ORACLE-TDSQL
兼容性常见问题

目录

1 常见问题

1.1 表字段总长度限制

TDSQL 表字段总长度不能超过 16K

1.2 表索引长度限制

表索引总长度不超过 3072 字节，否则失败，无法创建

1.3 TDSQL 和 Oracle 兼容性

1.3.1 TDSQL 不支持 Oracle 语句中存在 Rownum 写法

可改为 limit 替代

样例：

原 oracle 语句：

```
SELECT

*

FROM

ktlp_gycshu

WHERE

guiylxin = #guiylxin#  and farendma = #farendma# and rownum<4;
```

Rownum<4 改为 limit 3

limit 用法：

SELECT * FROM 表名 limit m,n;

SELECT * FROM table LIMIT [offset,] rows;

m 代表从 m+1 条记录行开始检索，n 代表取出 n 条数据。(m 可设为 0)

eg：

// 检索记录行 6-15

1、SELECT * FROM table LIMIT 5,10;

//为了检索从某一个偏移量到记录集的结束所有的记录行,可以指定第二个参数为 -1：

2、SELECT * FROM table LIMIT 95,-1; // 检索记录行 96-last.

//如果只给定一个参数，它表示返回最大的记录行数目：

3、SELECT * FROM table LIMIT 5; //检索前 5 个记录行，LIMIT n 等价于 LIMIT 0,n。

现 TDSQL 语句：

```
SELECT
*
FROM
ktlp_gycshu
WHERE
guiylxin = #guiylxin#  and farendma = #farendma#  limit 3;//检索前三条记录
```

limit 如何遇到排序 order by 的话放在 order by 后面

```
SELECT
*
FROM
```

```
articles

WHERE

category_id = 123

ORDER BY

id

LIMIT 0,10;
```

1.3.2 TDSQL 不支持 Oracle 语句中的 nvl 语句

使用 IFNULL 代替

原 oracle 语句：

```
select nvl(max(xuhaoooo),0)  as xuhaoooo from kcfb_cfdslx where

farendma=#farendma# and kehuhaoo =#kehuhaoo#
```

Oracle 中 nvl 替换为 TDSQL 中 ifnull

IFNULL 函数用法(expr1,expr2)

如果 expr1 不是 NULL，IFNULL()返回 expr1，否则它返回 expr2

现 TDSQL 语句：

```
select IFNULL(MAX(xuhaoooo),0) as xuhaoooo from kcfb_cfdslx where

farendma=#farendma# and kehuhaoo =#kehuhaoo#
```

1.3.3 TDSQL 不支持 Oracle 语句中序列查询

原序列查询：

```
select kapp_guiyls_seq.nextval from dual
```

可暂时改为平台提供的序号参数生成规则生成序号

调用如下：

```
IoApSeqGenStru cplIoSeqGenStru =
SysUtil.getInstance(IoApSeqGenStru.class);
IoApGenSeqNo ApGenSeqNoAPI =
SysUtil.getInstance(IoApGenSeqNo.class);
cplIoSeqGenStru.setRiqiiiii(CommTools.prcRunEnvs().getJiaoyirq());
// 非必传
cplIoSeqGenStru.setXuhozlei("DXZXH"); // 必传
cplIoSeqGenStru.setXuhaoomc("待销账序号"); // 必传
cplIoSeqGenStru.setCzzhouqi(E_QIXIANDW.None); // 必传
cplIoSeqGenStru.setSuoshumk(E_SUOSHUMK.TAAC); // 必传
cplIoSeqGenStru.setDlswclbz(E_SHIFOUBZ.YES);
long seq = ApGenSeqNoAPI.genSeqNo(cplIoSeqGenStru);
```

注：目前 TDSQL 也不支持 Oracle 自增的序列,但是支持全局唯一字段，不能保证自增。

1.3.4 TDSQL 不支持 Oracle 语句中的 decode 写法

样例 1：

原 oracle 语句：

```
SELECT
a.*,decode(substr(a.cunqiiii,2,1),'D','1','W','7','S','10','M','30','Q','90','
Y','360')*substr(a.cunqiiii,1,1)AS cunqi

FROM kdpf_cunqkz a

WHERE  a.chapbhao = #chapbhao#

AND a.huobdaih = #huobdaih#
```

```
AND a.farendma = #farendma#

and a.jiluztai = '0'

ORDER BY cunqi
```

decode ->case when 或者 if

现 TDSQL 语句 ：

```
SELECT

a.*,

CASE substr(a.cunqiiii, 2, 1)

WHEN 'D' then '1'

WHEN 'W' then '7'

WHEN 'S' then '10'

WHEN 'M' then '30'

WHEN 'Q' then '90'

WHEN 'Y' then '360'

ELSE '1'

END

 * substr(a.cunqiiii, 1, 1) AS cunqi

FROM kdpf_cunqkz a

WHERE  a.chapbhao = #chapbhao#

AND a.huobdaih = #huobdaih#

AND a.farendma = #farendma#
```

```
and a.jiluztai = '0'

ORDER BY cunqi
```

样例 2：

类似如下的 decode 写法如下：

```
select decode(max(xuhaoooo), null, 0, max(xuhaoooo)+1)  as xuhaoooo from
kcfb_cftylx where farendma=#farendma# and kehuhaoo =#kehuhaoo#
```

改写成如下：

```
select case when max(xuhaoooo) is NULL then 0 else max(xuhaoooo)+1 end as
xuhaoooo from kcfb_cftylx where farendma=#farendma# and kehuhaoo =#kehuhaoo#
```

嵌套 case when 样例：

```
select case l.lilvfdlx

when '0' then l.lilvleix

when '1' then (case when l.llxfzdds='99' then '990' else l.lilvleix-l.llxfzdds
end)

when '2' then (case when l.llxfbfbi='99' then '990' else l.lilvleix*(1-
l.llxfbfbi/100) end)

else l.lilvleix

end from kitp_lblvtb l where fdfwszfs <>'1'  and l.jiluztai='0' and
farendma='9999';
```

1.3.5 TDSQL 不支持 Oracle 语句中的 to_char 写法

样例：

原 oracle 语句：

```
select to_char((case when count(*) <> 0 then 0 else 1 end)) as xianzhbz
  from ktal_jylsmx
 where farendma = #farendma#
   and jiaoyirq = #jiaoyirq#
   and guiylius = #guiylius#
   and xianzhbz = '0'
```

to_char 可改成 TDSQL 中的 convert 函数

CONVERT(value, char)

用法：CONVERT(value, type)

可用的 type 类型如下：

二进制,同带 binary 前缀的效果 : BINARY

字符型,可带参数 : CHAR()

日期 : DATE

时间: TIME

日期时间型 : DATETIME

浮点数 : DECIMAL

整数 : SIGNED

无符号整数 : UNSIGNED

现 TDSQL 语句：

```
select convert((case
                 when count(*) <> 0 then
                  0
                 else
                  1
               end),char) as xianzhbz
  from ktal_jylsmx
 where farendma = #farendma#
   and jiaoyirq = #jiaoyirq#
   and guiylius = #guiylius#
   and xianzhbz = '0'
```

1.3.6 TDSQL 不支持 Oracle 语句中的 to_number 写法

样例：

原 oracle 语句：

```
SELECT
*
FROM
kcda_pngzdz
WHERE
```

```
farendma = #farendma# and pingzhxh>=to_number(#qspzxhao#)

AND pingzhxh <= to_number (

#zzpzxhao#) order by pingzhxh
```

原 to_number 写法改写成如下：

cast(11 as unsigned int) /*整型*/

cast(11 as decimal(10,2)) /*浮点型*/

现 TDSQL 语句：

```
SELECT

*

FROM

kcda_pngzdz

WHERE

farendma = #farendma# and pingzhxh>=cast(#qspzxhao# as unsigned int)

AND pingzhxh <= cast(

#zzpzxhao# as unsigned int) order by pingzhxh
```

1.3.7 TDSQL 不支持 Oracle 语法中的|| ||拼接写法

样例：

```
SELECT

*

FROM

ktap_zjywbm

WHERE

farendma = '999'

AND zhhuywzl = '10'

AND ywbimasm LIKE '%' || '111' || '%';
```

'%' || '111' || '%' 改写成 concat(str1,str2,str3,...)

改动后如下：

```
SELECT

*

FROM

ktap_zjywbm

WHERE

farendma = '9999'

AND zhhuywzl = '10'

AND ywbimasm LIKE concat('%','111','%');
```

1.3.8 TDSQL 不支持 Oracle 语句 listagg within group 写法

原 Oracle 语句：

```
SELECT
listagg (
a.weidleib || ':' || a.weiduzhi, '|'
) within GROUP (


ORDER BY
a.weidleib,
a.weiduzhi
) wduxxzhm
FROM
kcgp_cjsjmx a
```

listagg() within group() 改成 group_concat()

函数语法：

group_concat([DISTINCT] 要连接的字段 [Order BY 排序字段 ASC/DESC] [Separator '分隔符'])

现 TDSQL 语句：

```
SELECT
GROUP_CONCAT(
a.weidleib,
':',
```

```
a.weiduzhi

ORDER BY

a.weidleib,

a.weiduzhi SEPARATOR '|'

) wduxxzhm

FROM

kcgp_cjsjmx a
```

1.3.9 TDSQL 不支持 Oracle 语句中的 to_date 写法

样例：

原 oracle 语句：

```
SELECT

a.huobdhao,

sum(

c.meiqhkze - trunc (

a.ysyjlxfs * trunc (

to_date (b.daoqriqi, 'yyyymmdd') - to_date (

#jiaoyirq#,'yyyymmdd'))  + a.ysyjlixi)) as zongjine

FROM

klnl_dkjxmx a,

klnb_dkzhqg b,

klnb_dkhksx c
```

```
WHERE

a.farendma = #farendma# and b.farendma=#farendma# and c.farendma=#farendma#

AND substr(b.daoqriqi, 1, 6) = substr(

#jiaoyirq#,1,6)

AND b.benqizht = '0'

AND b.jiluztai = '0'

AND a.dkjiejuh = b.dkjiejuh

AND a.jixiriqi = #jiaoyirq# and a.jiluztai = '0'

AND c.dkjiejuh = b.dkjiejuh

AND c.jiluztai = '0'

AND EXISTS (

SELECT

*

FROM

klnb_dkhksx

WHERE

dkjiejuh = b.dkjiejuh

AND huankfsh = '3'

)

GROUP BY

a.huobdhao

ORDER BY
```

a. huobdhao

to_date 改成 str_to_date

用法：str_to_date(date,format)

date 参数是合法的日期，*format* 规定日期/时间的输出格式

格式如下

%a	缩写星期名
%b	缩写月名
%c	月，数值
%D	带有英文前缀的月中的天
%d	月的天，数值(00-31)
%e	月的天，数值(0-31)
%f	微秒
%H	小时 (00-23)
%h	小时 (01-12)
%I	小时 (01-12)
%i	分钟，数值(00-59)
%j	年的天 (001-366)
%k	小时 (0-23)
%l	小时 (1-12)
%M	月名
%m	月，数值(00-12)
%p	AM 或 PM

%r	时间，12-小时（hh:mm:ss AM 或 PM）
%S	秒(00-59)
%s	秒(00-59)
%T	时间, 24-小时 (hh:mm:ss)
%U	周 (00-53) 星期日是一周的第一天
%u	周 (00-53) 星期一是一周的第一天
%V	周 (01-53) 星期日是一周的第一天，与 %X 使用
%v	周 (01-53) 星期一是一周的第一天，与 %x 使用
%W	星期名
%w	周的天 （0=星期日, 6=星期六）
%X	年，其中的星期日是周的第一天，4 位，与 %V 使用
%x	年，其中的星期一是周的第一天，4 位，与 %v 使用

%Y	年，4 位
%y	年，2 位

现 TDSQL 语句：

```
SELECT

a.huobdhao,

SUM(

c.meiqhkze - TRUNCATE (

a.ysyjlxfs * TRUNCATE (

str_to_date(b.daoqriqi, '%Y%m%d') - str_to_date('20160101', '%Y%m%d'),

0

) + a.ysyjlixi,

0

)

) AS zongjine

FROM

klnl_dkjxmx a,

klnb_dkzhqg b,

klnb_dkhksx c

WHERE

a.farendma = '9999'

AND b.farendma = '9999'

AND c.farendma = '9999'
```

```
AND substr(b.daoqriqi, 1, 6) = substr('20160101', 1, 6)

AND b.benqizht = '0'

AND b.jiluztai = '0'

AND a.dkjiejuh = b.dkjiejuh

AND a.jixiriqi = '20160101'

AND a.jiluztai = '0'

AND c.dkjiejuh = b.dkjiejuh

AND c.jiluztai = '0'

AND EXISTS (

SELECT

*

FROM

klnb_dkhksx

WHERE

dkjiejuh = b.dkjiejuh

AND huankfsh = '3'

)

GROUP BY

a.huobdhao

ORDER BY

a.huobdhao
```

1.3.10 TDSQL 不支持 Oracle 语句中的 trunc 写法

样例：

原 oracle 语句：

```
SELECT
trunc (
(
SELECT
count(1)
FROM
ksys_jyyxrz
WHERE
pljypich = #pljypich# and farendma = #farendma# and jiaoyirq =
to_date(#jiaoyirq#, 'yyyymmdd') and xitongbs = #mbxtbios#)
/ (
SELECT
count(1)
FROM
ksys_jykzhq
WHERE
farendma = #farendma# and xitongbs = #mbxtbios# and zhixbzhi = '1'), 2) * 100
AS taskProgress
FROM
```

```
DUAL
```

trunc 改成 truncate(X,D)函数

用法：TRUNCATE(X,D)函数返回数值 X 保留到小数点后 D 位的值

返回数字 X，截断到 D 小数位。 如果 D 为 0，结果没有小数点或小数部分。 D 是负数，导致值 X 的小数点左边的 D 数字变为零

现 TDSQL 语句：

```
SELECT
TRUNCATE (
(
SELECT
count(1)
FROM
ksys_jyyxrz
WHERE
pljypich = '1111'
AND farendma = '9999'
AND jiaoyirq = STR_TO_DATE('20170601', '%Y%m%d')
AND xitongbs = 'onlBatch'
) / (
SELECT
count(1)
FROM
```

```
ksys_jykzhq

WHERE

farendma = ' 9999 '

AND xitongbs = ' onlBatch '

AND zhixbzhi = ' 1 '

),

2

) * 100 AS taskProgress
```

1.3.11 TDSQL 不支持 Oracle 语句中的 full join 写法

样例：

```
SELECT

IFNULL(C.PIKNGPIC, F.PIKNGPIC) AS PIKNGPIC,

IFNULL(C.BISHUZHI, 0) + IFNULL(F.BISHUZHI, 0) AS BISHUZHI,

IFNULL(JIEFNGJE, 0) AS JIEFNGJE,

IFNULL(DAIFNGJE, 0) AS DAIFNGJE

FROM

(

SELECT

PIKNGPIC,

COUNT(A.YEWUBHAO) AS BISHUZHI,

SUM(A.JIAOYIJE) AS JIEFNGJE
```

```
FROM

KSTB_TCJHDJ A

INNER JOIN KSTB_TCJHFJ B ON A.YEWUBHAO = B.YEWUBHAO

WHERE

A.FARENDMA = '9999'

AND A.YINGYJIG = '0101'

AND A.JHQYDAIM = '0099'

AND A.JIAOHURQ = '20170601'

AND A.JIAOHUCC = 2

AND (

A.JIAOHULX = '0'

OR A.JIAOHULX = '6'

)

AND A.JILUZTAI = '0'

AND B.PIKNGPIC IS NOT NULL

AND B.PIKNGPIC <> 0

GROUP BY

B.PIKNGPIC

ORDER BY

B.PIKNGPIC

) C

FULL JOIN (
```

```
SELECT

PIKNGPIC,

COUNT(D.YEWUBHAO) AS BISHUZHI,

SUM(D.JIAOYIJE) AS DAIFNGJE

FROM

KSTB_TCJHDJ D

INNER JOIN KSTB_TCJHFJ E ON D.YEWUBHAO = E.YEWUBHAO

WHERE

D.FARENDMA = '9999'

AND D.YINGYJIG = '0101'

AND D.JHQYDAIM = '0099'

AND D.JIAOHURQ = '20170601'

AND D.JIAOHUCC = 2

AND (

D.JIAOHULX = '1'

OR D.JIAOHULX = '7'

)

AND D.JILUZTAI = '0'

AND E.PIKNGPIC IS NOT NULL

AND E.PIKNGPIC <> 0

GROUP BY

E.PIKNGPIC
```

```
ORDER BY

E.PIKNGPIC

) F ON C.PIKNGPIC = F.PIKNGPIC
```

Oracle 中 FuLL JOIN 改为 TDSQL 的 left join + union + right jion

如下：

```
SELECT

*

FROM

A

FULL JOIN B ON B. NAME = A. NAME;
```

改为：

```
SELECT

*

FROM

A

LEFT JOIN B ON B. NAME = A. NAME

UNION

SELECT

*
```

```
FROM

A

RIGHT JOIN B ON B. NAME = A. NAME;
```

现 TDSQL 语句：

```
SELECT

IFNULL(C.PIKNGPIC, F.PIKNGPIC) AS PIKNGPIC,

IFNULL(C.BISHUZHI, 0) + IFNULL(F.BISHUZHI, 0) AS BISHUZHI,

IFNULL(JIEFNGJE, 0) AS JIEFNGJE,

IFNULL(DAIFNGJE, 0) AS DAIFNGJE

FROM

(

SELECT

PIKNGPIC,

COUNT(A.YEWUBHAO) AS BISHUZHI,

SUM(A.JIAOYIJE) AS JIEFNGJE

FROM

KSTB_TCJHDJ A

INNER JOIN KSTB_TCJHFJ B ON A.YEWUBHAO = B.YEWUBHAO

WHERE

A.FARENDMA = '9999'

AND A.YINGYJIG = '0101'
```

```
AND A.JHQYDAIM = '0099'

AND A.JIAOHURQ = '20170601'

AND A.JIAOHUCC = 2

AND (

A.JIAOHULX = '0'

OR A.JIAOHULX = '6'

)

AND A.JILUZTAI = '0'

AND B.PIKNGPIC IS NOT NULL

AND B.PIKNGPIC <> 0

GROUP BY

B.PIKNGPIC

ORDER BY

B.PIKNGPIC

) C

LEFT JOIN (

SELECT

PIKNGPIC,

COUNT(D.YEWUBHAO) AS BISHUZHI,

SUM(D.JIAOYIJE) AS DAIFNGJE

FROM

KSTB_TCJHDJ D
```

```
INNER JOIN KSTB_TCJHFJ E ON D.YEWUBHAO = E.YEWUBHAO

WHERE

D.FARENDMA = '9999'

AND D.YINGYJIG = '0101'

AND D.JHQYDAIM = '0099'

AND D.JIAOHURQ = '20170601'

AND D.JIAOHUCC = 2

AND (

D.JIAOHULX = '1'

OR D.JIAOHULX = '7'

)

AND D.JILUZTAI = '0'

AND E.PIKNGPIC IS NOT NULL

AND E.PIKNGPIC <> 0

GROUP BY

E.PIKNGPIC

ORDER BY

E.PIKNGPIC

) F ON C.PIKNGPIC = F.PIKNGPIC

UNION

SELECT

IFNULL(C.PIKNGPIC, F.PIKNGPIC) AS PIKNGPIC,
```

```
IFNULL(C.BISHUZHI, 0) + IFNULL(F.BISHUZHI, 0) AS BISHUZHI,

IFNULL(JIEFNGJE, 0) AS JIEFNGJE,

IFNULL(DAIFNGJE, 0) AS DAIFNGJE

FROM

(

SELECT

PIKNGPIC,

COUNT(A.YEWUBHAO) AS BISHUZHI,

SUM(A.JIAOYIJE) AS JIEFNGJE

FROM

KSTB_TCJHDJ A

INNER JOIN KSTB_TCJHFJ B ON A.YEWUBHAO = B.YEWUBHAO

WHERE

A.FARENDMA = '9999'

AND A.YINGYJIG = '0101'

AND A.JHQYDAIM = '0099'

AND A.JIAOHURQ = '20170601'

AND A.JIAOHUCC = 2

AND (

A.JIAOHULX = '0'

OR A.JIAOHULX = '6'

)
```

```
AND A.JILUZTAI = '0'

AND B.PIKNGPIC IS NOT NULL

AND B.PIKNGPIC <> 0

GROUP BY

B.PIKNGPIC

ORDER BY

B.PIKNGPIC

) C

RIGHT JOIN (

SELECT

PIKNGPIC,

COUNT(D.YEWUBHAO) AS BISHUZHI,

SUM(D.JIAOYIJE) AS DAIFNGJE

FROM

KSTB_TCJHDJ D

INNER JOIN KSTB_TCJHFJ E ON D.YEWUBHAO = E.YEWUBHAO

WHERE

D.FARENDMA = '9999'

AND D.YINGYJIG = '0101'

AND D.JHQYDAIM = '0099'

AND D.JIAOHURQ = '20170601'

AND D.JIAOHUCC = 2
```

```
AND (
D.JIAOHULX = '1'
OR D.JIAOHULX = '7'
)
AND D.JILUZTAI = '0'
AND E.PIKNGPIC IS NOT NULL
AND E.PIKNGPIC <> 0
GROUP BY
E.PIKNGPIC
ORDER BY
E.PIKNGPIC
) F ON C.PIKNGPIC = F.PIKNGPIC;
```

1.3.12 TDSQL 不支持 Oracle 语句中的 start with ...connect by prior...level 写法

样例：

```
SELECT
b.*
FROM
kdpb_dngjdj b
WHERE
b.zhanghao IN (
```

```
SELECT
a.sjxtzhao AS zhanghao
FROM
kdpa_zhzhjg a START WITH a.zhanghao = #zhanghao#
connect BY prior a.sjxtzhao = a.zhanghao
AND LEVEL <= 6
AND a.farendma = #farendma#)
AND b.jiedonbz = '0'
```

Oracle 中 start with ... connect by prior...level 递归查询目前没有对应的替代函数

目前采用 改造 sql+改造程序的方式解决这个不兼容问题：

改造 sql：

```
SELECT
b.*
FROM
kdpb_dngjdj b
WHERE
b.zhanghao IN (
#zhanghao#)
AND b.jiedonbz = '0';
```

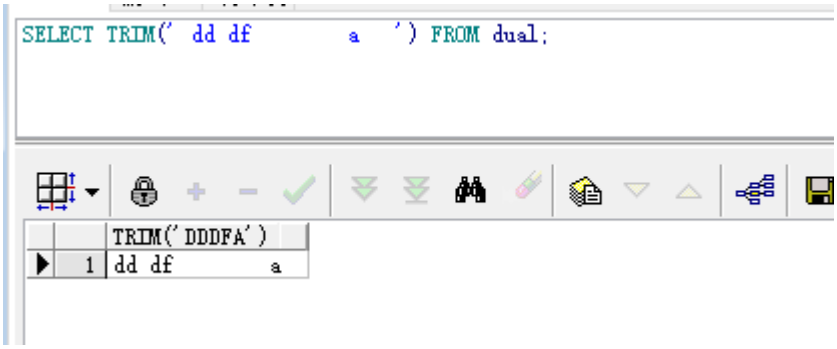
改造程序：

```
int  i=0;
String  tmpZhanghao=sZhanghao;
String  sEndZhanghao="";
while(true){
                kdpa_zhzhjg  tblKdpa_zhzhjg  =  Kdpa_zhzhjgDao.selectOdb
Zhanghao,false);
                if(CommUtil.isNotNull(tblKdpa_zhzhjg)  ){
                                if(CommUtil.isNotNull(sEndZhanghao)){
                                                sEndZhanghao=sEndZhanghao+","+tblK
dpa_zhzhjg.getSjxtzhao();
                                }else{
                                                sEndZhanghao=sEndZhanghao+tblKdpa_zh
o();
                                }
                               tmpZhanghao=tblKdpa_zhzhjg.gettblKdpa_zhzhjg();
                }else{
                                break;
                }
                if(i>5){
                      break;
                }
                i++;
}
```

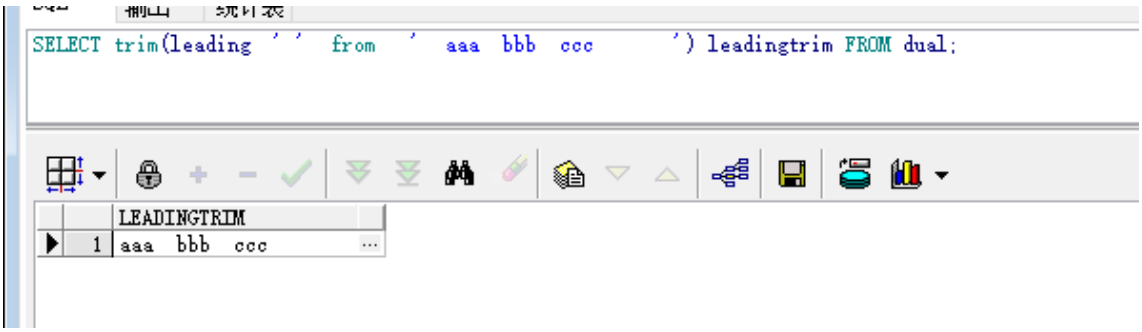
1.3.13 TDSQL 与 Oracle 语句中的 trim、rtrim、ltrim 用法差异

Oracle 中用法：

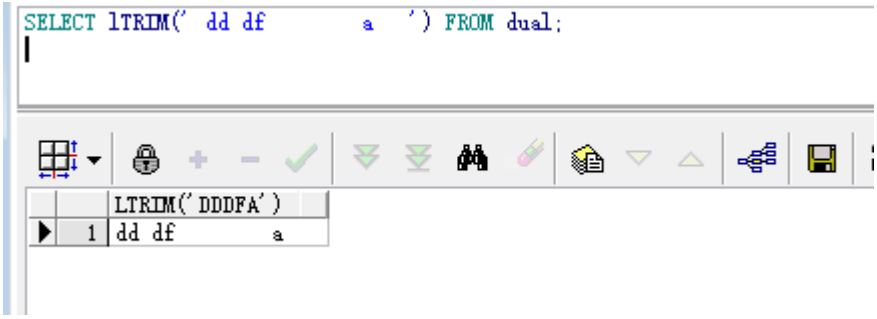
1、trim(str)：去除指定字符串 str 的左右空格



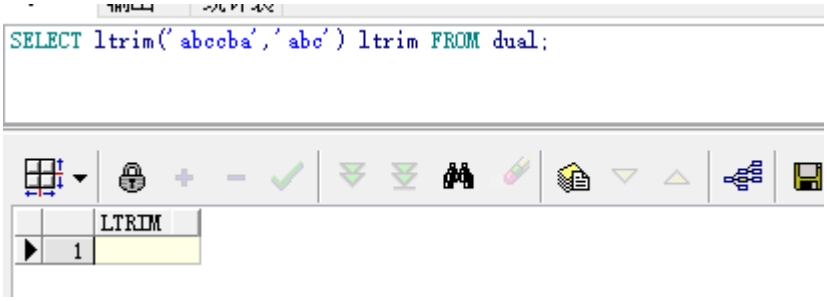
2、trim(leading | trailing | both string1 FROM string2)：从 string2 中去除左侧 | 右侧 | 左右两侧（默认是 both，即左右侧都去掉）的 string1 字符，注意，string1 只能是单个字符；

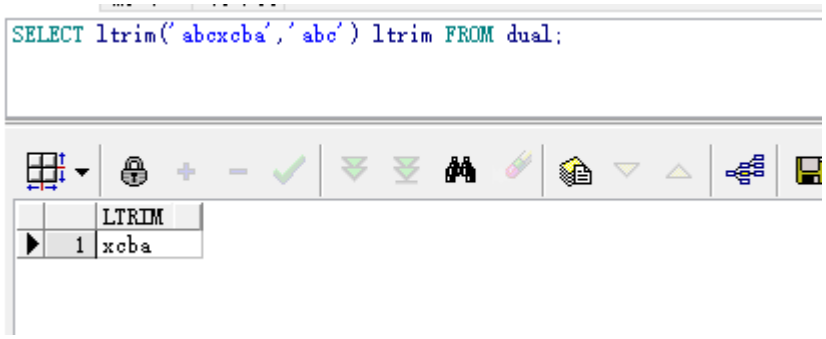


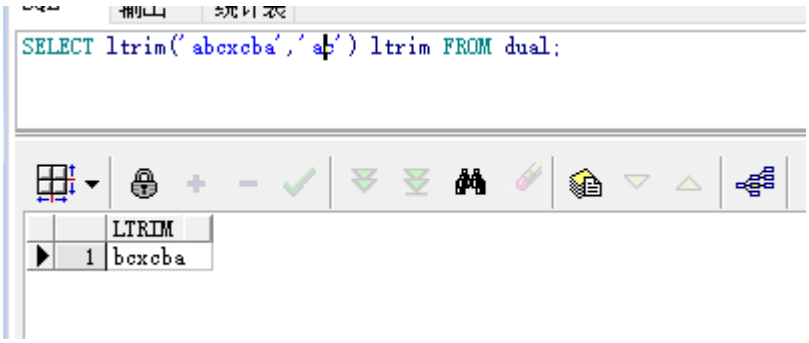
3、ltrim（str）：去除指定字符串 str 的左空格



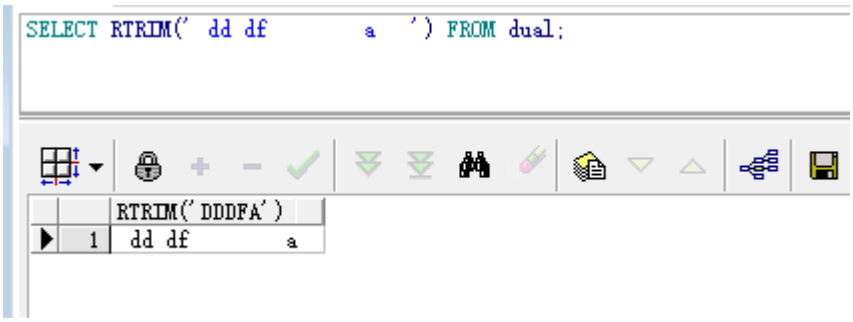
4、ltrim(str1,str2)：去除 str1 从左侧开始的与 str2 字符集合中匹配的单个字符，直到在 str1 中遇上某个字符，该字符不在 str2 字符集合中



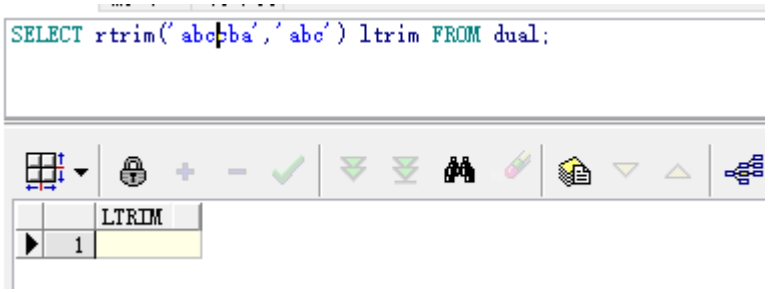


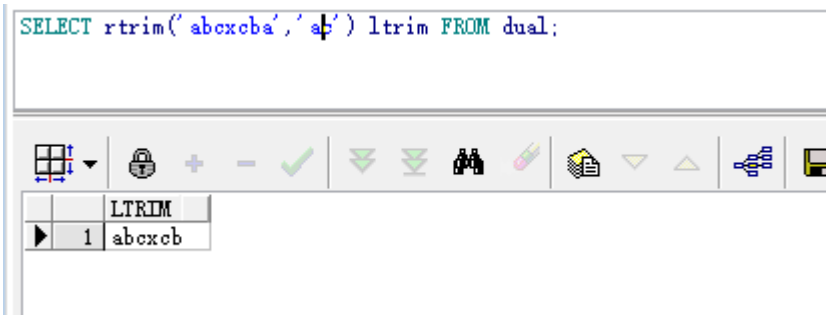


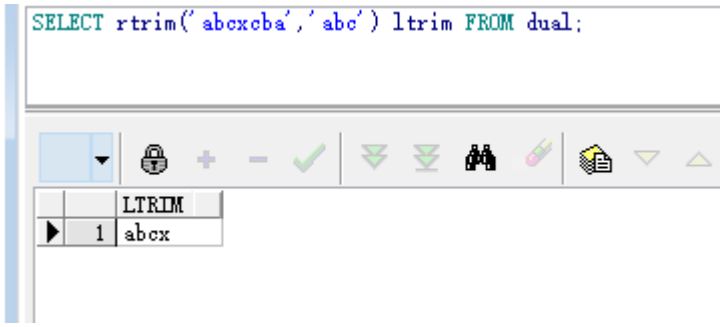
5、rtrim（str）：去除指定字符串 str 的右空格



6、rtrim（str1，str2）：去除 str1 从右侧开始的与 str2 字符集合中匹配的单个字符，直到在 str1 中遇上某个字符，该字符不在 str2 字符集合中

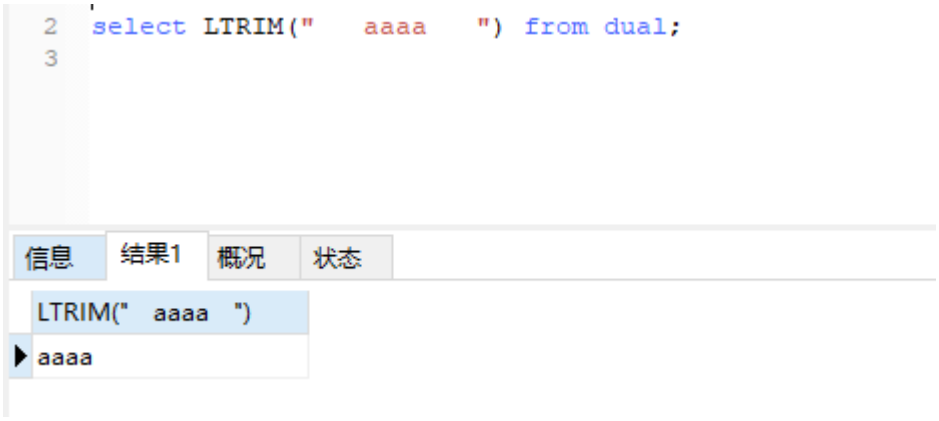






TDSQL 中用法：

1、ltrim(str)、rtrim(str)　:参数值只能是一个，去除 str 左右空字符



```
select RTRIM("    aaaa    ") from dual;
```

信息 结果1 概况 状态

RTRIM(" aaaa ")
aaaa

2、trim(str) :去除 str 左右两侧空格

```
select TRIM("    aaaa    ") from dual;
```

信息 结果1 概况 状态

TRIM(" aaaa ")
aaaa

3、

完整格式：TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)

简化格式：TRIM([remstr FROM] str)

```
select TRIM(LEADING ', ' FROM ', , barxxx') from dual;
```

信息 结果1 概况 状态

TRIM(LEADING ', ' FROM
barxxx

1.3.14 TDSQL 不支持 Oracle 语句中派生表查询

进行嵌套查询的时候子查询出来的的结果是作为一个派生表来进行上一级的查询的，所以子查询的结果必须要有一个别名

原 oracle 语句：

```
SELECT * from(

select *  FROM kceb_pzkcun

 where farendma = #farendma#

    and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#

and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#

and qishipzh <= #qishipzh# and #qishipzh# <= zzpzhhao

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#



union



select *  FROM kceb_pzkcun

where farendma = #farendma#

and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#
```

```
and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#

and qishipzh <= #zzpzhhao# and #zzpzhhao# <= zzpzhhao

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#



union



select *  FROM kceb_pzkcun

where farendma = #farendma#

and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#

and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#

and qishipzh >= #qishipzh# and zzpzhhao <= #zzpzhhao#

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#
```

```
)
```

select * from (select * from A)

类似的改写成 select * from (select * from A) B

from 后面 select 集合加上别名 B

现 TDSQL 语句：

```
SELECT * from(

select *  FROM kceb_pzkcun

 where farendma = #farendma#

    and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#

and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#

and qishipzh <= #qishipzh# and #qishipzh# <= zzpzhhao

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#



union
```

```
select *  FROM kceb_pzkcun

where farendma = #farendma#

and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#

and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#

and qishipzh <= #zzpzhhao# and #zzpzhhao# <= zzpzhhao

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#



union



select *  FROM kceb_pzkcun

where farendma = #farendma#

and yngyjigo = #yngyjigo#

and weixdhao = #weixdhao#

and pingzhzl = #pingzhzl#

and fenhdaim = #fenhdaim#

and pngzphao = #pngzphao#
```

```
and qishipzh >= #qishipzh# and zzpzhhao <= #zzpzhhao#

and mianzhii = #mianzhii#

and pzdanwei = #pzdanwei#

and dspzsyzt = #dspzsyzt#

and jiluztai = #jiluztai#

) t
```

后面加上别名 t

1.3.15 分布式数据库插入 Insert 语句修改

由于插入数据时 TSDSQL 需要只要路由关键字段值，才能决定数据落在哪个分片上，故插入语句必须指定字段名插入

原 oracle 插入语句：

```
INSERT INTO `sys_ops_conf` VALUES ('HxTransInput', 'farendma', '9999', '默认核心交易法人代码', '');
```

现 TDSQL 插入语句：

```
insert into sys_ops_conf (PARMCD, PRMKEY, PRMVAL, DESCTX, NAMECH)

values ('HxTransInput', 'farendma', '9999', '默认核心交易法人代码', '');
```

原 oracle 插入语句：

```
INSERT INTO `sys_ops_conf` (select * from sys_ops_conf_bak)  ;
```

TDSQL 分布式数据库暂不支持下面语句

现 TDSQL 插入语句：

```
insert into sys_ops_conf (PARMCD, PRMKEY, PRMVAL, DESCTX, NAMECH)

(select * from sys_ops_conf_bak);
```

1.3.16 TDSQL 不支持 Oracle 语句中的 USERENV()写法

样例：

```
select 1E+15 * USERENV('INSTANCE') + 1E+13 * mod(USERENV('SID'), 100)

  from dual;
```

TDSQL 支持该类写法，使用到的请根据业务需要改写兼容的 sql。

1.3.17 TDSQL 不支持 Oracle 语句中 add_months 写法

样例：

原 Oracle 语句：

```
SELECT

kehuzhao,

zhhaoxuh

FROM

kcev_zppzhe

WHERE
```

```
farendma = #farendma#

AND pingzhzl IN ('001', '002', '003')

AND yngyjigo = #yngyjigo#

AND jiaoyirq <= #jiaoyirq#

AND jiaoyirq > to_char (

add_months (

to_date (

#jiaoyirq#, 'YYYYMMDD'), -1), 'YYYYMMDD')

AND jiluztai = '0'

AND shifoubz = '0'

AND beiyngzd >= #kaihuriq#

GROUP BY

kehuzhao,

zhhaoxuh
```

Oracle 中 add_months()函数替换成 TDSQL 中 adddate 函数：

Adddate()函数用法：

ADDDATE(date,INTERVAL expr unit)

其中：

date 时间,expr 表达式值,unit 表达式对应的时间单位

unit ： SECOND,MINUTE,HOUR,DAY,MONTH,YEAR

当前时间：12：06：03

```
SELECT ADDDATE(NOW(),INTERVAL -60 SECOND)
```

信息 结果1 概况 状态

ADDDATE(NOW(),INTERVAL -60 SECOND)
2018-07-02 12:05:03

当前日期 20180702

```
SELECT ADDDATE(NOW(),INTERVAL -1 MONTH)
```

信息 结果1 概况 状态

ADDDATE(NOW(),INTERV
2018-06-02 12:06:51

现在 TDSQL 语句：

```
SELECT

kehuzhao,

zhhaoxuh

FROM

kcev_zppzhe

WHERE

farendma = #farendma#

AND pingzhzl IN ('001', '002', '003')

AND yngyjigo = #yngyjigo#

AND jiaoyirq <= #jiaoyirq#
```

```
AND jiaoyirq > CONVERT (

ADDDATE(

str_to_date(

#jiaoyirq#, '%Y%m%d'), INTERVAL -1 MONTH), date)

AND jiluztai = '0'

AND shifoubz = '0'

AND beiyngzd >= #kaihuriq#

GROUP BY

kehuzhao,

zhhaoxuh
```

1.3.18 TDSQL 与 Oracle 语句中 substr 用法差异

Oracle 中用法：

格式 1： substr(string string, int a, int b);样例：

格式 2：substr(string string, int a) ;

其中：

格式 1：

1、string 需要截取的字符串

2、a 截取字符串的开始位置（注：当 a 等于 0 或 1 时，都是从第一位开始截取）

3、b 要截取的字符串的长度

格式 2：

1、string 需要截取的字符串

2、a 可以理解为从第 a 个字符开始截取后面所有的字符串。

TDSQL 中用法：

格式 1: substr(string, start)；从 start 开始的位置，一直截取到最后

格式 2：substr(string,start,length)

其中：

格式 1：

1、string 需要截取的字符串

2、start 截取字符串的开始位置（注：start 只能从 1 开始）

格式 2：

1、string 需要截取的字符串

2、start 截取字符串的开始位置（注：start 只能从 1 开始）

3、length 要截取的字符串的长度

原 oracle 语句：

```
SELECT DISTINCT
(substr(kahaoooo, '0', '9')) AS ylchphao
FROM
kcda_pzjcxx
WHERE
farendma = #farendma# and
ksqriqii = #ksqriqii# and
```

```
(
pzsyztai = 'D'
OR pzsyztai = 'E'
OR pzsyztai = 'F'
OR pzsyztai = 'h'
)
AND (
kajiezhi = '1'
OR kajiezhi = '2'
OR kajiezhi = '3'
)
```

现 TDSQL 语句：

```
SELECT DISTINCT
(substr(kahaoooo, '1', '9')) AS ylchphao
FROM
kcda_pzjcxx
WHERE
farendma = #farendma# and
ksqriqii = #ksqriqii# and
(
pzsyztai = 'D'
OR pzsyztai = 'E'
```

```
OR pzsyztai = 'F'
OR pzsyztai = 'h'
)
AND (
kajiezhi = '1'
OR kajiezhi = '2'
OR kajiezhi = '3'
)
```

1.3.19 TDSQL 不支持 Oracle 语句中的 row_number() over 写法

Oracle 函数：

row_number() over(partition by expr1,expr2... order by expr1)

分组排序功能

目前 TDSQL 没有好的改写方法，业务上根据需要做出调整。

1.3.20 TDSQL 分布式数据库不支持 insert into select from 写法

目前 TDSQL 分布式数据库暂不支持 insert into select from 写法，业务上遇到根据业务情况临时调整，TDSQL 改造时间还未定。

样例如下：

```
insert into testaa (FARENDMA,MOKUAIII) select a.farendma,a.SUOSHUMK from
kapp_bmguiz a;
```

1.3.21 TDSQL 不支持 Update 的条件存在复杂的语句 exists()

样例：

原 Oracle 语句：

```
update khsb_linszf a

   set a.chulztai = #chulztai#

 where exists (SELECT 1

          FROM khst_daichl b

         where a.jiaoyirq = b.jiaoyirq

           and a.jioylius = b.jioylius

           and a.mxixuhao = b.mxixuhao

           and a.farendma = b.farendma

           and a.farendma = #farendma#

           and b.zhanghao = #zhanghao#)
```

TSDSQL 暂时不支持全局表和分片表的此类语法操作

业务遇到可将 sql 进行拆解，业务上可能也需要进行调整。

1.3.22 TDSQL 批量并发删除死锁问题

样例：

```
DELETE

FROM

kglb_zbminx
```

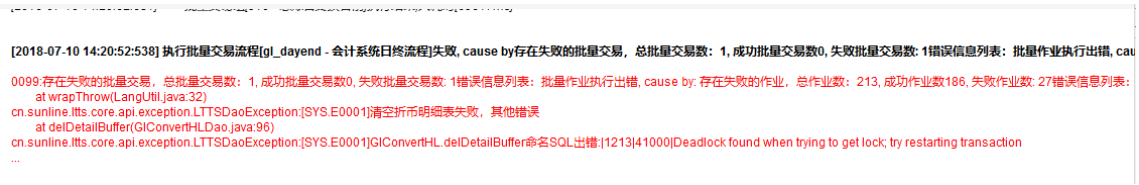
```
WHERE

farendma = #farendma#

AND jigouhao = #jigouhao#

AND baobleix = #baobleix#
```

按机构并发执行删除操作，Oracle 无问题，TDSQL 产生死锁报错



暂时解决方法改成 1 并发处理

1.3.23 TDSQL 不支持 select rownum from 语句

样例：

```
SELECT

r

FROM

(

SELECT

rownum AS r,

JIOYLIUS,

JIAOYIRQ,

MXIXUHAO

FROM
```

```
khst_daichl

WHERE

zhanghao = '990115601020171000050'

AND farendma = '9999'

) t2

WHERE

t1.jioylius = t2.jioylius

AND t1.jiaoyirq = t2.jiaoyirq

AND t1.mxixuhao = t2.mxixuhao
```

TDSQL 不支持 rownum 写法

业务根据需要进行调整

暂时无法处理

1.4 分片规则问题

1.4.1 分片表关键字段个数问题

目前分片表大部分都按一个字段比方 zhanghao 作为分片关键字，如果使用多字段组合作为分片字段，简单的拼接成一个字符串？

答：目前 tdsql 支持一个分片关键字

1.4.2 多唯一索引表的分片规则

比方账户对照表 kdpa_zhduiz 表唯一索引有两个，目前分片规则只有一个字段，核心更新的时候更新一个表数据的时候会把除当前唯一索引的其他的字段作为 set 条件，导致分片路由失败

答：目前仅支持一个分片关键字，故可选取其中比较特殊的字段，比方流水号、账号、

客户号之类的作为分片关键字。

注：目前 TDSQl 仅支持分片关键字在唯一索引中，不包含分片字段的唯一索引的创建不了（严重缺陷）（目前多唯一索引表删除了不包含分片字段的索引）

1.5 主键创建问题

答：目前主键创建在不改变原表结构的基础上，把第一个唯一索引升级为主键

1.5.1 少部分表没有唯一索引

答：没有唯一索引的表可根据表类型创建成全局表或者业务评估是否加上唯一索引

1.5.2 主键创建后是否删除同类型唯一索引

答：创建主键后的同类型的唯一索引和主键不冲突，目前保留。

1.6 数据导入问题

TDSQL 数据库目前 DML 数据导入语句必须带上表字段名进行导入，否则导入失败。

如 insert into kapp_jioyxx values(…) 这种方式就报错。

答：insert 语句导入数据时必须指定对应的字段名，针对分片表的话必须带上分片关键字字段名

2 其他问题

2.1 IDE 工具问题

IDE 导出 TDSQL 语句脚本暂时无法满足 TDSQL 建表要求，需要调整。