对应的分支部分，然后提交代码再并入主干。（3）

WHERE与HAVING的区别？

答：where是一个约束声明,使用where来约束来自数据库的数据;

where是在结果返回之前起作用的;

where中不能使用聚合函数。

having是一个过滤声明;

在查询返回结果集以后，对查询结果进行的过滤操作;

在having中可以使用聚合函数。

分析函数和聚合函数的区别？

答：简单计算用聚合函数，复杂计算用分析函数

聚合函数性能优于分析函数

聚合函数可以直接放在HAVING后作为条件，分析函数不可以。

SQL执行越来越慢？

答：先查日志，一般是索引问题，SQL性能不足，数据量越来越大导致效率越来越低，通过删除索引提升性能，最后通过更改执行计划来优化。

思考：

 存储过程 和 ETL过程的区别？

存储过程主要的功能是 同步数据 。ETL 主要的作用是 数据转换 。

 为什么不都用ETL工具实现数据同步？

ETL 工具贵，成本高(收费的工具价格比较高昂，免费的不够稳定) ，人工的成本相对比较低 。

 或者为什么不都用 SP 进行数据同步？

因为涉及到跨库的操作 通过代码实现过于繁琐，时间成本太高 。

数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策(Decision Making Support)。

数仓的特点 ：

 1 面试主题 。 业务主题 ，我们关注的指标和维度的集合 。

 2 数据是集成的 。

 3 反应历史变化的 。

 4 相对稳定的 。

在数仓中主要是做 数据分析 。跟数据库相比 数仓的时效性比数据库低很多，是高延迟的。而数据库是要求实时性的。

项目周期 ，一般是 1个月 ~ 1个半月 为一个版本 。也叫作版本周期 。

版本是什么？ 版本就是实现需求过程中的最小交付周期 。 1000个需求 分10个月交付，每个月就是一个版本 。

凡是说道版本 ，就一定要上线。否则就是版本内的迭代 。

在版本周期内我们分几个阶段 每个阶段分别做不同的事情：

 需求阶段 1周左右

 设计阶段 1周左右

 开发阶段

 测试阶段 2周左右

 上限阶段 测试完成前2天左右 。

 上线

 |\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

 需求分析阶段 设计开发阶段 测试验证阶段 上线阶段

 1周 1周 2周左右 上线演练

项目组有哪些人 ？

 产品经理 ：1 人 -- 管理需求 ，需求的唯一接口人

 项目经理 : 1 人 -- 管理开发的工作分工 开发进度 ，把控项目的开发进度和风险

 开发人员 ：

 ETL开发 >= 2 人

 数仓开发 >= 2 人

 报表开发 >= 2 人

 使用报表工具 >= 3 人

 --------------------------

 不用报表工具

 报表开发 >= 2

 JAVA >= 2 人

 前端开发 >= 2 人

 测试人员 ：

 人数 = 开发人数的 1/4 ~ 1/9 , >= 3

 -- QA 质量保证 ，这个岗位 可以管 需求 ，开发 ，测试 ；但是又没有 实权。

4 项目组的开发流程?

需求阶段

跟产品经理了解需求 。

然后根据项目经理分配的开任务 进一步分析如何实现需求，搞清楚 指标的计算逻辑 ，取数逻辑 ，字段映射关系。

维护数据字典 ，设计表结构 （一直持续到项目上线之前）

开发阶段

根据需求开发SP 或者 FUN 。

开发完成以后 ，做代码的UT测试(单元测试)。

在开发阶段的末期，项目经理组织进行代码评审，评审对象包括 我们的代码 和 数据字典。

测试阶段

根据测试提交的BUG进行分析，

如果是问题则定位BUG产生的原因然后修改BUG ，修改之后，将修改后的代码提交到SVN指定的路径下，再同时测试同时更新代码后验证。

如果不是问题，则跟测试同事沟通，关闭 问题单。

对沟通过程中的争议点，找产品经理最终拍板。

对于已经解决并验证通过的BUG，督促测试同事尽快关闭问题单。

对于因为修改BUG产生的新的BUG进行定位。注意 改BUG的时候 一定要尽量考虑的全貌一些，不要因为改一个点，带出一堆问题。

建议，当发现一个问题的时候，习惯性的进行错误推测，看下自己的其他代码是否有类似的问题，一起改了。避免测试给我们提过多的BUG，影响绩效（钱）。

上线阶段

测试阶段的末期，项目经理组织进行 上线演练/ 预发布。

上线当天（周五，周六），晚上项目组的全体成员参与 上线支持 。

在上线过程中，如果发现了自己的BUG，要及时定位并且快速解决，解决以后再到测试环境验证，通过后重新更新到生成环境 。

如果没自己名下的BUG， 可以帮其他同事定位问题。

项目组的需求是什么 (什么是需求 )？

需求 ： 需求规格说明书 ， 用户/客户有一些想法，

产品经理 通过跟客户进行沟通，把客户的 需求 转换成IT的可 实现方案 ，然后用文档记录下来，形成了 需求文档/需求规格说明书 ；

公司的领导/高层，想要看一些指标，由产品经理跟他们沟通，确认指标的计算逻辑/公式，然后把这些指标类的需求 交接给 开发 和 测试 。

BI 开发人员 不做可行性分析 ，我们一般会严格按照需求进行开发 。

 需求文档中包含的内容：

 项目背景

 项目的版本

 需求文档的阅读对象

 需求文档中 专业术语解释

 需求的指标

 需求的取数逻辑

 取值字段（目标表）

 上游系统

 指标展示样式（报表的样式）

 参考资料

对于需求中的内容，我们重点关注以下几点就足够了：

 1 版本

 2 指标

 3 取数字段

 4 上游系统（源表）

 5 取数逻辑（关联的逻辑 ，映射关系）

事实表和维度表之间，内关联还是外关联，关联字段 ，过滤条件 。

映射关系指源表 和 目标表 每个字段的对应关系，叫做 映射。

6 在项目组 如何进行 需求分析 ？

从产品经理那接到需求之后 ，先看一遍需求文档

确定版本之后， 先看指标是什么要计算什么，按什么维度算 ，

然后分析 要计算这个指标需要一张表中至少包含哪些列 ，

之后，分析 目标表的这些列 需要从 上游的哪些表中 取值 。

最后，确定每个字段的映射关系 。

从产品经理那接到需求以后我会先进行需求分析 ，搞清楚要计算什么指标

然后参加 产品经理 组织的需求评审会议 ，听产品经理给我们项目组的全体成员（项目经理，开发，测试 等等 ）

会后，根据项目经理给我安排的开发任务 ，进一步分析需求 搞清楚 取数逻辑 ，然后按照需求 开发 SP 或者 FUN 。

8 项目组BI的每一层的工作是如何进行的

 需求阶段 设计 开发 测试 上线

 ETL开发

 数仓开发

 报表开发

 上面的三种岗位，他们在同一个项目组是同步开发的。

 每一层开发都要建表，我们创建的是我们自己的 目标表。

 在项目组的开发环境是没有数据的，我们做UT测试的数据需要我们自己造。 使用 INSERT INTO / 循环 造数据 。

 UT测试的测试点：

 1 代码编译通过 。

 2 调用没问题。

 3 造数据验证 代码的每个逻辑分支 。

测试人员会写测试用例 来验证我们的代码功能

 BUG ： 开发代码未能按照需求的要求实现，这样的问题就是BUG。

 测试用例覆盖的测试点：

1 表

表的字段类型 必须跟需求保持一致。 特别是时间的字段 ，目标表中保存的时间 必须是 date 格式

 字段的长度 每个字段的长度必须大于等于 上游表的最大长度

 主键 必须按照公司的规范创建主键 ID\_表名

 索引 可以在表描述中看到 INDEX

 注释 COMMENT

 建表的回滚脚本

 同义词

 权限

2 SP

名称命名 ，必须跟项目组/公司的开发规范一致。

 参数个数 ，类型 ，按照需求实现。

 取数逻辑 ，关联方式，过滤条件

 代码的注释 ， 每个字段都要加注释，并且注释要正确， 比如 ：费率 和 费率% ； 每个变量 ，每段代码都要加注释。

 日志 ， 正常日志 ，异常的日志。

3 FUN

名称命名 ，必须跟项目组/公司的开发规范一致。

 参数个数 ，类型 类型 ，按照需求实现。

 计算逻辑/计算公式必须跟需求一致。

 计算结果，精度 ，四舍五入，保留小数 ，最后才做四舍五入，在计算的过程中不能进行四舍五入。

 统计口径 ，逻辑问题。

 统计范围 ，取数范围，关联的表的数量。

上线

 上线之前每个人要做一次脚本的检查

 脚本是否是最新的。

 脚本中是否存在 目标表的字段长度 小于 源表的最大长度。

 要跟项目经理/项目经理指定开发人员 汇报 本次上线 自己有多少个脚本，脚本名称，数量，新增脚本&回滚脚本。

 之后项目经理/项目经理指定开发人员会输出一个 上线脚本的清单(LIST)

 上线的时间一般是周末 ，周五晚上 ，周六 。

 上线会有一个 时间窗（上线变更的时间范围）。

 我们参与上线支持 ，如果有我们负责的需求的BUG，那么我们需要 及时定位并快速解决。

 生产环境的配置 ，统一说法，由甲方指定的同时负责，我们不负责生成环境的一切操作。

 如果生成环境产生BUG，要修复，必须先到测试环境验证通过以后，再更新到生产环境 。

 开发环境 ：开发人员使用的环境，只有代码，没有数据，数据需要开发自己造。

 测试环境 ：测试人员使用的环境，有测试数据 ，但是开发人员没有权限，我们开发可以申请权限，只能是查询权限。

 生产环境 ：上线以后 用户真正使用的环境，BOSS们用的环境，正式环境。