**XTTS脚本功能**

云和恩墨(北京)信息技术有限公司

技术顾问 燕鑫

http://www.enmotech.com

**文档控制：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序** | **版本号** | **更改人** | **日期** | **备注** |
| 1 | 1.0版 | 燕鑫 | 2019-04-18 | 初始版本 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 调优方法论 - 4 -](#_Toc3819010)

# V3版本源库不是PDB

## 环境准备

创建如下目录：

[oracle@host01 ~]$ mkdir xtts

[oracle@host01 ~]$ cd xtts

[oracle@host01 xtts]$ unzip rman\_xttconvert\_v3.zip

Archive: rman\_xttconvert\_v3.zip

inflating: xtt.properties

inflating: xttcnvrtbkupdest.sql

inflating: xttdbopen.sql

inflating: xttdriver.pl

inflating: xttprep.tmpl

extracting: xttstartupnomount.sql

[oracle@host01 xtts]$ mkdir convert

[oracle@host01 xtts]$ mkdir stage\_dest

[oracle@host01 xtts]$ mkdir tmp

[oracle@host01 xtts]$ mkdir tmp\_dest

[oracle@host01 xtts]$ mkdir xttbackup

[oracle@host01 xtts]$ mkdir xttdata

convert stage\_dest tmp tmp\_dest xttbackup xttcnvrtbkupdest.sql xttdata xttdbopen.sql xttdriver.pl xttprep.tmpl xtt.properties xttstartupnomount.sql

然后解释一下各个目录：

1. stage\_dest说白了就是目标端的阶段目录，就是放从源端出来的备份集的。  
   所以本来应该还有一个stage\_source，但是这里为了假装模拟配置了NFS，所以我的实验是在本机做的，所以不需要stage\_source，只需要一个目录就可以。
2. tmp目录是源端的TMPDIR，tmp\_dest是目标端的TMPDIR，其实如果是NFS上的话，这俩也能共用一个，也就是不用每次做增量恢复的时候传文件了。但是这里为了熟悉流程，我就把它们分开了。
3. xttdata这个目录就是目标端临时放数据文件的，在目标端进行了restore之后，这个目录下就会出现数据文件。  
   这里着重说一下，DFT和非DFT的区别。DFT就是直接把数据文件传送到目标端的存放数据文件的路径下，所以额外配俩个参数srcdir和dstdir（一定要注意这俩个参数是的真实路径分别是源端的数据文件路径和目标端的数据文件路径，这些路径必须在oracle中创建成directory，这里填写的是这些对应的directory的名字）。  
   也就说DFT不需要额外划分一块空间放全备！  
   如果我们不配这俩个参数，我们这里就需要配置xttdata作为恢复出来的数据文件的路径，那就是非DFT。  
   也就是说非DFT我们需要一块额外的空间来放全备。
4. convert这个目录是用来做oracle中的directory的，这个directory是使用metadata落地导入准备的，其实多此一举了，后面我们可以看到，其实在目标端直接见tmp\_dest的directory就可以了。
5. xttbackup暂时没理解到它的用处

然后我们看下要倒腾的表空间。我准备把PROD4的test表空间导入到YX这个库下，俩个库都是12.2的non-CDB。

我们来看下要移的这个test表空间：

SQL> select file#,t1.name,t2.name tablespace\_name,t2.blocksize,CHECKPOINT\_CHANGE# from v$datafile t1,ts$ t2 where t1.ts#=t2.ts# and t2.name='TEST';

FILE# NAME TABLESPACE\_NAME BLOCKSIZE CHECKPOINT\_CHANGE#

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

5 /u01/app/oracle/oradata/PROD4/test01.dbf TEST 16384 11484582

我们看到这里scn号比较新是因为这个查询是我实验完补充的。注意我实验用的表空间是16k的表空间。

自包含检查：

SQL> execute dbms\_tts.transport\_set\_check('TEST',true,true);

PL/SQL procedure successfully completed.

SQL> select \* from transport\_set\_violations;

no rows selected

用到这个表空间的也就是只有TEST用户。

所以我们在YX用户下把test用户创建出来。

SQL> create user test identified by test;

User created.

SQL> grant dba to test;

Grant succeeded.

这里给目标用户DBA权限，是为了后期导入方便，都弄完以后再revoke。

还有这次没开BCT，因为我写这个文档的主要目的是研究脚本功能。

## 参数文件

[oracle@host01 xtts]$ vi xtt.properties

tablespaces=TEST

platformid=13

srclink=TTSLINK

dfcopydir=/home/oracle/xtts/stage\_dest

backupformat=/home/oracle/xtts/stage\_dest

stageondest=/home/oracle/xtts/stage\_dest

storageondest=/home/oracle/xtts/xttdata

backupondest=/home/oracle/xtts/xttbackup

parallel=3

rollparallel=2

getfileparallel=4

tablespace就是要倒腾的表空间名，用逗号分隔列出来就行。

platform\_id是源端的platform\_id：

SQL> select PLATFORM\_ID,PLATFORM\_NAME,ENDIAN\_FORMAT from v$transportable\_platform order by platform\_id;

PLATFORM\_ID PLATFORM\_NAME ENDIAN\_FORMAT

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

1 Solaris[tm] OE (32-bit) Big

2 Solaris[tm] OE (64-bit) Big

3 HP-UX (64-bit) Big

4 HP-UX IA (64-bit) Big

5 HP Tru64 UNIX Little

6 AIX-Based Systems (64-bit) Big

7 Microsoft Windows IA (32-bit) Little

8 Microsoft Windows IA (64-bit) Little

9 IBM zSeries Based Linux Big

10 Linux IA (32-bit) Little

11 Linux IA (64-bit) Little

12 Microsoft Windows x86 64-bit Little

13 Linux x86 64-bit Little

15 HP Open VMS Little

16 Apple Mac OS Big

17 Solaris Operating System (x86) Little

18 IBM Power Based Linux Big

19 HP IA Open VMS Little

20 Solaris Operating System (x86-64) Little

21 Apple Mac OS (x86-64) Little

20 rows selected.

srclink这个是目标库到源库的dblink：

SQL> create public database link ttslink connect to system identified by oracle using 'PROD4';

Database link created.

dfcopydir和backupformat这俩目录是源端的目录：

1. dfcopydir是如果使用rman做0级备份的话，就把0级备份放这个目录下（我们本次也只讨论RMAN的方式）。
2. backupformat放增量备份的目录

stageondest和storageondest这俩目录是目标端的目录：

1. stageondest是放从源库传过来的全量备份和增量备份的
2. storageondest是放转化后的数据文件的，这个目录当然理所应当直接拿来作为最终的目标端数据文件存放路径。

既然我们假装是配了NFS，所以dfcopydir、backupformat和stageondest都指定为了stage\_dest。storageondest指定到了xttdata

parallel就是增量备份的并行度，DEVICE TYPE DISK PARALLELISM这个也能控制。

rollparallel就是前滚的并行度

getfileparallel就是使用DFT是传输文件的并行度，不指定这个参数可以，在使用xttdriver.pl时，指定-G xx是一样的效果。

## 正式割接前

### 修改环境变量

开俩个窗口一边假装源端一边假装目标端：

源端：

[oracle@host01 xtts]$ export ORACLE\_SID=PROD4

[oracle@host01 xtts]$ export TMPDIR=/home/oracle/xtts/tmp

目标端：

[oracle@host01 xtts]$ export ORACLE\_SID=yx

[oracle@host01 xtts]$ export TMPDIR=/home/oracle/xtts/tmp\_dest

### 源端0级备份（--backup）

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --backup

============================================================

trace file is /home/oracle/xtts/tmp/backup\_Apr17\_Wed\_15\_22\_44\_14//Apr17\_Wed\_15\_22\_44\_14\_.log

=============================================================

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

Parsing properties

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

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

Done parsing properties

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

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

Checking properties

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

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

Done checking properties

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

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

Starting prepare phase

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

Prepare source for Tablespaces:

'TEST' /home/oracle/xtts/stage\_dest

xttpreparesrc.sql for 'TEST' started at Wed Apr 17 15:22:44 2019

xttpreparesrc.sql for ended at Wed Apr 17 15:22:44 2019

Prepare source for Tablespaces:

'''' /home/oracle/xtts/stage\_dest

xttpreparesrc.sql for '''' started at Wed Apr 17 15:22:47 2019

xttpreparesrc.sql for ended at Wed Apr 17 15:22:47 2019

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

Done with prepare phase

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

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

Find list of datafiles in system

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

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

Done finding list of datafiles in system

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

做完备份，我们来看下源端的TMPDIR：

[oracle@host01 tmp]$ ls

backup\_Apr17\_Wed\_15\_22\_44\_14 incrbackups.txt tsbkupmap.txt xttnewdatafiles.txt xttplan.txt

我们看到多了一个文件夹，多了4个文件。

xttplan.txt是记录scn号和表空间所包含的文件号的，记住这个scn号。

[oracle@host01 tmp]$ cat xttplan.txt

TEST::::11475407

5

xttnewdatafiles.txt：

[oracle@host01 tmp]$ cat xttnewdatafiles.txt

::TEST

5,/home/oracle/xtts/xttdata/TEST\_5.dbf

tsbkupmap.txt是记录备份集对应关系的：

[oracle@host01 tmp]$ cat tsbkupmap.txt

::5:::1=TEST\_5\_0jtv7ne5\_1\_1.bkp

incrbackups.txt是记录备份集位置的：

[oracle@host01 tmp]$ cat incrbackups.txt

/home/oracle/xtts/stage\_dest/TEST\_5\_0jtv7ne5\_1\_1.bkp

注意tmp目录下的这些文件都是最新的，如果再做一次增量备份，这些文件的内容就会发生改变。而我们想要看这些文件在某一步时的历史内容，可以到对应这一步产生的文件夹中去找，这里就是backup\_Apr17\_Wed\_15\_22\_44\_14。

第一个下划线前面的就是生成该目录的指令名称，后面是具体时间。

我们到目录下看一下：

[oracle@host01 backup\_Apr17\_Wed\_15\_22\_44\_14]$ ls

Apr17\_Wed\_15\_22\_44\_14\_.log diff.sql fixedfile.txt.temp incrbackups.txt tsbkupmap.txt xttnewdatafiles.txt xttnewdatafiles.txt.all xttnewdatafiles.txt.temp xttplan.txt xttprepare.cmdApr17\_Wed\_15\_22\_44\_14 xttpreparesrc.sql

log里的内容跟我们执行指令时界面输出的内容一模一样。

这里面我们主要看cmd文件，首先说下，为啥这一步文件夹里没出现cmd文件。因为cmd文件是根据xttpreparesrc.sql这个生成的，而这个脚本中的表空间变量每次只传输一个，也就是说我们最终看到这个文件的时候，已经是按最后一个表空间的值生成的脚本。

那么为什么最后一个表空间值是空值呢？这个在1.5.1中有详细解释。

但这里还是需要小改动下脚本，因为我不知道为啥脚本里面没有指定表空间，需要在第一个sql中把我们要倒腾的表空间加上。

然后下面是我事后用脚本生成出来的cmd文件（所以scn会比实验中的scn新）：

[oracle@host01 backup\_Apr17\_Wed\_15\_22\_44\_14]$ cat xttprepare.cmd

#PLAN:TEST::::11484582

backup for transport allow inconsistent incremental level 0 datafile

#NEWDESTDF:5,/home/oracle/xtts/xttdata/TEST\_5.dbf

5

#PLAN:5

format '/home/oracle/xtts/stage\_dest/%N\_%f\_%U.bkp';

这里我们去掉注释，就是我们rman中可以执行的脚本，也就是我们做0级备份的指令。

这一部分的核心代码是函数generate\_batch\_tsoutput，流程应该是xttdriver.pl生成xttpreparesrc.sql，执行该sql生成cmd文件，然后xttdriver.pl执行cmd文件。

#### RMAN指令

backup for transport allow inconsistent incremental level 0 datafile 5

format '/home/oracle/xtts/stage\_dest/%N\_%f\_%U.bkp';

### 目标端restore（--restore）

首先要把源库的TMPDIR下tsbkupmap.txt，xttnewdatafiles.txt拷到目标端的TMPDIR下：

[oracle@host01 tmp]$ cp \*.txt ../tmp\_dest/

这里注意，因为我们假装配置了nfs，所以源端和目标端放备份的文件夹是一个，所以这里就不用拷备份了。

正常可以这样拷备份：

scp `cat incrbackups.txt` oracle@dest:/stageondest

然后我们在目标端，执行restore过程，再次提醒，目标端我们的TMPDIR是tmp\_dest，所以过程中生成的文件以及目录都会出现在tmp\_dest中。

做之前我们先看下storageondest：

[oracle@host01 xtts]$ ls xttdata/

确实目前什么都没有。

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --restore

============================================================

trace file is /home/oracle/xtts/tmp\_dest/restore\_Apr17\_Wed\_15\_25\_36\_588//Apr17\_Wed\_15\_25\_36\_588\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Start restore/recover

End of restore/recover phase

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

然后我们在看storageondest目录：

[oracle@host01 xtts]$ ls xttdata/

TEST\_5.dbf

我们的数据文件出来了。

然后我们到tmp\_dest目录下看一下，会发现多了一个restore\_Apr17\_Wed\_15\_25\_36\_588的目录。

我们到这个目录下，主要来看下cmd文件：

[oracle@host01 restore\_Apr17\_Wed\_15\_25\_36\_588]$ cat xttresrec\_5.cmd

restore from platform 'Linux x86 64-bit' FOREIGN DATAFILE 5 format '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/TEST\_5\_0jtv7ne5\_1\_1.bkp';

我们注意看这个foreign datafile，这就是在数据库外对数据文件进行restore、recover的精髓。我们会看到，后面增量recover时也是需要指定关键字FOREIGN DATAFILE。

#### RMAN指令

restore from platform 'Linux x86 64-bit' FOREIGN DATAFILE 5 format '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/TEST\_5\_0jtv7ne5\_1\_1.bkp';

### 源端1级增量备份（--bkpinc）

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --bkpinc

============================================================

trace file is /home/oracle/xtts/tmp/bkpincr\_Apr17\_Wed\_15\_34\_58\_239//Apr17\_Wed\_15\_34\_58\_239\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Backup incremental

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

Prepare newscn for Tablespaces: 'TEST'

Prepare newscn for Tablespaces: ''''

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

Starting incremental backup

Done backing up incrementals

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

增量备份完之后，我们看日志输出，注意我高亮的那一行，我们可以看到，多了个“空”表空间，这个就是在第一步，我们看到的xttpreparesrc.sql中的表空间那里是空的输出体现。

然后我们看tmp目录下的变化：

[oracle@host01 tmp]$ ls

backup\_Apr17\_Wed\_15\_22\_44\_14 bkpincr\_Apr17\_Wed\_15\_34\_58\_239 incrbackups.txt tsbkupmap.txt xttnewdatafiles.txt xttplan.txt xttplan.txt.new

我们看到多了个目录不新鲜，但还多了个文件xttplan.txt.new。

我们来打开xttplan.txt和xttplan.txt.new来比对下：

[oracle@host01 tmp]$ cat xttplan.txt

TEST::::11475407

5

[oracle@host01 tmp]$ cat xttplan.txt.new

TEST::::11477645

5

我们看到new的scn要新一些，其实流程是这样的：

1. 全备做之前，脚本会查一下v$datafile的CHECKPOINT\_CHANGE#，记录到xttplan.txt中
2. 做增量备份前再查一次v$datafile的CHECKPOINT\_CHANGE#，记录到xttplan.txt.new中
3. 然后做增量备份的时候，使用xttplan.txt中的scn做为起使scn，这一点在之后的cmd文件中得到确认。

**所以我们如果在手工做的时候，一定要在每次备份之前，先进行checkpoint\_change#的查询，再做备份**！

我们这里还需要注意的一个地方就是，incrbackups.txt是发生了变化的。

[oracle@host01 tmp]$ cat incrbackups.txt

/home/oracle/xtts/stage\_dest/0ktv7o54\_1\_1

现在该文件指向的是新的备份集，所以一会儿这个还要重新拷到目标端的TMPDIR。

最后我们就可以到对应目录下看cmd文件了：

[oracle@host01 bkpincr\_Apr17\_Wed\_15\_34\_58\_239]$ cat rmanincr.cmd

set nocfau;

host 'echo ts::TEST';

backup for transport allow INCONSISTENT incremental from scn 11475407

tablespace 'TEST' format

'/home/oracle/xtts/stage\_dest/%U';

#### 注意事项

注意事项如下：

1. 我们看到每一次备份，包括全备都是要加INCONSISTENT。
2. from scn正是xttplan.txt中的scn
3. 表空间名这里一定要用单引号扩起来，在11g中，只大写表空间名，可以不用单引号括起来，所以V3脚本在11g的环境中跑增量备份不会报错，在12.2的环境中跑就要报错。  
   所以强烈建议把v3脚本的函数backincr的内容进行修改：  
   把" $props{'tablespaces'}"改成" '$props{'tablespaces'}'"
4. set nocfau的意思即使开了controlfile autobackup，本次备份也不会进行controlfile autobackup。
5. host指令跟sqlplus的host或者!一样。

#### RMAN指令

set nocfau;

backup for transport allow INCONSISTENT incremental from scn 11475407

tablespace 'TEST' format

'/home/oracle/xtts/stage\_dest/%U';

### 目标端应用增量recover（--recover）

把这个3个文件：xttplan.txt、tsbkupmap.txt、incrbackups.txt拷到目标端的TMPDIR。

[oracle@host01 tmp]$ cp \*.txt ../tmp\_dest/

当然如果不是nfs，还要把新增的备份集拷到目标端：

scp `cat incrbackups.txt` oracle@dest:/stageondest

然后再目标端执行：

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --recover

============================================================

trace file is /home/oracle/xtts/tmp\_dest/recover\_Apr17\_Wed\_15\_38\_53\_5//Apr17\_Wed\_15\_38\_53\_5\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Start restore/recover

End of restore/recover phase

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

然后我们直接看cmd文件就行了：

[oracle@host01 recover\_Apr17\_Wed\_15\_38\_53\_5]$ cat xttresrec\_5.cmd

recover from platform 'Linux x86 64-bit' FOREIGN DATAFILECOPY '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/0ktv7o54\_1\_1';

注意这里是foreign datafilecopy了，而不是foreign datafile了。

#### RMAN指令

recover from platform 'Linux x86 64-bit' FOREIGN DATAFILECOPY '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/0ktv7o54\_1\_1';

### 源端生成新的下一次备份起始的scn（-s）

这一步一定要在成功完成本次增量备份后再做！

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl -s

============================================================

trace file is /home/oracle/xtts/tmp/determinescn\_Apr17\_Wed\_15\_39\_44\_414//Apr17\_Wed\_15\_39\_44\_414\_.log

=============================================================

Parsing properties

Done parsing properties

Checking properties

Done checking properties

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

Prepare newscn for Tablespaces: 'TEST'

Prepare newscn for Tablespaces: ''''

New /home/oracle/xtts/tmp/xttplan.txt with FROM SCN's generated

这一步说白了就是把xttplan.txt.new这个文件mv成xttplan.txt，做完该操作后，我们会发现，xttplan.txt.new这个文件已经没了。如下：

[oracle@host01 tmp]$ ls

backup\_Apr17\_Wed\_15\_22\_44\_14 bkpincr\_Apr17\_Wed\_15\_34\_58\_239 determinescn\_Apr17\_Wed\_15\_39\_44\_414 incrbackups.txt tsbkupmap.txt xttnewdatafiles.txt xttplan.txt

[oracle@host01 tmp]$ cat xttplan.txt

TEST::::11477645

5

当然该步骤也会产生一个目录：determinescn开头的目录。

1.3.4到1.3.6可以多次做，说白了就是不停的追增量。

后来我又做了一次增量备份加roll forward。这期间每次做增量前我都进行了事务操作，这里就不细说了。

## 正式割接

### 源库readonly表空间

SQL> alter tablespace test read only;

Tablespace altered.

### 源库最后一次增量备份（--bkpexport）

这里如果我们使用dblink的方式，我们就还是把前面的1.3.4-1.3.6做一遍，最后用如下指令来生成通过dblink来dump元数据的指令：

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl -e

这次实验，我们不用dblink，我们使用落地的方式导元数据。

所以我们最后一次增量备份使用的指令参数也会不同：

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --bkpexport

============================================================

trace file is /home/oracle/xtts/tmp/bkpexport\_Apr17\_Wed\_15\_44\_35\_193//Apr17\_Wed\_15\_44\_35\_193\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Backup incremental

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

============================================================

No new datafiles added

=============================================================

Prepare newscn for Tablespaces: 'TEST'

Prepare newscn for Tablespaces: ''''

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

Starting incremental backup

Done backing up incrementals

tmp目录下就只多了bkpexport\_Apr17\_Wed\_15\_44\_35\_193这个目录，我们进这个目录看一下。

先看备份片列表：

[oracle@host01 bkpexport\_Apr17\_Wed\_15\_44\_35\_193]$ cat incrbackups.txt

/home/oracle/xtts/stage\_dest/0ntv7ont\_1\_1

/home/oracle/xtts/stage\_dest/0mtv7ons\_1\_1

我们发现，怎么多了一个备份片？！

我们来看看cmd文件，这一步做了什么：

[oracle@host01 bkpexport\_Apr17\_Wed\_15\_44\_35\_193]$ cat rmanincr.cmd

BACKUP FOR TRANSPORT INCREMENTAL from scn 11479294 TABLESPACE 'TEST' FORMAT '/home/oracle/xtts/stage\_dest/%U' DATAPUMP FORMAT '/home/oracle/xtts/stage\_dest/%U';

我们看到这个指令跟前面的增量备份不同的就是，多了一个datapump的过程。

这个具体是个什么操作，其实可以看rman的log得知，但是我没搞清楚这个脚本怎么生成rmanlog（肯定是可以生成的，因为有GetRMANTrace这个函数）。那么我就直接把第一次实验的时候手动执行该指令的一个输出拿出来研究下：

RMAN> BACKUP FOR TRANSPORT INCREMENTAL from scn 11448805 TABLESPACE 'TEST' FORMAT '/home/oracle/xtts/stage\_dest/%U' DATAPUMP FORMAT '/home/oracle/xtts/stage\_dest/%U';

Starting backup at 17-APR-19

using target database control file instead of recovery catalog

allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=507 device type=DISK

Running TRANSPORT\_SET\_CHECK on specified tablespaces

TRANSPORT\_SET\_CHECK completed successfully

Performing export of metadata for specified tablespaces...

EXPDP> Starting "SYS"."TRANSPORT\_EXP\_PROD4\_avDd":

EXPDP> Processing object type TRANSPORTABLE\_EXPORT/TABLE\_STATISTICS

EXPDP> Processing object type TRANSPORTABLE\_EXPORT/STATISTICS/MARKER

EXPDP> Processing object type TRANSPORTABLE\_EXPORT/PLUGTS\_BLK

EXPDP> Processing object type TRANSPORTABLE\_EXPORT/POST\_INSTANCE/PLUGTS\_BLK

EXPDP> Processing object type TRANSPORTABLE\_EXPORT/TABLE

EXPDP> Master table "SYS"."TRANSPORT\_EXP\_PROD4\_avDd" successfully loaded/unloaded

EXPDP> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXPDP> Dump file set for SYS.TRANSPORT\_EXP\_PROD4\_avDd is:

EXPDP> /u01/app/oracle/product/12c/db\_1/dbs/backup\_tts\_PROD4\_22345.dmp

EXPDP> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXPDP> Datafiles required for transportable tablespace TEST:

EXPDP> /u01/app/oracle/oradata/PROD4/test01.dbf

EXPDP> Job "SYS"."TRANSPORT\_EXP\_PROD4\_avDd" successfully completed at Wed Apr 17 13:50:20 2019 elapsed 0 00:00:30

Export completed

channel ORA\_DISK\_1: starting full datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set

input datafile file number=00005 name=/u01/app/oracle/oradata/PROD4/test01.dbf

channel ORA\_DISK\_1: starting piece 1 at 17-APR-19

channel ORA\_DISK\_1: finished piece 1 at 17-APR-19

piece handle=/home/oracle/xtts/stage\_dest/0htv7i0t\_1\_1 tag=TAG20190417T134942 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:01

channel ORA\_DISK\_1: starting full datafile backup set

input Data Pump dump file=/u01/app/oracle/product/12c/db\_1/dbs/backup\_tts\_PROD4\_22345.dmp

channel ORA\_DISK\_1: starting piece 1 at 17-APR-19

channel ORA\_DISK\_1: finished piece 1 at 17-APR-19

piece handle=/home/oracle/xtts/stage\_dest/0itv7i0u\_1\_1 tag=TAG20190417T134942 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:01

Finished backup at 17-APR-19

RMAN> exit

Recovery Manager complete.

所以我们看到，效果就是把最后一次增量备份做了，并且把元数据临时导出到$ORACLE\_HOME/dbs下，然后单独做一个备份集。

那么这样做的风险点显然就是，如果$ORACLE\_HOME下没有足够的空间，肯定要报错。

不过这个路径应该是有办法指定的，只是我目前还不知道方法，这个后面再深究。

#### RMAN指令

BACKUP FOR TRANSPORT INCREMENTAL from scn 11479294 TABLESPACE 'TEST' FORMAT '/home/oracle/xtts/stage\_dest/%U' DATAPUMP FORMAT '/home/oracle/xtts/stage\_dest/%U';

### 目标库最后一次还原（--resincrdmp）

仍然是要把这3个文件：xttplan.txt、tsbkupmap.txt、incrbackups.txt拷到目标端的TMPDIR。

[oracle@host01 tmp]$ cp \*.txt ../tmp\_dest/

当然如果不是nfs，还要把新增的备份集拷到目标端：

scp `cat incrbackups.txt` oracle@dest:/stageondest

然后进行最后一次还原：

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl --resincrdmp

============================================================

trace file is /home/oracle/xtts/tmp\_dest/resincrdmp\_Apr17\_Wed\_15\_45\_51\_770//Apr17\_Wed\_15\_45\_51\_770\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Start restore/recover

End of restore/recover phase

Start creating dumpfile

End of creating dumpfile

Generating plugin

Done generating plugin file /home/oracle/xtts/tmp\_dest/xttplugin.txt

我们注意，最后一次还原一定不能用--recover，因为备份集列表中，有dump文件的备份，如果用recover这个参数，会报错，因为认不到最后这个备份集。

我们来看cmd文件：

[oracle@host01 resincrdmp\_Apr17\_Wed\_15\_45\_51\_770]$ cat xttresrec\_5.cmd

recover from platform 'Linux x86 64-bit' FOREIGN DATAFILECOPY '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/0mtv7ons\_1\_1';

我们看到，只有一个备份集的还原过程。

那么dump文件是怎么还原的呢？

我们可以看到，在xtts目录下，多了一个rman\_createdmp.cmd这个文件。

[oracle@host01 xtts]$ ls

convert rman\_createdmp.cmd stage\_dest tmp tmp\_dest xttbackup xttcnvrtbkupdest.sql xttdata xttdbopen.sql xttdriver.pl xttprep.tmpl xtt.properties xttstartupnomount.sql

我们看下它的内容：

[oracle@host01 xtts]$ cat rman\_createdmp.cmd

restore from platform 'Linux x86 64-bit' dump file 'impdp\_Apr17\_Wed\_15\_45\_51\_770.dmp' datapump destination '/home/oracle/xtts/tmp\_dest' from backupset '/home/oracle/xtts/stage\_dest/0ntv7ont\_1\_1';

我们看到，这一步不仅对dump做了平台转换，而且将dmp文件还原到了目标端的TMPDIR中。

所以我们完全可以把目标端的TMPDIR做一个directory，而不需要额外建个directory（比如本次实验的convert目录）。

最后我们看下该指令生成的另一个重要文件：tmp\_dest/xttplugin.txt

[oracle@host01 tmp\_dest]$ cat xttplugin.txt

impdp directory=<DATA\_PUMP\_DIR> logfile=<tts\_imp.log> \

dumpfile=impdp\_Apr17\_Wed\_15\_45\_51\_770.dmp \

transport\_datafiles='/home/oracle/xtts/xttdata/TEST\_5.dbf'

这个我们可以把用户名密码也加进去，也可以不加，执行的时候输。用system用户导入就行。

注意，我这次实验会报这个错误：

Processing object type TRANSPORTABLE\_EXPORT/PLUGTS\_BLK

ORA-39123: Data Pump transportable tablespace job aborted

ORA-29339: tablespace block size 16384 does not match configured block sizes

原因很简单，我们最开始就查了的，我们的实验表空间TEST是16k块大小的表空间，所以必须在目标端库中给db\_16k\_cache\_size一个值。可以改成跟源库一样：

SQL> alter system set db\_16k\_cache\_size=32M;

System altered.

成功导入元数据后，把表空间read write，验证一下数据是不是跟主库一致。

通过-e参数，可以生成使用dblink导入元数据的指令，当然，xtt.properties中一定要把srclink指定好。

[oracle@host01 xtts]$ $ORACLE\_HOME/perl/bin/perl xttdriver.pl -e

============================================================

trace file is /home/oracle/xtts/tmp/generate\_Apr17\_Wed\_14\_17\_10\_501//Apr17\_Wed\_14\_17\_10\_501\_.log

=============================================================

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

Parsing properties

Done parsing properties

Checking properties

Done checking properties

Generating plugin

Done generating plugin file /home/oracle/xtts/tmp/xttplugin.txt

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

[oracle@host01 xtts]$ cat /home/oracle/xtts/tmp/xttplugin.txt

impdp directory=<DATA\_PUMP\_DIR> logfile=<tts\_imp.log> \

network\_link=<ttslink> transport\_full\_check=no \

transport\_tablespaces=TEST \

transport\_datafiles='/home/oracle/xtts/xttdata/TEST\_5.dbf'

#### RMAN指令

recover from platform 'Linux x86 64-bit' FOREIGN DATAFILECOPY '/home/oracle/xtts/xttdata/TEST\_5.dbf' from backupset '/home/oracle/xtts/stage\_dest/0mtv7ons\_1\_1';

restore from platform 'Linux x86 64-bit' dump file 'impdp\_Apr17\_Wed\_15\_45\_51\_770.dmp' datapump destination '/home/oracle/xtts/tmp\_dest' from backupset '/home/oracle/xtts/stage\_dest/0ntv7ont\_1\_1';

#### impdp指令

落地：

impdp directory=<DATA\_PUMP\_DIR> logfile=<tts\_imp.log> \

dumpfile=impdp\_Apr17\_Wed\_15\_45\_51\_770.dmp \

transport\_datafiles='/home/oracle/xtts/xttdata/TEST\_5.dbf'

不落地：

impdp directory=<DATA\_PUMP\_DIR> logfile=<tts\_imp.log> \

network\_link=<ttslink> transport\_full\_check=no \

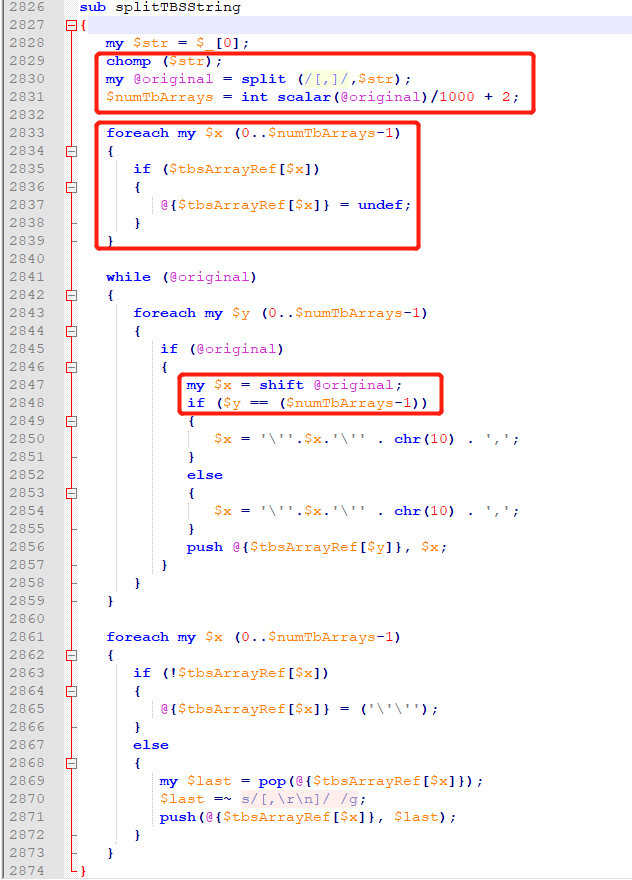
transport\_tablespaces=TEST \

transport\_datafiles='/home/oracle/xtts/xttdata/TEST\_5.dbf'

## 相关文件内容生成脚本

我们先把三个重要的文件列出来：xttnewdatafiles.txt、xttplan.txt、tsbkupmap.txt、incrbackups.txt

### splitTBSString函数

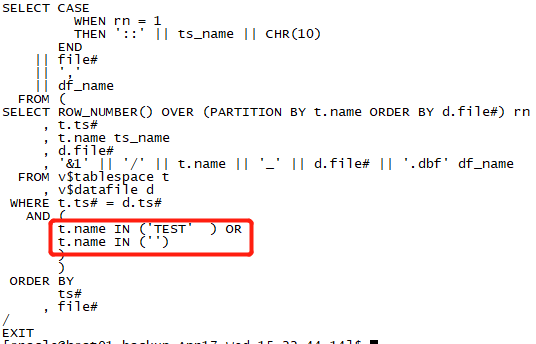


这个函数的关键作用就是把我们再参数文件中写的tablespaces字符串解析出来：

1. 将xxt.properties的tablespaces的值放到$str中
2. chomp函数可以把字符串中的换行符去掉
3. @original就是个数组，@符号就是perl语言定义数组的，这一部分，就是把$str按逗号切分，作为@original的每一个元素
4. $numTbArrays这个并不是@original的元素个数。它的意思就是要建一个有$numTbArrays个元素的数组，然后这个数组中每个元素本身就是一个数组。就是一个二元数组。  
   所以我们看$numTbArrays = int scalar(@original)/1000 + 2的意思就是，当@original的元素少于1000个的时候，$numTbArrays都是2。
5. 然后就是$tbsArrayRef[$x]，这个变量（其实就是个二元数组），有$numTbArrays个元素，这个循环就是把$tbsArrayRef中的内容都清空：@{$tbsArrayRef[$x]} = undef
6. 接下来就是一个循环读@original的每一个元素，这里举例子说明：  
   如果是@oraiginal现在是[TS1 TS2 TS3 TS4]，那么就把  
   TS1放在tbsArrayRef[0][0]中，  
   TS2放在tbsArrayRef[1][0]中，  
   TS3放在tbsArrayRef[0][1]中，  
   TS4放在tbsArrayRef[1][1]中，  
   说白了就是把要迁移的表空间分成俩组。最终效果就是：  
   @{$tbsArrayRef[0]}中放着'TS1','TS3'  
   @{$tbsArrayRef[1]}中放着'TS2','TS4'

所以，这就解释了为什么我们如果只倒腾一个表空间，会出现一个空值。因为最开始对$tbsArrayRef初始化时，把@{$tbsArrayRef[0]}和@{$tbsArrayRef[1]}都置为了undef。所以当只有一个表空间时，@{$tbsArrayRef[1]}就是空值。

这个效果可以从下面将要说的diff.sql看出：



OR前面就是@{$tbsArrayRef[0]}，OR后面就是@{$tbsArrayRef[1]}。

脚本调试过程就是，把开头加个#!/bin/perl，然后下面修修改改，注意打印变量是print "变量"。

如下：

#!/bin/perl

$str="TS1,TS2,TS3,TS4";

@original = split (/[,]/,$str);

print "@original";

$numTbArrays = int scalar(@original)/1000 + 2;

print "$numTbArrays";

foreach my $x (0..$numTbArrays-1)

{

if ($tbsArrayRef[$x])

{

@{$tbsArrayRef[$x]} = undef;

}

}

while (@original)

{

foreach my $y (0..$numTbArrays-1)

{

if (@original)

{

my $x = shift @original;

if ($y == ($numTbArrays-1))

{

$x = '\''.$x.'\'' . chr(10) . ',';

}

else

{

$x = '\''.$x.'\'' . chr(10) . ',';

}

push @{$tbsArrayRef[$y]}, $x;

}

}

}

print "--------";

print "@{$tbsArrayRef[1]}";

输出结果：

[oracle@host01 ~]$ perl perl1.pl

TS1 TS2 TS3 TS42--------'TS2'

, 'TS4'

### xttnewdatafiles.txt（第一步restore需要）

具体的过程可以在getxttnewdf函数中找到

生成xttnewdatafiles.txt的脚本内容如下：

SET TRIMSPOOL ON TAB OFF PAGES 0 EMB OFF LINES 32767 FEEDBACK OFF SERVEROUTPUT ON FORMAT TRUNCATE COLSEP "" RECSEP OFF DEFINE "&" VERIFY OFF

SELECT CASE

WHEN rn = 1

THEN '::' || ts\_name || CHR(10)

END

|| file#

|| ','

|| df\_name

FROM (

SELECT ROW\_NUMBER() OVER (PARTITION BY t.name ORDER BY d.file#) rn

, t.ts#

, t.name ts\_name

, d.file#

, '&1' || '/' || t.name || '\_' || d.file# || '.dbf' df\_name

, checkpoint\_change# ckpn

FROM v$tablespace t

, v$datafile d

WHERE t.ts# = d.ts# and t.name in ('TS1','TS2'));

用紫色标记的俩个位置，&1就是我们storageondest的路径，我参数文件中设置的是/home/oracle/xtt/oradata。第二个标紫的位置是我们要迁移的表空间的列表。

所以这里执行脚本的效果：

SQL> /

Enter value for 1: /home/oracle/xtt/oradata

::TS1

31,/home/oracle/xtt/oradata/TS1\_31.dbf

33,/home/oracle/xtt/oradata/TS1\_33.dbf

::TS2

32,/home/oracle/xtt/oradata/TS2\_32.dbf

再看下我们的xttnewdatafiles.txt：

[oracle@host01 backup\_Apr20\_Sat\_11\_01\_18\_484]$ cat xttnewdatafiles.txt

::TS1

31,/home/oracle/xtt/oradata/TS1\_31.dbf

33,/home/oracle/xtt/oradata/TS1\_33.dbf

::TS2

32,/home/oracle/xtt/oradata/TS2\_32.dbf

### generate\_batch\_tsoutput

最最核心的函数。

这个函数生成主要生成xttpreparesrc.sql文件，并通过过滤该sql文件的输出生成xttplan.txt。

我们们把这3个文件贴一下：

[oracle@host01 backup\_Apr20\_Sat\_11\_01\_18\_484]$ cat tsbkupmap.txt

::31,33:::1=TS1\_33\_2jtvf581\_1\_1.bkp

::32:::1=TS2\_32\_2ktvf585\_1\_1.bkp

[oracle@host01 backup\_Apr20\_Sat\_11\_01\_18\_484]$ cat incrbackups.txt

/home/oracle/xtt/dest\_backups/TS1\_33\_2jtvf581\_1\_1.bkp

/home/oracle/xtt/dest\_backups/TS2\_32\_2ktvf585\_1\_1.bkp

[oracle@host01 backup\_Apr20\_Sat\_11\_01\_18\_484]$ cat xttplan.txt

TS1::::12309768

31

33

TS2::::12309768

32

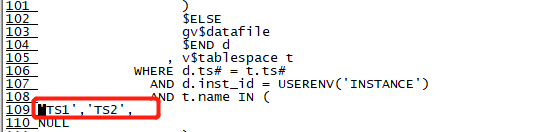
### xttpreparesrc.sql



首先，这个sql文件是0级backup+prepare阶段时使用的！

这个sql文件中，我们只需要修改109行的要迁移的表空间的名字。用逗号分隔，记得最后一个表空间名字后面要加个逗号，因为原sql里后面还跟了个NULL。

比如我修改成：



那么输出如下：

[oracle@host01 backup\_Apr20\_Sat\_11\_01\_18\_484]$ cat xttprepare.cmd

#PLAN:TS1::::12343255

backup for transport allow inconsistent incremental level 0 datafile

#NEWDESTDF:31,/home/oracle/xtt/oradata/TS1\_31.dbf

31

#PLAN:31

#NEWDESTDF:33,/home/oracle/xtt/oradata/TS1\_33.dbf

,33

#PLAN:33

#PLAN:TS2::::12343284

#NEWDESTDF:32,/home/oracle/xtt/oradata/TS2\_32.dbf

,32

#PLAN:32

format '/home/oracle/xtt/dest\_backups/%N\_%f\_%U.bkp';

我们执行完脚本，可以到生成的xttprepare.cmd下去找到我们输出结果。

### xttplan.txt

我们把上述我们修改了的xttpreparesrc.sql的输出结果中的“#PLAN:”的行都取出来，然后把“#PLAN:”去掉，我们就会发现，剩余的就是我们的xttplan.txt文件。

而在generate\_batch\_tsoutput（）函数中，也正是这样处理的。

### xttpreparenextiter.sql和xttplan.txt.new



这个文件就是再增量备份时使用的！

这个脚本修改方式也是跟前面一样，看下它的输出：

[oracle@host01 bkpincr\_Apr20\_Sat\_11\_06\_25\_71]$ cat xttprepare.cmd

TS1::::12429361

31

33

TS2::::12429361

32

其实这个就是xttplan.txt.new。

也就是说在增量备份之前，先把下一次xttplan.txt准备好。

### backincr

这个函数就是增量备份的核心函数，函数里面是通过读取xplan.txt得到scn号，然后根据这个scn把备份脚本的cmd文件🡪 rmanincr.cmd生成出来：

[oracle@host01 bkpincr\_Apr20\_Sat\_11\_06\_25\_71]$ cat rmanincr.cmd

set nocfau;

host 'echo ts::TS1';

backup for transport allow INCONSISTENT incremental from scn 12309768

tablespace 'TS1' format

'/home/oracle/xtt/dest\_backups/%U';

set nocfau;

host 'echo ts::TS2';

backup for transport allow INCONSISTENT incremental from scn 12309768

tablespace 'TS2' format

'/home/oracle/xtt/dest\_backups/%U';

### tsbkupmap.txt、incrbackups.txt（每次recover都要用）

真正生成tsbkupmap.txt、incrbackups.txt的函数是gentablespace\_backupmap。原理是通过rman的日志输出来进行文本过滤。

所以这俩文件虽然简单，但是目前由于v3脚本没有支持rmanlog的附带输出（虽然有参数rmanlog支持，但是人家脚本里明说了，该功能有问题，3.1才能使用，我们即使在参数文件中定义rmanlog，但是到了v3的脚本里，它会不管你定义的是什么，都把$rmanLog置为0），所以，真要是表空间多了，去构造这俩文件，还是有点儿难受的。

注意！backup for transport产生的备份集的备份信息是不记录在controlfile中的，也就是说，我们在源库的视图里是无法进行查询的。

## 目标库为PDB

因为是目标库，所以并不影响我们之前的操作流程，只需要在导入元数据的时候注意，平时怎么往pdb里导数据的，这里就怎么导。

如下是例子：

impdp system/oracle@pdbprod3 directory=yx logfile=xtt\_prod3.log \

network\_link=to\_pdbprod1 transport\_full\_check=no \

transport\_tablespaces=TS1,TS2 \

transport\_datafiles='/home/oracle/xtt/oradata/TS1\_31.dbf','/home/oracle/xtt/oradata/TS1\_33.dbf','/home/oracle/xtt/oradata/TS2\_32.dbf'

# V3版本源库是PDB

## 参数文件

多指定一个参数：

srcconnstr=sys/oracle@pdbprod1

其实我们前面截图的脚本中就可以看到类似如下的写法：

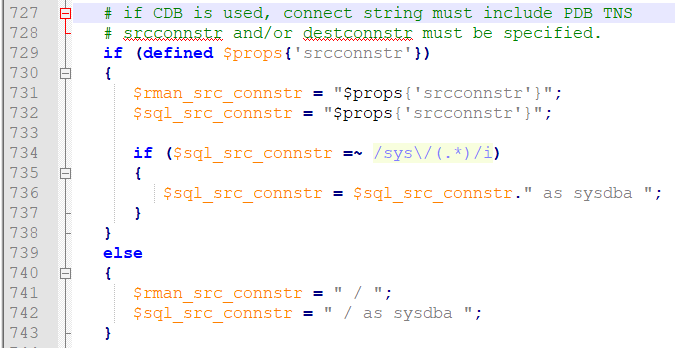
my $cmdStr = "sqlplus -L -s $sql\_src\_connstr \@$sqlfile ".

$props{'storageondest'};

my $sqlOutputStr = `$cmdStr`;

V3中所有的sqlplus的连接都是采取该方式，只是参数文件的介绍中并没有介绍该参数。那么怎么找到$sql\_src\_connstr对应的参数是哪个，很简单，到脚本开头定义变量的地方找。

最前面有：our $sql\_src\_connstr;的定义，然后ctrl+f找就行了：



这里可以说已经是写的非常清晰了。

## 实验

这个实验我都做了，从pdb🡪pdb、pdb🡪非cdb、非cdb🡪pdb（这个前面说过）。都没问题！

因为我们要理解这个脚本的本质，这个脚本能不能成功完全取决于rman的阶段能不能成功。

如果我们以连接串的方式连接数据库，其实是否是pdb已经完全没有区别了。

所以如果我们即使不涉及pdb，依然定义连接串，也是可以的。为了方便，一定要定义成：

srcconnstr=sys/密码@tnsname

我们在前面的截图也看到了，当连接sqlplus的时候，脚本会自动将"as sysdba"补充进去。