**1.exist和in的区别**

in 和 exists的区别: 如果子查询得出的结果集记录较少，主查询中的表较大且又有索引时应该用in, 反之如果外层的主查询记录较少，子查询中的表大，又有索引时使用exists。其实我们区分in和exists主要是造成了驱动顺序的改变(这是性能变化的关键)，如果是exists，那么以外层表为驱动表，先被访问，如果是IN，那么先执行子查询，所以我们会以驱动表的快速返回为目标，那么就会考虑到索引及结果集的关系了 ，另外IN时不对NULL进行处理。

　　　　in 是把外表和内表作hash 连接，而exists是对外表作loop循环，每次loop循环再对内表进行查询。一直以来认为exists比in效率高的说法是不准确的

**2.包，函数，存储过程的作用和联系**

**\*包**就是把相关的存储过程、函数、变量、常量和游标等 PL/SQL 程序组合在一起， 并赋予一定的管理功能的程序块。

一个程序包由两部分组成：包定义和包体。

在一个公司，会有N多的函数 和 存储过程，如果都是单个的展示，其不容易管理，

所以需要把相关的 函数 和 存储过程 通过 包 管理起来

一般在企业，是以项目为单位，一个项目建一个包，一个项目所有的函数和存储过程被这个包所管理

**\*存储过程**是由过程化[SQL语句](https://so.csdn.net/so/search?q=SQL%E8%AF%AD%E5%8F%A5&spm=1001.2101.3001.7020)书写的过程，这个过程经编译和优化后存储在数据库服务器中，因此称它为存储过程，使用时只要调用即可。

\***用户自定义函数**和存储过程一样，包含一系列SQL语句，不同之处在于函数必须指定返回类型。

**3. Oracle 降低高水位线的方法**

高水位（HIGH WARTER MARK，HWM）好比水库中储水的水位，用于描述数据库中段的扩展方式。高水位对全表扫描方式有着至关重要的影响。当使用DELETE删除表记录时，高水位并不会下降，随之导致的是全表扫描的实际开销并没有任何减少。

例如，首先新建一张空表，大小占用64K，然后插入数据直到表大小变为50G，此时使用DELETE删除所有的数据并且提交，这个时候查询表的大小的时候依然是50G，这就是因为表的高水位没有释放的缘故，而在这时如果使用“SELECT \* FROM TABLE\_NAME;”语句来查询数据的话，那么查询过程就会很慢，因为Oracle要执行全表扫描，从高水位下所有的块都得去扫描，直到50G的所有块全部扫描完毕。曾遇到一个同事使用DELETE删除了一个很大的分区表，然后执行SELECT查询很久都没有结果，以为是数据库HANG住了，其实这个问题是由于高水位的缘故。所以，表执行了TRUNCATE操作，再次SELECT的时候就可以很快返回结果了。

**释放表的高水位通常有如下几种办法**：

（1）对表进行MOVE操作：ALTER TABLE TABLE\_NAME MOVE;。若表上存在索引，则记得重建索引。

（2）对表进行SHRINK SPACE（压缩空间）操作：ALTER TABLE TABLE\_NAME SHRINK SPACE;，注意，在执行该指令之前必须开启行移动：ALTER TABLE TABLE\_NAME ENABLE ROW MOVEMENT;。该方法的优点是：在碎片整理结束后，表上相关的索引仍然有效，缺点是会产生大量的UNDO和REDO。

（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）exp/imp或expdp/impdp重构表。

（5）若表中没有数据则直接使用TRUNCATE来释放高水位。

如何找出系统中哪些表拥有高水位呢？这里给出两种办法，①比较表的行数和表的大小关系。如果行数为0，而表的当前占用大小减去初始化时的大小（INITIAL\_EXTENT）后依然很大，那么说明该表有高水位。②行数和块数的比率，即查看一个块可以存储多少行数据。如果一个块存储的行数少于5行甚至更少，那么说明有高水位。注意，这两种方法都不是十分准确，需要再对查询结果进行筛选。需要注意的是，在查询表的高水位

时，首先需要分析表，以得到最准确的统计信息。

**4.truncate和delectable区别**

**（1. TRUNCATE 是 DDL 命令，删除的数据不能恢复；DELETE 命令是 DML 命令，删除后的**

**数据可以通过日志文件恢复。**

**2. 如果一个表中数据记录很多，TRUNCATE 相对 DELETE 速度快。**

**由于 TRUNCATE 命令比较危险，因此在实际开发中，TRUNCATE 命令慎用。**

**再加上 drop 的对比：**

**1)在速度上，一般来说，drop> truncate > delete。**

**2)DELETE 删除的数据可以恢复；**

**3)如果想删除部分数据用 delete，注意带上 where 子句；**

**4)truncate 不需要提交或者回滚，delete 需要提交或者回滚）**

1、在功能上，truncate是清空一个表的内容，它相当于delete from table\_name。

2、delete是dml操作，truncate是ddl操作；因此，用delete删除整个表的数据时，会产生大量的roolback，占用很多的rollback segments， 而truncate不会。

3、在内存中，用delete删除数据，表空间中其被删除数据的表占用的空间还在，便于以后的使用，另外它是“假相”的删除，相当于windows中用delete删除数据是把数据放到回收站中，还可以恢复，当然如果这个时候重新启动系统（OS或者RDBMS），它也就不能恢复了！

而用truncate清除数据，内存中表空间中其被删除数据的表占用的空间会被立即释放，相当于windows中用shift+delete删除数据，不能够恢复！

4、truncate 调整high water mark 而delete不；truncate之后，TABLE的HWM退回到 INITIAL和NEXT的位置（默认）delete 则不可以。

5、truncate 只能对TABLE，delete 可以是table,view,synonym。

6、TRUNCATE TABLE 的对象必须是本模式下的，或者有drop any table的权限 而 DELETE 则是对象必须是本模式下的，或被授予 DELETE ON SCHEMA.TABLE 或DELETE ANY TABLE的权限。

7、在外层中，truncate或者delete后，其占用的空间都将释放。

8、truncate和delete只删除数据，而drop则删除整个表（结构和数据）。

9.

1.TRUNCATE TABLE是非常快的

　　2.TRUNCATE之后的自增字段从头开始计数了，而DELETE的仍保留原来的最大数值

　　TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。

DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

　　TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。

对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的 DELETE 语句。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器。

TRUNCATE TABLE 不能用于参与了索引视图的表。

**5.总账科目**

总账科目是 总帐中根据各个明细科目汇总登记，使用的 会计科目分类。比如招待客人吃饭用的钱.总账科目为 管理费用，明细科目为业务招待费。

总账科目为一级科目，也就是现金、银行存款、 原材料、 产成品、 其他应收款、 其他应付款、 应付账款、 应交税金、 本年利润、管理费用、 销售费用、 财务费用、 利润分配、生产成本、 制造费用....

明细科目为二级科目，也就是对一级科目再进行细分，比方说银行存款的二级科目可以为\*\*工行、\*\*建行、\*\*农行等等

**6.调用需要做哪些准备**

**7.什么情况下会导致数据重复**

**8.分区表索引，全局索引，局部索引**

\*分区索引

就是在所有每个区上单独创建索引，它能自动维护，在drop或truncate某个分区时不影响该索引的其他分区索引的使用，也就是索引不会失效，维护起来比较方便，但是在查询性能稍微有点影响。

\*全局索引global index

1. 全局索引的分区键和分区数和表的分区键和分区数可能都不相同，表和全局索引的分区机制不一样。

2. 全局索引可以分区，也可以是不分区索引，全局索引必须是前缀索引，即全局索引的索引列必须是以索引分区键作为其前几列。

3. 全局分区索引的索引条目可能指向若干个分区，因此，对于全局分区索引，即使只动，截断一个分区中的数据，都需要rebulid若干个分区甚

至是整个索引。

4. 全局索引多应用于oltp系统中。

5. 全局分区索引只按范围或者散列hash分区，hash分区是10g以后才支持。

6. oracle9i以后对分区表做move或者truncate的时可以用update global indexes语句来同步更新全局分区索引，用消耗一定资源来换取高度的可用性。

7. 表用a列作分区，索引用b做局部分区索引，若where条件中用b来查询，那么oracle会扫描所有的表和索引的分区，成本会比分区更高，此时可以考虑用b做全局分区索引

\*局部索引local index

1. 局部索引一定是分区索引，分区键等同于表的分区键，分区数等同于表的分区说，一句话，局部索引的分区机制和表的分区机制一样。

2. 如果局部索引的索引列以分区键开头，则称为前缀局部索引。

3. 如果局部索引的列不是以分区键开头，或者不包含分区键列，则称为非前缀索引。

4. 前缀和非前缀索引都可以支持索引分区消除，前提是查询的条件中包含索引分区键。

5. 局部索引只支持分区内的唯一性，无法支持表上的唯一性，因此如果要用局部索引去给表做唯一性约束，则约束中必须要包括分区键列。

6. 局部分区索引是对单个分区的，每个分区索引只指向一个表分区，全局索引则不然，一个分区索引能指向n个表分区，同时，一个表分区，也可能指向n个索引分区，

对分区表中的某个分区做truncate或者move，shrink等，可能会影响到n个全局索引分区，正因为这点，局部分区索引具有更高的可用性。

7. 位图索引只能为局部分区索引。

8. 局部索引多应用于数据仓库环境中。

**9.拉链表查询某一特定日期的切片数据**

****

****

****

**10.事务的作用**

\*事务是什么？

实际的SQL执行过程中，有时候我们生产过程需要保证某些SQL要么一起并顺序执行成功，要么一起失败回滚，不做任何操作。那么就有了事务的概念。事务是数据库中保证交易可靠的机制。

\*事务的作用是保证数据的一致性、完整性。

比如银行转账过程，都是简单分三步：第一步查询付款人的账户的余额是否足够扣的金额；第二步：如果余额足够，扣付款人账户的款项金额；第三步：对收款人账户，余额加上转账的款项金额，完成付款。那么三步操作只一或两步操作成功，不是全部成功且按顺序成功的话，就可能出现付款人账户不足却扣款到0甚至负数，付款人扣款了但收款人没收到，等等多种不可遇见的异常情况。而这些异常，生产是不能够接受的。那么如果有了事务就能保证一致成功或失败，数据是完整提交或回滚。

**11.Hadoop介绍**

Hadoop的核心为HDFS（分布式文件系统）和MapReduce（离线计算框架）。简单来说，HDFS就是将要储存的文件分散在不同的硬盘上，并记录他们的位置，而MapReduce就是将计算任务分配给多个计算单元，下面针对这两个核心再做进一步的说明。

HDFS

HDFS已经成为现在大数据的储存标准，他适合储存一次写入，多次读取的数据，并且他有自动检错、快速回复数据的功能，能够避免数据因为硬盘损坏而丢失的危险。

HDFS是由DataNode和NameNode组成的，DataNode负责储存数据，而NameNode负责管理数据，一个NameNode对应多个DataNode，NameNode记录着每个DataNode储存的数据内容，并曝露给上层系统调用，也会根据上层的指令对DataNode进行增、删、复制。

MapReduce

MapReduce是将计算任务分配给数据就近的处理节点，进行完运算后再合并导入结果，能很好的去进行大量数据的调取，但是延时较高，不适合处理实时流数据。

MapReduce可以分为Map和Reduce两个处理步骤。首先Map将用户输入的指令解析出一个个的Key/Value，然后再将转化成一组新的KV值，将原本的任务拆解成小的而且是临近数据的，并且确保这些运算任务彼此不会影响。而Reduce则是将这些运算的结果汇总起来，将结果写入。

另外YARN和Zookeepr都是用来管理的，YARN是面对计算资源的管理，而Zookeeper是面对服务器集群的管理。

YARN：资源管理框架，用来管理和调度CPU、内存的资源，避免所有的计算资源被某些任务大量占用，有点像是云管理平台可以创造不同的容器和虚拟机，并将这些硬件资源按用户的意愿分配给计算任务。

Zookeeper：集用来做群管理，跟微服务里的功能相似，可以在集群里面选出一个leader，并保证集群里面服务器的一致性、可靠性和实时性。

**11、Hadoop常用组件介绍**

1、Hive

Hive是将Hadoop包装成使用简单的软件，用户可以用比较熟悉的SQL语言来调取数据，也就是说，Hive其实就是将Hadoop包装成MySQL。Hive适合使用在对实时性要求不高的结构化数据处理。像是每天、每周用户的登录次数、登录时间统计；每周用户增长比例之类的BI应用。

2、HBase

HBase是用来储存和查询非结构化和半结构化数据的工具，利用row key的方式来访问数据。HBase适合处理大量的非结构化数据，例如图片、音频、视频等，在训练机器学习时，可以快速的透过标签将相对应的数据全部调出。

3、Storm

前面两个都是用来处理非实时的数据，对于某些讲求高实时性（毫秒级）的应用，就需要使用Storm。Storm也是具有容错和分布式计算的特性，架构为master-slave，可横向扩充多节点进行处理，每个节点每秒可以处理上百万条记录。可用在金融领域的风控上。

4、Impala

Impala和Hive的相似度很高，最大的不同是Impala使用了基于MPP的SQL查询，实时性比MapReduce好很多，但是无法像Hive一样可以处理大量的数据。Impala提供了快速轻量查询的功能，方便开发人员快速的查询新产生的数据。

**12. 拿到一个新的需求，你该怎么做**

评估这个需求 确定需求是不是明确 有没有重复 所需的数据模型是否存在 是可以直接配置指标 还是需要建立新的数据模型

+项目流程