Oracle实践

###### 一、空值判断，空值转换

使用is null、is not null，判断是否是空值。使用nvl(exp1,exp2)将空值转换为其他值。

exp1表示可能为空的字段，exp2表示转换成什么值。

###### 二、列转行 pivot

语法：select \* from table pivot (聚合函数 for column in (‘常量值1’,’常量值2’));

相当于对column字段进行分组处理，聚合函数为需要统计的字段值。

|  |
| --- |
| select \* from (select \* from t\_merchant\_product\_trade\_summary where mcht\_id = '366521058121455')  pivot (sum(amount\_summ) for trade\_type in ('微信支付' wx,'支付宝支付' ali)); |

###### 三、行转列 unpivot 待定

语法：

###### 四、将某一列的值进行字符创拼接输出 listagg

语法：listagg(column,'分隔符') within group(order by 排序字段)

查询商户下所有的分店id，并且分店id已‘；’来分割

|  |
| --- |
| select mcht\_id,listagg(branch\_id,';')within group(order by mcht\_id) from t\_branch\_info where mcht\_id='821330156910869'  group by mcht\_id; |

###### 五、批量插入

Insert all

into table () values ()

into table () values ()

…

select 1 from dual;

###### 六、NUMBER的使用

NUMBER(precision,scale)的说明

presion表示从左边第一个不为0的有效位最大为precision，如果没有指定默认是38位

scale表示小数位最多为scale，正数表示精确到小数点右边位数，负数表示精确到小数点左边位数，默认是0

s   >   0 ，精确到小数点右边s位，并四舍五入。若s>p，小数点右侧至少有s-p个0填充（必须从小数点处开始并连续），然后检验有效位是否   <=   p。

s   <   0 ，精确到小数点左边s位，并四舍五入。然后检验有效位是否   <=   p   +   |s|。

s   =   0 ，此时NUMBER表示整数。

NUMBER：等价于double，对小数位没有明确要求的情况下使用。

NUMBER(2): 最大两位整数，定义整数使用。

NUMBER(7,2): 有效位数7位，小数位2位。用于指定小数位数的情况。

###### 七、对于查询的列的说明

将查询的列看做是一个变量，可以查询原始列，也可以对列进行运算，对列使用函数等操作

###### 八、distinct去除重复行

Distinct 作用其后所有的列，group by 也是作用于其后所有的列。

###### 九、常用函数

字符函数：

upper,lower,concat ,length

字符串截取函数substr(‘hello’, 1, 3):表示从第一个字符开始截取，一共截取3个字符。hel

替换函数replace(‘aba123’,’a’,’A’):替换字符串中出现的字符或字符串AbA123

去空格函数trim(‘ ab ’):去除首尾空格

数值函数

日期函数

last\_day(date):获取当前月的最后一天

add\_months(date, int):对日期date加或者减多少个月

转换函数:

to\_char,to\_date,to\_number

select to\_char(sysdate, 'yyyy-MM-dd hh:mi:ss') from dual;

select to\_date('2019-05-07 10:57:12','yyyy-MM-dd hh:mi:ss') from dual;

select to\_number('1234') + to\_number('4321') from dual;

通用函数

nvl,decode,case when then end,trunc;

nvl(exp1,exp2):对exp1进行控制判断，如果为空取exp2的值，否则去exp1的值。

decode(val,if1,then1,if2,then2,else):当val值为if1时返回then1,条件都不满足时返回else值。

select decode(8, 1, 2, 2,3, 4) from dual; 值为4

模式一：case val when a then A when b then B else C end:当val的值为a时返回A,为b时返回B,都不满足时返回C。

select case 8 when 1 then 2 when 2 then 3 else 4 end from dual; 值为4

模式二：case搜索函数：

case when sex = '1' then '男'

when sex = '2' then '女'

else '其他' end

trunc(val, digit):截取函数，不进行四舍五入。val表示要截取的值，digit表示保留的精度

###### 十、分组函数

count,avg,max,min,sum

分组函数会省略列中的空值，可用nvl函数强制对空值进行处理

如：select avg(nvl(comm,0)) from emp;

###### 十一、group by 子句

group by 作用于其后的所有字段，即可对多个字段进行分组，分组的顺序按字段出现的顺序。当使用group by 子句后，select 后的字段必须是分组字段或者是分组函数。

需要注意的是如果没有group by 子句那么不允许列和分组函数混合使用

###### 十二、having 子句

一般用于分组后的过滤

###### 十三、多表查询连接方式

1.笛卡尔集 结果集数量为count(A)\*count(B)

Select \* from A,B;

2.等值连接（内连接）同时满足连接条件才显示

Select \* from A，B where A.id=B.aid; 等价于

Select \* from A inner join B on A.id=B.aid;

3.自连接 同表做关联

Select \* from A a1 inner join A a2 on a1.pid=a2.pid;

4.左连接 以左表（A）为主表，左边记录都显示，右表记录满足连接条件显示

Select \* from A left join B on a.id=B.aid;

5.右连接 以右表（B）为主表，右表记录都显示，左表记录满足连接条件显示

Select \* from A right join B on a.id=B.aid;

6.全连接 结合了左连接和右连接，等价于左连接和右连接的并集union

Select \* from A full join B on A.id=B.aid; 等价于

Select \* from A left join B on a.id=B.aid;

Union

Select \* from A right join B on a.id=B.aid;

###### 十四、集合操作

集合操作是对查出来的多张表进行的处理。前提是这多张表具体相同的结构（字段）

1.union 并集，自动去重

2.union all 并集 不去重

3.intersect 交集 只显示重复的内容

4.minus 差集 只显示对方没有的（和顺序相关）

###### 十五、子查询

1.单行子查询 子查询返回的一行记录并且是一个值

select \* from emp

where sal > (select sal from emp where empno = 7566);

2.多行子查询 表示返回多条记录，此时要使用到关键字in,any(返回记录的任意一个)，all(返回记录的所有)

3.topN查询 类似于最多最少的场景。根据查询条件查询前N条记录，使用到了排序+rownum

select \* from emp where rownum<=5 order by sal desc

4.分页查询

常用：select \* from (select rownum rn,t.\* from t\_merchant\_info t where rownum <= 30)

where rn > 20;

方式二：select \* from (select rownum rn,t.\* from t\_merchant\_info t)

where rn > 20 and rn <= 30;

5.exists关键字 使用语法：where exsits（相关子查询）

相关子查询：指的是子查询查询条件依赖于外部父查询，因此需要根据外部查询的每一条记录来进行子查询。当子查询返回结果为真，那么这条记录放入结果集中。

Exsits关键字后的子查询只返回逻辑值true或false。子查询有结果集返回true,无结果集返回false。

not exists 和exists功能相反。

exists和in的区别：

exists后面接相关子查询。in后面接不相关子查询（子查询不依赖于父查询）.

exsits是外层表驱动内层表，适用于外层表较小的场景。in 是内层表驱动外层表，当子查询结果集较小时使用in。

###### 十六、高级查询

###### 十七、oracle数据类型

Char固定字符串，长度不可变，速度比varchar2快。

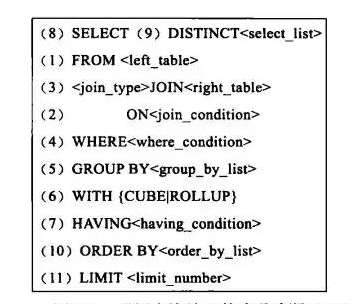
Varchar2可变字符串，长度根据实际长度调整，用的最多。

Number:见上述解释

Date：时间精确到秒

Timestamp：时间精确到秒后6位

###### 十八、sql执行顺序



###### 十九、DDL语言

1.创建表

Create table student(

Sid number(10),

Sname varchar2(10)

);

使用子查询创建表：

create table myemp as select \* from emp;

2.修改表

Rename student to student1;

3.删除表

Drop table student;

4.添加字段

Alter table student add age number(5);

5.修改字段

Alter table student modify age number(10);

Alter table table2 rename column result to result2;

6.删除字段

Alter table student drop column age;

###### 二十、DML语句

Insert、update、delete

Insert表间拷贝数据：

Insert into dept1(id,name) select deptno,dname from dept;

###### 二十一、约束

约束就是对插入数据的各种限制。

Not null：非空约束

Unique: 唯一约束（空值除外）

Primary key: 主键约束 唯一非空

Check: 条件约束，插入的条件必须满足给定的条件

如：age NUMBER CHECK(age BETWEEN 0 AND 150)

Foreign key: 外键约束

ON DELETE CASCADE：级联删除 指的是删除主表时，关联的外键的记录也被删除了。

一般来说，在创建表时就添加约束。

创建表后添加约束：

以上两张表中没有任何约束，下面使用 alter命令为表添加约束

例子：person和book关系时一对多

1、 为两个表添加主键：

person表pid为主键：

ALTER TABLE person ADD CONSTRAINT person\_pid\_pk PRIMARY KEY(pid) ;

book表bid为主键：

ALTER TABLE book ADD CONSTRAINT book\_bid\_pk PRIMARY KEY(bid) ;

2、 为person表中的tel添加唯一约束：

ALTER TABLE person ADD CONSTRAINT person\_tel\_uk UNIQUE(tel) ;

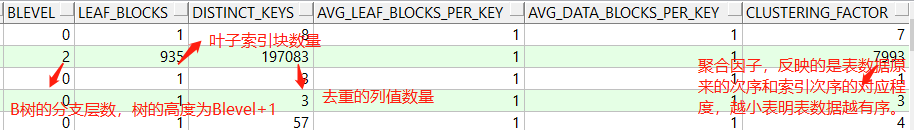
3、 为person表中的age添加检查约束：

ALTER TABLE person ADD CONSTRAINT person\_age\_ck CHECK(age BETWEEN 0 AND 150) ;

4、 为book表中的pid添加与person的主-外键约束，要求带级联删除

ALTER TABLE book ADD CONSTRAINT person\_book\_pid\_fk FOREIGN KEY (pid) REFERENCES person(pid) ON DELETE CASCADE ;

###### 二十二、索引



索引的创建：create [unique] index index\_name on table\_name (column);

索引的删除：drop index index\_name;

查看全部索引：select \* from user\_indexes;

唯一索引：索引列值可以为空，但只要有值则必须唯一。

非唯一索引：即普通所以，对唯一性没有要求。

两者区别：执行计划的差异。通过唯一索引查找记录时使用的是index unique scan操作，直接定位到相应的叶子数据块，返回rowid。通过普通索引查找记录时，使用的是index range scan操作，沿着索引树叶子结点水平扫描，返回符合条件的索引值的rowid列表。

Oracel默认的索引实现原理：B树索引，B树是一棵平衡树，每个结点的左子树和右子树的高度差不超过1。

B树索引的创建和查找

创建是从下往上，查找是从上往下。

1.要建索引先排序 ，所以索引其实是有序的，对索引列排序。

2.列值入块成索引，索引块就是这么来的。

3.填满一块接一块，数据量增大在索引的体现，就是叶子块横向扩展。

4.同级两块需人管，叶子块有2个或以上时候，就需要有一个老大来管理多个叶子块，这个老大块里放的是每个叶子块的指针，所以索引的高度不容易变得特别大也是这个道理。

B树索引的特点：

索引高度较低：50w数据量的查询效率和500w的可能是一样的。

索引存储列值：对聚合函数使用索引，提升性能。避免回表，提升性能。

所有本身是有序的：能够消除排序如order by

Blevel：branch level 分支层数，即不包含叶子结点层。千万级的数据blevel也就2层，所以通过索引查找数据是非常快的。

理想状态下，一个高度为3的主键索引B+tree树，至少可以存多少数据呢？（个人理解）

oracle：假设主键为BIGINT，占8个字节，rowid占8个字节，一个指针通常也为4或8个字节，那一个块为8k，数据即 8k/16b = 500条，一个2级的块就能存500个指针，总数就是500\*500\*500 = 1.25亿条数据。

组合索引：索引排序（第一列先排序，第二列后排序）

在等值查询情况下，组合索引的列无论哪列在前，性能都一样。

组合索引的两列，当一列是范围查询，一列是等值查询时，等值查询在前，范围查询在后更高效。

创建索引：创建索引有排序和锁表的动作，切记在业务高峰期操作。

优劣：提升查询效率；插入，删除，更新变慢。

###### 二十三、SQL优化

合理使用索引

最左前缀，利用组合索引的前置列查询

当业务需要返回列极少时，避免使用select \*，消除TABLE ACCESS BY INDEX ROWID回表动作改进性能，直接返回索引字段。如果字段不够可以考虑组合索引，一般不超过3个。如果一定有回表动作，聚合因子越小的索引性能越高。（不用跨多个数据块去匹配数据），如果需要优化聚合因子，需要重组表记录。

表关联的时候使用小表连接大表

使用exist代替in

优化表字段数量，提升全表扫描性能

统计时使用聚合函数，对列合理设置索引能提升性能

Union all 通常比union更高效

###### 二十四、执行计划

<https://coderbee.net/index.php/db/20150503/1224>

<https://blog.csdn.net/u010692693/article/details/102794395>

<https://www.cnblogs.com/littlewu/p/8630366.html>

执行计划可以看做是一条SQL语句的执行过程，对于SQL的语句的优化一般从看懂执行计划开始。通过创建索引或者修改SQL语句优化语句。一般cost越小，sql性能越好，但最终应该以测试结果为准。

1. 执行顺序的原则：从右到左，从上到下

Sql语句+F5 或者

Explain plan for sql语句;

Select \* from table(dbms\_xplan.display);

2.表的访问方式：

TABLE ACCESS FULL（全表扫描）

TABLE ACCESS BY ROWID（通过ROWID的表存取）

TABLE ACCESS BY INDEX SCAN（索引扫描）

3.索引的扫描方式：CBO(Cost-base optimization)

INDEX UNIQUE SCAN：索引唯一扫描，出现在索引的“所有列”上使用“等于”条件的 SQL 中，最多返回一行结果。

INDEX RANGE SCAN：索引范围扫描

在唯一索引列上使用了范围操作符（如：> < <> >= <= between）

在组合索引上，只使用部分列进行查询（查询时必须包含前导列，否则会走全表扫描）

对非唯一索引列上进行的任何查询

INDEX FULL SCAN：索引全扫描，按顺序扫描索引中的所有叶节点块。可以从左往右扫描，也可以从右往左扫描。是有序的

INDEX FULL SCAN(MAX/MIN)：使用max/min函数时可以走索引全扫描，因为索引时有序的，所有只需要看叶子块的最右边和最左边就行。

INDEX FAST FULL SCAN：索引快速全扫描，并行读取多个数据块进行扫描，扫描得到的结果是无序的。使用count(\*)/avg/sum聚合函数可以走索引快速全扫描，因为无需有序，需要保证索引的列不为空。

INDEX SKIP SCAN：索引跳跃扫描，出现在复合索引中，对非前缀列上使用任意比较条件的 SQL 中。Oracle 在扫描时，会扫描所有的 branch block，然后对 branch block 中的每一个值，都会去扫描 leaf block。适合于复合索引的前缀列不同值个数很少的情况，否则其扫描成本非常高。场景少

3.表的连接类型：

内连接INNER JOIN

外连接OUTER JOIN(LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN)

4.表的连接方式：

区分哪张是驱动表，哪张是匹配表，一般驱动表为小表，匹配表为大表。

就是两个表获得满足条件的数据时的连接过程。主要有三种表连接方式，嵌套循环（NESTED LOOPS）、哈希连接（HASH JOIN）和排序-合并连接（SORT MERGE JOIN）

5. 说明：

对于表的连接顺序，多数情况下使用的是嵌套循环，尤其是在索引可用性好的情况下，使用嵌套循环式最好的，但当ORACLE发现需要访问的数据表较大，索引的成本较高或者没有合适的索引可用时，会考虑使用哈希连接，以提高效率。排序合并连接的性能最差，但在存在排序需求，或者存在非等值连接无法使用哈希连接的情况下，排序合并的效率，也可能比哈希连接或嵌套循环要好。

###### 二十五、存储过程

|  |
| --- |
| CREATE OR REPLACE PROCEDURE gwj\_test  IS  BEGIN  for i in 1 ..164  loop  insert into t\_gwj\_stall\_reservation (AREA\_ID, AREA, STALL, STATUS, RENT\_START\_TIME, RENT\_END\_TIME, PRICE, IS\_LOCK, MONTH\_PRICE, QUARTER\_PRICE)  values (2, '衣裳街B区', '红' || i, '0', null, null, '3', null, '11', '32');  end loop;  commit;  END; |

调用：call gwj\_test();

###### 二十六、取分组后的最值

ROW\_NUMBER() OVER(PARTITION BY 列A ORDER BY 列B DESC) rn：对表中某个字段进行分组，并且根据某个字段排序，然后去每个分组的第一条数据。

PARTITION BY index\_etc：表示对index\_etc分组

ORDER BY trade\_time DESC：表示对trade\_time降序排列

WHERE rn = 1：表示去分组后的第一条数据

|  |
| --- |
| select \*  from (select ROW\_NUMBER() OVER(PARTITION BY index\_etc ORDER BY trade\_time DESC)  rn,t\_untp\_trade\_list\_deduction.\*  from t\_untp\_trade\_list\_deduction)  where rn = 1; |

**二十七、查看表占用的存储空间**

segment\_name必须要大写！

|  |
| --- |
| select t.owner,t.segment\_name, t.segment\_type, sum(t.bytes / 1024 / 1024) "占用空间(M)"  from dba\_segments t  where t.segment\_type='TABLE'  and t.segment\_name='T\_AGGREGATE\_PAYMENT\_ORDER'  group by OWNER, t.segment\_name, t.segment\_type; |



**二十七、数据库迁移**

使用exp/imp命令完成。在用户oracle中完成

dump文件中包含了表中的数据和表结构信息。

1. 登录到linux服务器，切换用户su – oracle。

注意符号-，否则无法使用oracle配置的环境变量，导致exp/imp命令不可用，需要进入到bin目录。

1. 将表数据导出为dmp文件。

按表导入：exp username/password@实例名file=dmp输出路径 tables=’table1,table2’

exp zjallinpaytest/zjallinpaytest@ora11g file=/project/daochu.dmp tables='t\_aggregate\_merchant\_info,t\_aggregate\_refund\_order'

按用户导入：exp username/password@实例名file=dmp输出路径 owner=username

完全导出：exp username/password@实例名file=dmp输出路径 full=y 这种需要username为dba权限。

1. 在目标数据库中创建用户。

登录到linux服务器，切换用户su – oracle

以管理员登录：sqlplus / as sysdba

创建用户：CREATE USER c##zjjtdev IDENTIFIED BY Asd\_13579; 需要以c##或C##开头，默认使用的表空间为users

赋予dba权限：GRANT CONNECT,RESOURCE,DBA TO c##zjjtdev;

用新用户登录：sqlplus c##zjjtdev/Asd\_13579

4．导入dmp文件。imp username/password@servicename file=dmp路径

imp c##zjjtdev/Asd\_13579@orcl file=/project/prepay/daochu.dmp ignore=y full=y

全部导入：full=y

igore=y: 如果表结构存在则不创建表结构，如果表结构不存在则创建表结构