1. 不是很有把握的事情，回答：那块不是我主要负责的，但我有了解
2. 总之别跟面试官犟，不太有把握的就反过来请教他们，让他们显摆，估计他们自己也不太懂
3. 我之前没遇到过这种情况
4. 拉链表 缓慢变化维

保存历史数据的方法 （在ODS 层）

TYPE1:不保存历史数据，直接更新 — 直接覆盖原值

TYPE2:保存所有的历史数据 ————添加维度行 代理主键

TYPE3:保存当前或者上一次的历史数据————添加属性列

1. Oracle 提供了以下几种分区类型：

范围分区（range） 最为常用，分区键经常采用日期 变化量大的选择范围分区，

哈希分区（hash） 变化量小的选择哈希分区

列表分区（list）

范围－哈希复合分区（range-hash）

范围－列表复合分区（range-list）

1. Case when 与 decode的区别

DECODE 只有Oracle 才有，其它数据库不支持;
      2.CASE WHEN的用法， Oracle、SQL Server、 MySQL 都支持;
      3.DECODE 只能用做相等判断,但是可以配合sign函数进行大于，小于，等于的判断,CASE when可用于=,>=,<,<=,<>,is null,is not null 等的判断;
      4.DECODE 使用其来比较简洁，CASE 虽然复杂但更为灵活;
      5.另外，在decode中，null和null是相等的，但在case when中，只能用is null来判断，示例如下：
      emp表中有一列comm，如果这列为null,则显示为0，否则，显示为原值

1. 存储过程的异常处理

操作过程中遇到异常怎么处理

BEGIN

 --可执行部分

 EXCEPTION -- 异常处理开始

 WHEN 异常名1 THEN

 --对应异常处理

 WHEN 异常名2 THEN

 --对应异常处理

 ……

 WHEN OTHERS THEN

 --其他异常处理

 END;

|  |  |  |
| --- | --- | --- |
| DUP\_VAL\_ON\_INDEX | ORA-00001 | 试图向唯一索引列插入重复值 |
| INVALID\_CURSOR | ORA-01001 | 试图进行非法游标操作 |
| INVALID\_NUMBER | ORA-01722 | 试图将字符串转换为数字 |
| NO\_DATA\_FOUND | ORA-01403 | SELECT INTO 语句中没有返回任何记录 |
| TOO\_MANY\_ROWS | ORA-01422 | SELECT INTO 语句中返回多于 1 条记录 |
| ZERO\_DIVIDE | ORA-01476 | 试图用 0 作为除数 |
| CURSOR\_ALREADY\_OPEN | ORA-06511 | 试图打开一个已经打开的游标 |

1. 存储过程规范

变量 变量V\_游标\_CUR

游标

包

事务处理

数据封装

数据访问

日志书写

错误处理

书写

书写优化

1. 需求分析

根据需求文档从取值字段、取值逻辑、映射关系、版本、源表进行需求分析，编写数据宝典

1. 去重

Distinct and rowid union两个结果集可以达到去重的效果

delete from emp e

where rowid not in

(select min(rowid) from emp f where e.empno = f.empno);

1. 对SQL语句怎么进行性能优化

一般根据sql语句的执行计划判断优化的方向，主要是从扫描方式和关联机制两个方面入手，根据sql语句的执行情况通过创建索引，改变表的关联方式等情况进行优化；

1. 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，如果能用关联就用关联，关联性能比in和exist都好，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）

存储过程

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。

1. 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。

1. 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。

1. 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。

表的数据上亿查询非常慢，怎么弄 拆表union all

分区表优化

游标的种类

显示游标和隐式游标

隐式游标需要手动打开吗

显示游标需要手动打开

Oracle调优，从哪些方式去调优

索引 唯一、组合、位图、函数索引

几种索引，建了索引，关联条件都用了索引，为什么在查看执行计划的时候，为什么没有走这个索引，有哪些情况？

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。

查看执行计划的时候，会看到表的连接方式，有哪些连接方式？具体怎么连的，具体的案例

嵌套循环、哈希连接、（归并）排序合并连接

优化会提到表分区优化：工作当中常用的有哪些表分区

1.范围分区：RANGE 最为常用的，并且分区键经常采用日期

2.列表分区：LIST

3.散列（哈希）分区：HASH

4.组合分区

1. 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

Er建模

三范式以及具体的概念

三范式

第一范式（1NF）：所有字段值都是不可分解的原子值，即不能同行同列出现两个值。

第二范式（2NF）：满足第一范式为前提，一定要有主属性键，且每一列都和主键相关，而不能只与主键的某一部分相关。

第三范式（3NF）：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。

物化视图？

Ctm调度工具（配置）

PowerBI tableau finereport

row\_number

Rank

Dense rank

行转列 case when ipvout

多少个业务规模去接入这些数据

Linux命令都用来干啥

HINTS 强制执行索引和并行

指向查询结果的一个指针

分为 隐式游标 和 显式游标

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)