**表空间的一些小总结**

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 | 2019-03-08 | 初始版本 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. v$segstat和dba\_hist\_seg\_stat - 4 -](#_Toc2948410)

[2. LOB段对表空间的使用 - 8 -](#_Toc2948411)

[2.1 如果是个lob，那么一次占一个extent，还是占多个？ - 8 -](#_Toc2948412)

[2.2 那为一个值分配的extent一定要连续么？ - 11 -](#_Toc2948413)

[2.3 那是不是可不可以跨文件呢？ - 16 -](#_Toc2948414)

[2.4 总结 - 19 -](#_Toc2948415)

[3. 表空间不足的报错代码和trace - 19 -](#_Toc2948416)

[3.1 ORA-01691 - 19 -](#_Toc2948417)

[3.2 ORA-01652 - 20 -](#_Toc2948418)

[3.3 ORA-01653 - 22 -](#_Toc2948419)

[3.4 ORA-01654 - 24 -](#_Toc2948420)

[3.5 ORA-01658 - 24 -](#_Toc2948421)

[3.6 如何trace - 26 -](#_Toc2948422)

[3.6.1 开启 - 26 -](#_Toc2948423)

[3.6.2 关闭 - 28 -](#_Toc2948424)

# v$segstat和dba\_hist\_seg\_stat

这俩个视图是观察segment变化的视图，一个是实时的，一个是历史的。

我们先找一个表进行实验，先在yx用户下创建实验表：

SQL> create table test\_stat as select \* from dba\_objects;

Table created.

SQL> select object\_id from user\_objects where object\_name ='TEST\_STAT';

OBJECT\_ID

----------

78983

我们来看下v$segstat，下面的查询要在sys用户下查，因为涉及到ts$：

SQL> col object\_name for a20

SQL> col statistic\_name for a20

SQL> select c.name,b.object\_name,STATISTIC\_NAME,value from v$segstat a,dba\_objects b,ts$ c where STATISTIC\_NAME like 'space%' and a.obj#=b.object\_id and a.ts#=c.ts# and obj#=78983;

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

USERS TEST\_STAT space used 11665408

USERS TEST\_STAT space allocated 12582912

SQL> select sum(blocks)\*8192 from dba\_extents where segment\_name='TEST\_STAT';

SUM(BLOCKS)\*8192

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

12582912

SQL> select (12582912-11665408)/8192 from dual;

(12582912-11665408)/8192

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

112

SQL> select max(blocks) from dba\_extents where segment\_name='TEST\_STAT';

MAX(BLOCKS)

-----------

128

从上面我们可以看出：

space allocated是反映的是已经分配了的空间大小，且空间分配的最小单位是extent。实际使用的空间大小与已分配空间大小的差值，正好不足下一个要分配的extent，所以最后但是因为到数第二extent就已经是128个block的extent了，所以必须分配一个不小于128个block的extent。

接下来我们对表test\_stat做delete操作，不用提交：

SQL> delete from test\_stat;

73068 rows deleted.

SQL> select c.name,b.object\_name,STATISTIC\_NAME,value from v$segstat a,dba\_objects b,ts$ c where STATISTIC\_NAME like 'space%' and a.obj#=b.object\_id and a.ts#=c.ts# and obj#=78983;

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

USERS TEST\_STAT space used 0

USERS TEST\_STAT space allocated 12582912

我们看到已使用的空间为0，而已分配的空间没变，这就是delete不会降高水位的证明。

那么我们rollback，当然space used就又有值了，但是实验表明会有一点点的缩小，具体原理还没有研究，但暂时对于本次实验无关痛痒。

然后我们来测试truncate：

SQL> truncate table test\_stat;

Table truncated.

SQL> select c.name,b.object\_name,STATISTIC\_NAME,value from v$segstat a,dba\_objects b,ts$ c where STATISTIC\_NAME like 'space%' and a.obj#=b.object\_id and a.ts#=c.ts# and obj#=78983;

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

USERS TEST\_STAT space used 10144752

USERS TEST\_STAT space allocated 12582912

USERS TEST\_STAT space used 0

USERS TEST\_STAT space allocated -12517376

我们看到，oracle为该表另起了一批记录，同时，space allocated为负数，也就是说，space allocated只是想表达一种segment的变化动态。

我们再次往表里插入数据再观察：

SQL> insert into test\_stat select \* from dba\_objects;

73005 rows created.

SQL> /

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

USERS TEST\_STAT space used 10144752

USERS TEST\_STAT space allocated 12582912

USERS TEST\_STAT space used 11538102

USERS TEST\_STAT space allocated 0

现在我们看到space allocated变为了0，印证了我们的想法。那么我们对表进行一个ddl操作呢？oracle也会另一起一批记录么？

SQL> alter table test\_stat modify (object\_name varchar(100));

Table altered.

SQL> /

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

USERS TEST\_STAT space used 10144752

USERS TEST\_STAT space allocated 12582912

USERS TEST\_STAT space used -57510

USERS TEST\_STAT space allocated 0

我们看到改短了一列，space used变成了负数，但oracle并没有另起一批数据。也就是说这也是一个动态表达。

那么我们想看到底这个segment真实的total可以查看dba\_hist\_seg\_stat：

SQL> select SNAP\_ID,INSTANCE\_NUMBER,SPACE\_USED\_TOTAL,SPACE\_USED\_DELTA,SPACE\_ALLOCATED\_TOTAL,SPACE\_ALLOCATED\_DELTA from dba\_hist\_seg\_stat where obj#=78983;

SNAP\_ID INSTANCE\_NUMBER SPACE\_USED\_TOTAL SPACE\_USED\_DELTA SPACE\_ALLOCATED\_TOTAL SPACE\_ALLOCATED\_DELTA

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

487 1 11665408 11665408 12582912 12582912

489 1 10144752 -1520656 12582912 0

这里就先到这里不做深究了。

# LOB段对表空间的使用

我们先创建一个将近1m的文件：

SQL> set echo off

SQL> spool obj.txt

SQL> select \* from dba\_objects;

[oracle@host01 ~]$ tail -n 12000 obj.txt >object\_b.txt

[oracle@host01 ~]$ ll object\_b.txt

-rw-r--r-- 1 oracle oinstall 923634 Feb 28 14:50 object\_b.txt

[oracle@host01 ~]$ mv object\_b.txt obj.txt

然后我们建个表空间：

SQL> create tablespace test3 datafile '/home/oracle/test3.dbf' size 3m autoextend off uniform size 64k;

Tablespace created.

这是一个3m的表空间，每次只分配8个block的extent。

建立实验表：

SQL> create table t1(name clob) tablespace test lob(name) store AS BASICFILE (tablespace test3 disable storage in row nocache);

Table created.

## 如果是个lob，那么一次占一个extent，还是占多个？

插入一个lob值。

我们先创建目录：

SQL> create directory dir1 as '/home/oracle';

Directory created.

然后导入之前准备好的文件的内容：

declare

l\_clob clob;

l\_bfile bfile;

begin

insert into t1 values ( empty\_clob())

returning name into l\_clob;

l\_bfile := bfilename( 'DIR1', 'obj.txt' );

dbms\_lob.fileopen( l\_bfile );

dbms\_lob.loadfromfile( l\_clob, l\_bfile,dbms\_lob.getlength( l\_bfile ) );

dbms\_lob.fileclose( l\_bfile );

end;

/

PL/SQL procedure successfully completed.

现在我们确认lob段名：

col column\_name for a15

col segment\_name for a30

col index\_name for a30

col table\_name for a10

SQL> select table\_name,column\_name,segment\_name,tablespace\_name,index\_name from user\_lobs;

TABLE\_NAME COLUMN\_NAME SEGMENT\_NAME TABLESPACE\_NAME INDEX\_NAME

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

T1 NAME SYS\_LOB0000078969C00001$$ TEST3 SYS\_IL0000078969C00001$$

SQL> select object\_id,object\_name from user\_objects;

OBJECT\_ID OBJECT\_NAME

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

78969 T1

78971 SYS\_IL0000078969C00001$$

78970 SYS\_LOB0000078969C00001$$

…

然后我们来看SYS\_LOB0000078969C00001$$占用的extent的情况：

SQL> col name for a10

SQL> col object\_name for a25

SQL> select c.name, b.object\_name, STATISTIC\_NAME, value

2 from v$segstat a, dba\_objects b, ts$ c

3 where STATISTIC\_NAME like 'space%'

4 and a.obj# = b.object\_id

5 and a.ts# = c.ts#

6 and obj# = 78970;

NAME OBJECT\_NAME STATISTIC\_NAME VALUE

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

TEST3 SYS\_LOB0000078969C00001$$ space used 1867776

TEST3 SYS\_LOB0000078969C00001$$ space allocated 3276800

SQL> select extent\_id,blocks,BLOCK\_ID from dba\_extents where segment\_name='SYS\_LOB0000078969C00001$$' and owner='YX';

EXTENT\_ID BLOCKS BLOCK\_ID

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

0 8 8

1 8 24

2 8 32

3 8 40

4 8 48

5 8 56

6 8 64

7 8 72

8 8 80

9 8 88

10 8 96

11 8 104

12 8 112

13 8 120

14 8 128

15 8 136

16 8 144

17 rows selected.

我们看到即使字段很大，extent该怎么分配就怎么分配。而不是一定要一个值占一个extent。

## 那为一个值分配的extent一定要连续么？

我们从上一个实验看到，在有条件的情况下，extent是连续分配的，但这是必须的么？

我们来制造碎片：

SQL> create table t2 tablespace test3 as select \* from dba\_objects where rownum<100;

Table created.

SQL> create table t3 tablespace test3 as select \* from dba\_objects where rownum<1000;

Table created.

SQL> create table t4 tablespace test3 as select \* from dba\_objects where rownum<5000;

Table created.

我们先来看一下extent的占用情况：

SQL> select extent\_id,blocks,BLOCK\_ID from dba\_extents where segment\_name='T2' and owner='YX';

EXTENT\_ID BLOCKS BLOCK\_ID

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

0 8 152

SQL> c/2/3

1\* select extent\_id,blocks,BLOCK\_ID from dba\_extents where segment\_name='T3' and owner='YX'

SQL> /

EXTENT\_ID BLOCKS BLOCK\_ID

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

0 8 160

1 8 168

2 8 176

SQL> c/3/4

1\* select extent\_id,blocks,BLOCK\_ID from dba\_extents where segment\_name='T4' and owner='YX'

SQL> /

EXTENT\_ID BLOCKS BLOCK\_ID

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

0 8 184

1 8 192

2 8 200

3 8 208

4 8 216

5 8 224

6 8 232

7 8 240

8 8 248

9 8 256

10 8 264

11 8 272

12 rows selected.

我们看到现在extent分布都是连续的，我们看下还剩多少能分配的，因为是3m的表空间，所以如下计算最后一个block\_id：

SQL> select 3\*1024/8 from dual;

3\*1024/8

----------

384

SQL> select /\*+rule\*/ OWNER,

2 SEGMENT\_NAME,

3 SEGMENT\_TYPE,

4 TABLESPACE\_NAME,

5 EXTENT\_ID,

6 FILE\_ID,

7 BLOCK\_ID,

8 BLOCKS,

9 nvl(BLOCK\_1, 384 - block\_id-blocks) \* 8 diff

10 from (select b.\*,

11 lead(block\_id) over(partition by file\_id order by block\_id) - block\_id block\_1

12 from dba\_extents b

13 where file\_id = 8) a

14 where a.block\_1 <> a.blocks

15 or a.block\_1 is null;

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME EXTENT\_ID FILE\_ID BLOCK\_ID BLOCKS DIFF

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

YX T4 TABLE TEST3 11 8 272 8 832

我们看到只有文件尾部的832k的空间了，这当然是放不下将近1m的数据的。

那么我们现在制造碎片，把t3表删除，就可以构造出剩余空间大小够放数据，但是却不是连续的情况：

SQL> drop table t3 purge;

Table dropped.

我们再来看碎片分布：

SQL> select /\*+rule\*/ OWNER,

2 SEGMENT\_NAME,

3 SEGMENT\_TYPE,

4 TABLESPACE\_NAME,

5 EXTENT\_ID,

6 FILE\_ID,

7 BLOCK\_ID,

8 BLOCKS,

9 nvl(BLOCK\_1, 384 - block\_id-blocks) \* 8 diff

10 from (select b.\*,

11 lead(block\_id) over(partition by file\_id order by block\_id) - block\_id block\_1

12 from dba\_extents b

13 where file\_id = 8) a

14 where a.block\_1 <> a.blocks

15 or a.block\_1 is null;

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME EXTENT\_ID FILE\_ID BLOCK\_ID BLOCKS DIFF

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

YX T2 TABLE TEST3 0 8 152 8 256

YX T4 TABLE TEST3 11 8 272 8 832

可以看到有两部分不连续的空间，分别是256k和832k。

现在我们来插入一次lob：

declare

l\_clob clob;

l\_bfile bfile;

begin

insert into t1 values ( empty\_clob())

returning name into l\_clob;

l\_bfile := bfilename( 'DIR1', 'obj.txt' );

dbms\_lob.fileopen( l\_bfile );

dbms\_lob.loadfromfile( l\_clob, l\_bfile,dbms\_lob.getlength( l\_bfile ) );

dbms\_lob.fileclose( l\_bfile );

end;

/

PL/SQL procedure successfully completed.

SQL> select count(1) from t1;

COUNT(1)

----------

2

可以看到插入成功了，再查一下碎片：

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME EXTENT\_ID FILE\_ID BLOCK\_ID BLOCKS DIFF

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

YX SYS\_LOB0000078969C00001$$ LOBSEGMENT TEST3 32 8 376 8 0

我们看到，现在已经没有可用的空闲的block了。所有block都已经被分配了。那么就说明，为一个字段分配extent时，extent并不需要连续。

当然，原来属于t3的extent现在也确实属于lob了，这里不展示了。

## 那是不是可不可以跨文件呢？

我们知道一个extent不可以跨文件，对于一个值的俩个extent，可不可以在不同的文件。

很好办，我们再为实验表空间添加一个数据文件，然后把t4删了，制造碎片：

SQL> alter tablespace test3 add datafile '/home/oracle/test3\_02.dbf' size 1m;

Tablespace altered.

SQL> drop table t4 purge;

Table dropped.

我们又添加了一个1m的数据文件（这个file#是9）。之所以添加1m也是想看看，在有碎片的情况下，到底会不会优先使用碎片。

我们删了t4，再看碎片：

SQL> drop table t4 purge;

Table dropped.

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME EXTENT\_ID FILE\_ID BLOCK\_ID BLOCKS DIFF

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

YX SYS\_LOB0000078969C00001$$ LOBSEGMENT TEST3 19 8 176 8 832

YX SYS\_LOB0000078969C00001$$ LOBSEGMENT TEST3 32 8 376 8 0

我们看到，有一段832k的碎片。

现在我们再次插入lob：

SQL> declare

2 l\_clob clob;

3 l\_bfile bfile;

4 begin

5 insert into t1 values ( empty\_clob())

6 returning name into l\_clob;

7 l\_bfile := bfilename( 'DIR1', 'obj.txt' );

8 dbms\_lob.fileopen( l\_bfile );

9

10 dbms\_lob.loadfromfile( l\_clob, l\_bfile,dbms\_lob.getlength( l\_bfile ) );

11

12 dbms\_lob.fileclose( l\_bfile );

13 end;

14 /

PL/SQL procedure successfully completed.

SQL> select count(1) from t1;

COUNT(1)

----------

3

我们再来看碎片分布：

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME EXTENT\_ID FILE\_ID BLOCK\_ID BLOCKS DIFF

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

YX SYS\_LOB0000078969C00001$$ LOBSEGMENT TEST3 48 8 240 8 320

YX SYS\_LOB0000078969C00001$$ LOBSEGMENT TEST3 32 8 376 8 0

我们看到，碎片由原来的832k变为了320k。

再看9号数据文件：

SQL> select file\_id,extent\_id,blocks,block\_id from dba\_extents where segment\_name='SYS\_LOB0000078969C00001$$' and file\_id=9;

FILE\_ID EXTENT\_ID BLOCKS BLOCK\_ID

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

9 33 8 8

9 35 8 16

9 37 8 24

9 39 8 32

9 41 8 40

9 43 8 48

9 45 8 56

9 47 8 64

9 49 8 72

9 rows selected.

所以，同一个值得不同extent是可以在不同数据文件的。

## 总结

因为用字符串实验太费劲了，所以用lob实验，我暂时认为上述得出的结论是具有普适性的，不单单只针对lob。

# 表空间不足的报错代码和trace

首先准备一个小表空间：

SQL> create tablespace test3 datafile '/home/oracle/test3.dbf' size 1m;

Tablespace created.

SQL> alter database datafile '/home/oracle/test3.dbf' autoextend off;

Database altered.

## ORA-01691

这一节的实验数据来自第二节的实验：

SQL> declare

2 l\_clob clob;

3 l\_bfile bfile;

4 begin

5 insert into t1 values ( empty\_clob())

6 returning name into l\_clob;

7

8 l\_bfile := bfilename( 'DIR1', 'obj.txt' );

9 dbms\_lob.fileopen( l\_bfile );

10

11 dbms\_lob.loadfromfile( l\_clob, l\_bfile,dbms\_lob.getlength( l\_bfile ) );

12

13 dbms\_lob.fileclose( l\_bfile );

14 end;

15 /

PL/SQL procedure successfully completed.

SQL> /

declare

\*

ERROR at line 1:

ORA-01691: unable to extend lob segment YX.SYS\_LOB0000078969C00001$$ by 8 in tablespace TEST3

ORA-06512: at "SYS.DBMS\_LOB", line 954

ORA-06512: at line 11

LOB字段无法扩展，报错时1691。

## ORA-01652

如果是CREATE TABLE AS遇到表空间不足：

SQL> create table t1 tablespace test3 as select \* from dba\_objects;

create table t1 tablespace test3 as select \* from dba\_objects

\*

ERROR at line 1:

ORA-01652: unable to extend temp segment by 8 in tablespace TEST3

我们看到是1652无法扩展临时段。

因为CTAS建表的原理是先生成临时段，再变成永久段。

类似的原理还有move tablespace的过程：

SQL> create table t2 as select \* from dba\_objects;

Table created.

SQL> alter table t2 move tablespace test3;

alter table t2 move tablespace test3

\*

ERROR at line 1:

ORA-01652: unable to extend temp segment by 8 in tablespace TEST3

这里说个题外话，在move tablespace的过程中，开并行一定要谨慎，因为开并行就意味着多个进程同时去修改block，会产生大量的redo，如果开了归档，归档空间清理日志速度跟不上的话，会导致数据库因无法归档而hang住。解决办法就是把以有的归档备份走或删掉，说白了就是把归档空间空出来。

建立索引表空间不足也是这个：

SQL> create table t1 tablespace test3 as select \* from dba\_objects where rownum<4700;

Table created.

SQL> select extent\_id,block\_id,blocks from dba\_extents where segment\_name='T1' and owner='YX';

EXTENT\_ID BLOCK\_ID BLOCKS

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

0 8 8

1 16 8

2 24 8

3 32 8

4 40 8

5 48 8

6 56 8

7 64 8

8 72 8

9 80 8

10 88 8

11 96 8

12 rows selected.

SQL> create index idx\_t1 on t1(object\_id,object\_name,SUBOBJECT\_NAME) tablespace test3;

create index idx\_t1 on t1(object\_id,object\_name,SUBOBJECT\_NAME) tablespace test3

\*

ERROR at line 1:

ORA-01652: unable to extend temp segment by 8 in tablespace TEST3

## ORA-01653

SQL> create table t1 tablespace test3 as select \* from dba\_objects where rownum<5000;

Table created.

SQL> insert into t1 select \* from dba\_objects;

insert into t1 select \* from dba\_objects

\*

ERROR at line 1:

ORA-01653: unable to extend table YX.T1 by 8 in tablespace TEST3

正儿八经的数据表空间无法扩展，所有这些by xx，说的都是要扩展的extent需要的block数。所以根据报错，我们也可以知道它下一个extent想要多大。

还有一种情况，也是意外实验出来，但是这个还不确定：

SQL> insert into t1 select \* from dba\_objects where rownum<650;

649 rows created.

SQL> select extent\_id,block\_id,blocks from dba\_extents where segment\_name='IDX\_T1' and owner='YX';

EXTENT\_ID BLOCK\_ID BLOCKS

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

0 88 8

1 96 8

2 104 8

3 112 8

SQL> select extent\_id,block\_id,blocks from dba\_extents where segment\_name='T1' and owner='YX';

EXTENT\_ID BLOCK\_ID BLOCKS

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

0 8 8

1 16 8

2 24 8

3 32 8

4 40 8

5 48 8

6 56 8

7 64 8

8 72 8

9 80 8

10 120 8

11 rows selected.

一直在寻找合适的数据插入量，可以报出索引无法扩展段，但都失败了，然后选择了这个数据量来观察extent的变化。

一般来说，oracle是顺序分配block的，我们看到表的新的extent是从120号block开始分配，而索引的新的extent是从112号block开始分配（这里说明一下，在做这个insert之前，表的最后一个extent的blockid是80，索引的最后一个blockid是104）。

也就是说，索引的extent先分配好了，所以不管怎么做，都会报1653。包括我们直接插入一个很大的数据量，也是直接报1653。

注意，这是索引和表在同一个表空间的情况。但该结论仅限推测。

## ORA-01654

SQL> create table t1 as select \* from dba\_objects where rownum<10000;

Table created.

SQL> create index idx\_t1 on t1(object\_id,object\_name,SUBOBJECT\_NAME) tablespace test3;

Index created.

SQL> insert into t1 select \* from t1;

9999 rows created.

SQL> insert into t1 select \* from t1;

insert into t1 select \* from t1

\*

ERROR at line 1:

ORA-01654: unable to extend index YX.IDX\_T1 by 8 in tablespace TEST3

我们，当insert数据时导致的索引增加，会产生1654的报错。

## ORA-01658

SQL> create index idx\_t1 on t1(object\_id,object\_name,SUBOBJECT\_NAME) tablespace test3;

create index idx\_t1 on t1(object\_id,object\_name,SUBOBJECT\_NAME) tablespace test3

\*

ERROR at line 1:

ORA-01658: unable to create INITIAL extent for segment in tablespace TEST3

我们看这个报错，无法创建初始extent就会是1658。我们验证一下：

SQL> select extent\_id,block\_id,blocks from dba\_extents where segment\_name='T1' and owner='YX';

EXTENT\_ID BLOCK\_ID BLOCKS

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

0 8 8

1 16 8

2 24 8

3 32 8

4 40 8

5 48 8

6 56 8

7 64 8

8 72 8

9 80 8

10 88 8

11 96 8

12 104 8

13 112 8

14 120 8

15 rows selected.

SQL> select 1024/8 from dual;

1024/8

----------

128

确实是没有可分配的block给新的extent了。

## 如何trace

### 开启

想要trace这些报错，找到到底是什么sql导致的报错，可以如下设置event：

alter system set events='错误号 trace name errorstack forever,level 1';

举个例子，比如我们要跟踪导致index无法扩展的sql：

SQL> oradebug SETMYPID

Statement processed.

SQL> oradebug eventdump system

1691 trace name errorstack forever,level 1

1653 trace name errorstack forever,level 1

SQL> alter system set events='1654 trace name errorstack forever,level 1';

System altered.

SQL> oradebug SETMYPID

Statement processed.

SQL> oradebug eventdump system

1654 trace name errorstack forever,level 1

1691 trace name errorstack forever,level 1

1653 trace name errorstack forever,level 1

然后我们再次执行之前的insert：

SQL> insert into t1 select \* from t1;

insert into t1 select \* from t1

\*

ERROR at line 1:

ORA-01654: unable to extend index YX.IDX\_T1 by 8 in tablespace TEST3

我们看alert日志：

ORA-1654: unable to extend index YX.IDX\_T1 by 8 in tablespace TEST3

2019-03-08T14:20:34.105547+08:00

Errors in file /u01/app/oracle/diag/rdbms/prod4/PROD4/trace/PROD4\_ora\_90346.trc:

ORA-01654: unable to extend index YX.IDX\_T1 by 8 in tablespace TEST3

现在有了trace，注意，没开对应的event是不会有trace的。

在trace中我们找到了sql：

----- Error Stack Dump -----

ORA-01654: unable to extend index YX.IDX\_T1 by 8 in tablespace TEST3

----- Current SQL Statement for this session (sql\_id=9vp603v4vmjrn) -----

insert into t1 select \* from t1

还有一种看当前数据库设置了哪些event的方式：

set serveroutput on size 1000000

declare

event\_level number;

begin

for i in 1..100000 loop

sys.dbms\_system.read\_ev(i,event\_level);

if (event\_level > 0) then

dbms\_output.put\_line('Event '||to\_char(i)||' set at level '||

to\_char(event\_level));

end if;

end loop;

end;

/

Event 1653 set at level 1

Event 1654 set at level 1

Event 1691 set at level 1

PL/SQL procedure successfully completed.

### 关闭

要是想关掉该event的话如下：

SQL> alter system set events='1654 trace name errorstack off';

System altered.

SQL> oradebug eventdump system

1691 trace name errorstack forever,level 1

1653 trace name errorstack forever,level 1