

下面通过 undo 的一致性读分析 undo:

```
[oracle@localhost ~]$ lsb_release -a
```

```
LSB Version:      :core-3.1-ia32:core-3.1-noarch:graphics-3.1-ia32:graphics-3.1-noarch
```

```
Distributor ID: EnterpriseEnterpriseServer
```

```
Description:      Enterprise Linux Enterprise Linux Server release 5.5 (Carthage)
```

```
Release:          5.5
```

```
Codename:         Carthage
```

```
SQL> select * from v$version where rownum<2;
```

BANNER

```
-----  
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
```

```
SQL> create table t(id number,name varchar2(10));
```

表已创建。

已用时间: 00: 00: 00.15

```
SQL> set timing off;
```

```
SQL> show user;
```

USER 为 "HR"

```
SQL> insert into t values(1,'a');
```

已创建 1 行。

```
SQL> insert into t values(2,'b');
```

已创建 1 行。

```
SQL> commit;
```

提交完成。

```
SQL> update t set name='c' where id=1;
```

已更新 1 行。

```
SQL> select * from t;
```

```
      ID NAME  
-----  
1      c  
2      b
```

注意没提交。

重新打开一个 session:

```
SQL> select * from t;
```

ID	NAME
1	a
2	b

此时还是读取到修改之前的数据，这里的 a 是重 undo 里读取的，下面 dump 分析这个过程：

```
SQL> select t.*,rowid from t;
```

ID	NAME	ROWID
1	c	AAASunAAEAAAABuuAAA
2	b	AAASunAAEAAAABuuAAB

我们利用 oracle 提供的包，可以获得第一条数据所在的数据文件号及块号：

```
SQL> show user;
```

USER 为 "SYS"

```
SQL> select dbms_rowid.rowid_relative_fno(rowid) fno,dbms_rowid.rowid_block_number(rowid) bno
from hr.t;
```

FNO	BNO
4	7086
4	7086

此时我们可以 dump 4 号文件的第 7086 块：

```
SQL> alter system dump datafile 4 block 7086;
```

系统已更改。

下面是摘自部分转储文件：

*** 2015-04-20 15:26:34.151

Block header dump: 0x01001bae

Object id on Block? Y

seg/obj: 0x12ba7 csc: 0x00.6658e2 itc: 2 flg: E typ: 1 - DATA

brn: 0 bdba: 0x1001ba8 ver: 0x01 opc: 0

inc: 0 exflg: 0

Itl	Xid	Uba	Flag	Lck	Scn
/Fsc					

```

                                Uba => undo block address
0x01    0x0003.017.0000098a    0x00c009ac.02f0.24    C---    0    scn 0x0000.0066582c
0x02    0x0008.00d.00000a84    0x00c02e17.0258.28    ---- (表示事务锁定)    1 (锁定一条数
据)    fsc 0x0000.00000000
bdba:
0x01001bae

```

```
data_block_dump,data header at 0xa18264
```

```
=====
```

```

tsiz: 0x1f98
hsiz: 0x16
pbl: 0x00a18264
      76543210
flag=-----
ntab=1
nrow=2
frre=-1
fsbo=0x16
fseo=0x1f88
avsp=0x1f70
tosp=0x1f70
0xe:pti[0] nrow=2 offs=0
0x12:pri[0] offs=0x1f90
0x14:pri[1] offs=0x1f88
block_row_dump:

```

```
tab 0, row 0, @0x1f90
```

```

t1: 8 fb: --H-FL-- lb (锁标记): 0x2    cc: 2
      lb:lock byte

```

```
col 0: [ 2]    c1 02
```

```
col 1: [ 1]    63
```

第一行第二列已经被上锁了，被一个事务锁定。

=>这列是从 4 号文件 7086 块中读取的

63 代表是 c，实际已经被改了

```
tab 0, row 1, @0x1f88
```

```
t1: 8 fb: --H-FL-- lb: 0x0    cc: 2
```

```
col 0: [ 2]    c1 03
```

```
col 1: [ 1]    62
```

```
end_of_block_dump
```

```
End dump data blocks tsn: 4 file#: 4 minblk 7086 maxblk 7086
```

第二个 session 我们查出的第一行第二列却是 a，也是从 4 号文件 7086 个块上读取得，但是发现第二列已经被上锁，不能读取。

我们通过 uba 提供的地址，可以获取修改之前的数据 a 存放的位置：

uba:

0x00c02e17.0258.28

先把这个十六进制数转换成十进制数，oracle 提供的十进制包可以获取文件号及块号：

```
SQL> select to_number('00c02e17','XXXXXXXXXXXXXXXXXX') from dual;
```

```
TO_NUMBER('00C02E17','XXXXXXXXXXXXXXXXXX')
```

12594711

```
SQL> select dbms_utility.data_block_address_file(12594711) from dual;
```

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(12594711)
```

3

```
SQL> select dbms_utility.data_block_address_block(12594711) from dual;
```

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(12594711)
```

11799

在第二个 session 中，oracle 提示读取第一行第二列需要到 3 号文件上第 11799 号块上读取：

```
SQL> alter system dump datafile 3 block 11799;
```

系统已更改。

下面宅在部分转储文件：

uba: 0x00c02e17.0258.25 ctl max scn: 0x0000.00665307 prv tx scn: 0x0000.00665325

txn start scn: scn: 0x0000.006658a2 logon user: 91

prev brb: 12594708 prev bcl: 0

KDO undo record:

KTB Redo

op: 0x03 ver: 0x01

compat bit: 4 (post-11) padding: 1

op: Z

Array Update of 1 rows:

tabn: 0 slot: 0(0x0) flag: 0x2c lock: 0 ckix: 191

ncol: 2 nnew: 1 size: 0

KDO Op code: 21 row dependencies Disabled

xtype: XA xtype KDO_KDOM2 flags: 0x00000080 bdba: 0x01001bae hdba: 0x01001baa

itli: 2 ispac: 0 maxfr: 4858

vect = 3

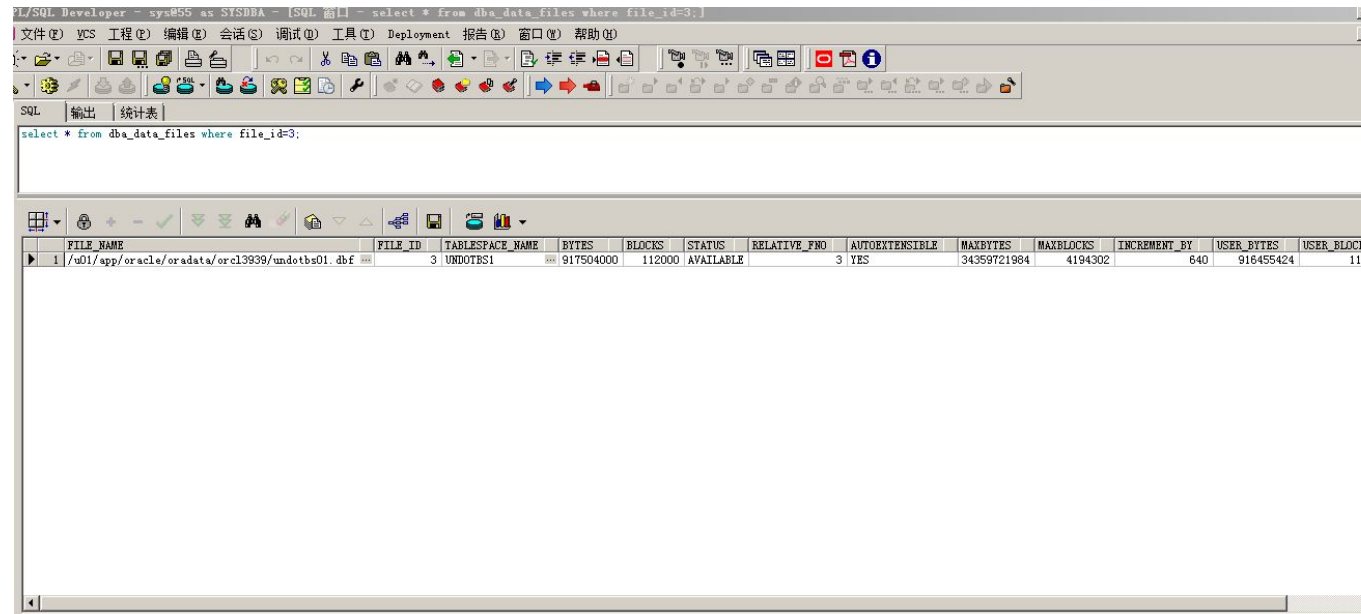
col 1: [1] 61

End dump data blocks tsn: 2 file#: 3 minblk 11799 maxblk 11799

所以通过 undo 读取了 61（代表 a）

我们查看数据文件 3 是什么文件类型：

```
select * from dba_data_files where file_id=3;
```



是 undo 数据文件。

```
SQL> show parameter undo;
```

NAME	TYPE	VALUE
undo_management	string	AUTO
undo_retention	integer	900
undo_tablespace	string	UNDOTBS1

默认的是 undotbs1 表空间，可以创建很多 undo 表空间，但是一个实例只用一个 undo 表空间。

undo 表空间是有 undo segments 组成，查看有多少 undo 段：

```
select * from dba_rollback_segs;
```

文件(F) 工程(E) 编辑(E) 会话(S) 调试(D) 工具(T) Deployment 报告(R) 窗口(W) 帮助(H)

SQL 输出 统计表

```
select * from dba_rollback_segs;
```

	SEGMENT_NAME	OWNER	TABLESPACE_NAME	SEGMENT_ID	FILE_ID	BLOCK_ID	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS	PCT_INCREASE	STATUS	INSTANCE_NUM	RELATIVE_FNO
1	SYSTEM	...	SYS	SYSTEM	...	0	1	128	114688	57344	1	32765
2	_SYSSMU1_592353410\$...	PUBLIC	UNDOTBS1	...	1	3	128	131072	65536	2	32765
3	_SYSSMU2_967517692\$...	PUBLIC	UNDOTBS1	...	2	3	144	131072	65536	2	32765
4	_SYSSMU3_1204390608\$...	PUBLIC	UNDOTBS1	...	3	3	160	131072	65536	2	32765
5	_SYSSMU4_1003442803\$...	PUBLIC	UNDOTBS1	...	4	3	176	131072	65536	2	32765
6	_SYSSMU5_538557934\$...	PUBLIC	UNDOTBS1	...	5	3	192	131072	65536	2	32765
7	_SYSSMU6_2897970769\$...	PUBLIC	UNDOTBS1	...	6	3	208	131072	65536	2	32765
8	_SYSSMU7_3517345427\$...	PUBLIC	UNDOTBS1	...	7	3	224	131072	65536	2	32765
9	_SYSSMU8_3901294357\$...	PUBLIC	UNDOTBS1	...	8	3	240	131072	65536	2	32765
10	_SYSSMU9_1735643669\$...	PUBLIC	UNDOTBS1	...	9	3	256	131072	65536	2	32765
11	_SYSSMU10_4131489474\$...	PUBLIC	UNDOTBS1	...	10	3	272	131072	65536	2	32765
12	_SYSSMU11_676451924\$...	PUBLIC	UNDO_#	...	21	9	128	131072	65536	2	2147483645
13	_SYSSMU12_420523439\$...	PUBLIC	UNDO_#	...	22	9	144	131072	65536	2	2147483645
14	_SYSSMU13_3923100083\$...	PUBLIC	UNDO_#	...	23	9	160	131072	65536	2	2147483645
15	_SYSSMU14_854438118\$...	PUBLIC	UNDO_#	...	24	9	176	131072	65536	2	2147483645
16	_SYSSMU15_2140685254\$...	PUBLIC	UNDO_#	...	25	9	192	131072	65536	2	2147483645
17	_SYSSMU16_4093222534\$...	PUBLIC	UNDO_#	...	26	9	208	131072	65536	2	2147483645
18	_SYSSMU17_4171122479\$...	PUBLIC	UNDO_#	...	27	9	224	131072	65536	2	2147483645
19	_SYSSMU18_952609167\$...	PUBLIC	UNDO_#	...	28	9	240	131072	65536	2	2147483645
20	_SYSSMU19_550118252\$...	PUBLIC	UNDO_#	...	29	9	256	131072	65536	2	2147483645
21	_SYSSMU30_9302802\$...	PUBLIC	UNDO_#	...	30	9	272	131072	65536	2	2147483645

sys955 AS SYSDBA 21 行被选择, 耗时 0.28 秒

注意上面的 owner 列，如果是 public，则该实例创建的 undo 段可以被数据库其他实例使用，但是 sys 表示的是私有 undo 段，只可以被该 undo 段创建者使用。

注意到 undo 段的状态了没，现在默认的 undo 表空间是 undotbs1，所以该 undo 段都是在线，undo_w 表空间的 undo 段都是离线。

我们可以通过修改参数 undo_tablespace 设置默认 undo 表空间。

oracle 对于处于 online 的 undo 段进行监视, 通过视图 v\$rollstat 查看:

文件(F) 工程(E) 编辑(E) 会话(S) 调试(D) 工具(T) Deployment 报告(R) 窗口(W) 帮助(H)

SQL 输出 统计表

```
select * from v$rollstat;
```

	USN	LATCH	EXTENTS	SSIZE	WRITES	XACTS	GETS	WAITS	OPTSIZE	HMMSIZE	SHRINKS	WRAPS	EXTENDS	AVESHRINK	AVEACTIVE	STATUS	CUREXT	CURBLK
1	0	0	6	385024	8112	0	116	0		385024	0	0	0	0	0	ONLINE	5	1
2	1	1	3	1171456	210662	0	592	0		1171456	0	0	0	0	0	ONLINE	2	62
3	2	2	3	1171456	282508	0	835	0		1171456	0	0	0	0	0	ONLINE	2	63
4	3	0	6	4317184	160836	0	590	0		4317184	0	0	0	0	0	ONLINE	4	50
5	4	1	4	2220032	175976	0	595	0		2220032	0	0	0	0	0	ONLINE	2	92
6	5	2	5	3268608	425806	0	877	0		3268608	0	1	0	0	104857	ONLINE	4	49
7	6	0	4	2220032	2392494	0	1314	2		4317184	1	4	0	2097152	191357	ONLINE	2	78
8	7	1	4	2220032	210870	0	586	0		2220032	0	1	0	0	104857	ONLINE	3	23
9	8	2	3	1171456	414724	0	730	1		2220032	1	3	0	1048576	101809	ONLINE	2	2
10	9	0	5	3268608	611646	0	1023	1		12705792	1	3	1	10485760	106970	ONLINE	2	43
11	10	1	4	2220032	341824	0	650	0		2220032	0	3	0	0	103201	ONLINE	2	22

上面总共有 11 条，usn 是 undo 段编号

一个事务使用一个 undo 段

下面执行一个事物：

SQL> update t set name='c' where id=1;

已更新 1 行。

select * from v\$transaction;

The screenshot shows the SQL*Plus interface with the command 'select * from v\$transaction;' entered. The output displays a single row of transaction details.

	ADDR	XIDUSN	XIDSLOT	XIDSQN	UBAFIL	UBABLK	UBASQN	UBAREC	STATUS	START_TIME	START_SCNB	START_SCNW	START_UEXT	START_UBAFIL	START_UBABLK	START_UBASQN	START_UBAREC
1	37FD0424	10	9	1859	3	4375	663	29	ACTIVE	04/20/15 16:44:20	6707052	0	2	3	4375	663	29

注意 XIDUSN 列表示的是 undo 段编号，此时该事务使用的是 10 号 undo 段

查看 10 号 undo 段：

select * from v\$rollstat;

The screenshot shows the SQL*Plus interface with the command 'select * from v\$rollstat;' entered. The output displays a table with 11 rows of undo segment statistics.

	USN	LATCH	EXTENTS	RSSIZE	WRITES	XACTS	GETS	WAITS	OPTSIZE	HWMSIZE	SHRINKS	WRAPS	EXTENDS	AVESHRINK	AVEACTIVE	STATUS	CUREXT	CURBLK
1	0	0	6	385024	8112	0	119	0		385024	0	0	0	0	0	ONLINE	5	1
2	1	1	3	1171456	214180	0	604	0		1171456	0	0	0	0	0	ONLINE	2	63
3	2	2	3	1171456	290734	0	853	0		1171456	0	0	0	0	0	ONLINE	2	63
4	3	0	6	4317184	162454	0	604	0		4317184	0	0	0	0	0	ONLINE	4	50
5	4	1	4	2220032	177918	0	599	0		2220032	0	0	0	0	0	ONLINE	2	92
6	5	2	5	3268608	441592	0	895	0		3268608	0	1	0	0	104857	ONLINE	4	51
7	6	0	4	2220032	2393556	0	1328	2		4317184	1	4	0	2097152	191357	ONLINE	2	78
8	7	1	4	2220032	218206	0	600	0		2220032	0	1	0	0	104857	ONLINE	3	24
9	8	2	3	1171456	419628	0	747	1		2220032	1	3	0	1048576	101809	ONLINE	2	3
10	9	0	5	3268608	613836	0	1039	1		12705792	1	3	1	10485760	106970	ONLINE	2	43
11	10	1	4	2220032	350378	1	661	0		2220032	0	3	0	0	103201	ONLINE	2	23

XACTS 列表示的是该 10 号 undo 段上具有活动的事务数量

此时修改默认 undo 表空间：

SQL> alter system set undo_tablespace=undo_w;

系统已更改。

SQL> show parameter undo;

NAME	TYPE	VALUE
undo_management	string	AUTO
undo_retention	integer	900
undo_tablespace	string	UNDO_W

此时查看 undo 段状态:

select * from dba_rollback_segs;

SEGMENT_NAME	OWNER	TABLESPACE_NAME	SEGMENT_ID	FILE_ID	BLOCK_ID	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS	PCT_INCREASE	STATUS	INSTANCE_NUM
1_SYSTEM	SYS	SYSTEM	0	1	128	114688	57344	1	32765		ONLINE	
2_SYSSMU1_592353410\$	PUBLIC	UNDOTBS1	1	3	128	131072	65536	2	32765		OFFLINE	
3_SYSSMU2_987517602\$	PUBLIC	UNDOTBS1	2	3	144	131072	65536	2	32765		OFFLINE	
4_SYSSMU3_120439060\$	PUBLIC	UNDOTBS1	3	3	160	131072	65536	2	32765		OFFLINE	
5_SYSSMU4_100344280\$	PUBLIC	UNDOTBS1	4	3	176	131072	65536	2	32765		OFFLINE	
6_SYSSMU5_538557934\$	PUBLIC	UNDOTBS1	5	3	192	131072	65536	2	32765		OFFLINE	
7_SYSSMU6_289797076\$	PUBLIC	UNDOTBS1	6	3	208	131072	65536	2	32765		OFFLINE	
8_SYSSMU7_351734542\$	PUBLIC	UNDOTBS1	7	3	224	131072	65536	2	32765		OFFLINE	
9_SYSSMU8_390129435\$	PUBLIC	UNDOTBS1	8	3	240	131072	65536	2	32765		OFFLINE	
10_SYSSMU9_173564368\$	PUBLIC	UNDOTBS1	9	3	256	131072	65536	2	32765		OFFLINE	
11_SYSSMU10_413148947\$	PUBLIC	UNDOTBS1	10	3	272	131072	65536	2	32765		ONLINE	
12_SYSSMU21_676451824\$	PUBLIC	UNDO_W	21	9	128	131072	65536	2	32765		ONLINE	
13_SYSSMU22_420523439\$	PUBLIC	UNDO_W	22	9	144	131072	65536	2	32765		ONLINE	
14_SYSSMU23_392310008\$	PUBLIC	UNDO_W	23	9	160	131072	65536	2	32765		ONLINE	
15_SYSSMU24_854438118\$	PUBLIC	UNDO_W	24	9	176	131072	65536	2	32765		ONLINE	
16_SYSSMU25_2140685254\$	PUBLIC	UNDO_W	25	9	192	131072	65536	2	32765		ONLINE	
17_SYSSMU26_4093222534\$	PUBLIC	UNDO_W	26	9	208	131072	65536	2	32765		ONLINE	
18_SYSSMU27_4171122479\$	PUBLIC	UNDO_W	27	9	224	131072	65536	2	32765		ONLINE	
19_SYSSMU28_352609167\$	PUBLIC	UNDO_W	28	9	240	131072	65536	2	32765		ONLINE	
20_SYSSMU29_550118252\$	PUBLIC	UNDO_W	29	9	256	131072	65536	2	32765		ONLINE	
21_SYSSMU30_5302802\$	PUBLIC	UNDO_W	30	9	272	131072	65536	2	32765		ONLINE	

undotbs1 里的 undo 段除了 10 号,其他的都处于 offline,因为它任被使用,事务结束后,自动变为 offline

通过查看视图:

select * from v\$rollstat;

USN	LATCH	EXTENTS	RSSIZE	WRITES	XACTS	GETS	WAITS	OPTSIZE	HMMSIZE	SHRINKS	WRAPS	EXTENDS	AVESHKINK	AVEACTIVE	STATUS	CUREXT	CUR
1	0	0	6	385024	18274	0	190	0	385024	0	0	0	0	0	ONLINE	5	
2	10	1	4	2220032	350378	1	664	0	2220032	0	3	0	0	103201	PENDING OF	2	
3	21	0	2	122880	708	0	12	0	122880	0	0	0	0	0	ONLINE	0	
4	22	1	2	122880	2428	0	16	0	122880	0	0	0	0	0	ONLINE	0	
5	23	2	2	122880	1424	0	12	0	122880	0	0	0	0	0	ONLINE	0	
6	24	0	2	122880	1522	0	14	0	122880	0	0	0	0	0	ONLINE	0	
7	25	1	2	122880	1406	0	16	0	122880	0	0	0	0	0	ONLINE	0	
8	26	2	2	122880	1440	0	14	0	122880	0	0	0	0	0	ONLINE	0	
9	27	0	2	122880	1216	0	14	0	122880	0	0	0	0	0	ONLINE	0	
10	28	1	2	122880	3274	0	18	0	122880	0	0	0	0	0	ONLINE	0	
11	29	2	2	122880	976	0	14	0	122880	0	0	0	0	0	ONLINE	0	
12	30	0	2	122880	1920	0	14	0	122880	0	0	0	0	0	ONLINE	0	

发现 10 号 undo 段状态是 pending offline (pending 在等待...期间)

会疑问,每个回滚段段上到底可以被几个事务使用呢?

SQL> show parameter roll

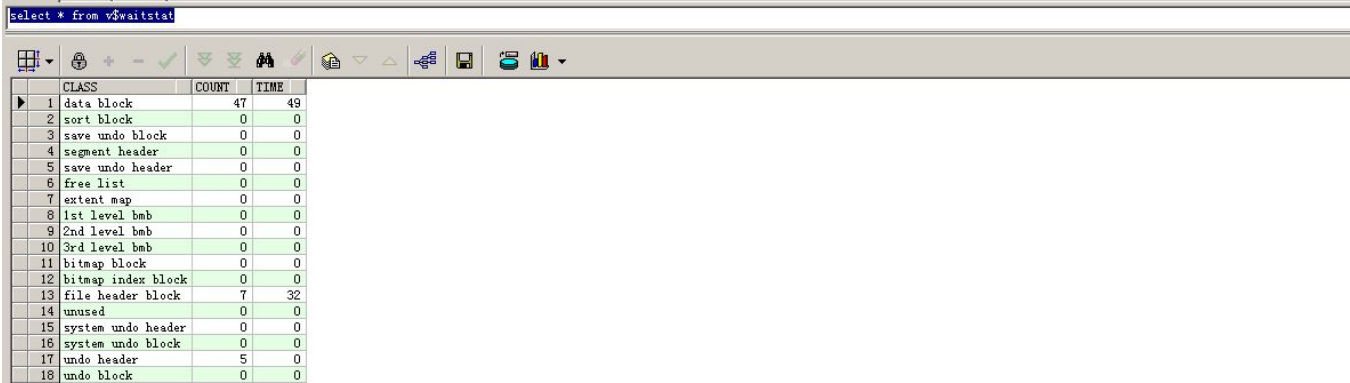
NAME	TYPE	VALUE
fast_start_parallel_rollback	string	LOW
rollback_segments	string	
transactions_per_rollback_segment	integer	5

可以被 5 个事务使用，但是这是在 u n d o 表空间没有自动管理之前，自从 u n d o 表空间自动管理后，该 parameter 不起作用。

一个 undo 段只能被一个事务使用，若 undo 被事务用完后，则 oracle background process smon 自动创建 undo 段。

如果一个回滚段被多个事务使用的话，undo 段头会有等待，影响并发性，我们可以通过视图 V\$WAITSTAT 查看等待事件：

```
select * from v$waitstat;
```



	CLASS	COUNT	TIME
1	data block	47	49
2	sort block	0	0
3	save undo block	0	0
4	segment header	0	0
5	save undo header	0	0
6	free list	0	0
7	extent map	0	0
8	1st level bmb	0	0
9	2nd level bmb	0	0
10	3rd level bmb	0	0
11	bitmap block	0	0
12	bitmap index block	0	0
13	file header block	7	32
14	unused	0	0
15	system undo header	0	0
16	system undo block	0	0
17	undo header	5	0
18	undo block	0	0

可以通过执行多个事务，模拟 smon 自动创建 undo 段，自行模拟试验。（smon 创建的 undo 段不会因为事务结束而回收）

下面看下一个很重要的参数：undo_retention 单位是秒（表示的是事务提交以后，放在 undo 里的数据保留的时间）

```
SQL> show parameter undo;
```

NAME	TYPE	VALUE
undo_management	string	AUTO
undo_retention	integer	900
undo_tablespace	string	UNDO_W

```
SQL> show parameter roll
```

很有名的 oracle 一个错误：ORA-01555（快照太旧）

实际情况下查询发生在修改之前，比较少。

出现这个错的可能情况：

undo 表空间太小

查询数据的时间过长（sql 查询性能差）

undo_rentention 太小

我们通过视图 dba_tablespaces, 引出一个参数：

```
SELECT TABLESPACE_NAME, RETENTION FROM DBA_TABLESPACES;
```

SQL 输出 统计表		
SELECT TABLESPACE_NAME,RETENTION FROM DBA_TABLESPACES;		
	TABLESPACE_NAME	RETENTION
1	SYSTEM	NOT APPLY
2	SYSaux	NOT APPLY
3	UNDOTBS1	NOGUARANTEE
4	TEMP	NOT APPLY
5	USERS	NOT APPLY
6	EXAMPLE	NOT APPLY
7	WANG	NOT APPLY
8	CHAO	NOT APPLY
9	UNDO_1	NOGUARANTEE
10	TEMP_WANG	NOT APPLY
11	BIG_FILE	NOT APPLY

retention 这列，undo 表空间缺省的是 NOGUARANTEE, 但是我们可以修改这个参数，强制保留。

```
SQL> alter tablespace undotbs1 retention GUARANTEE;
```

表空间已更改。

一定会保留 900 秒。

当让可以改 undo_retention

```
SQL> alter system set undo_retention=1200;
```

系统已更改。

上面列出的出现 ORA-01555 的三种情况，这三者之间有关系，比如增加了 undo_retention, 从而需要的 undo 表空间要更多。

oracle 里提供了 undo advisor (顾问)，在 Undo Advisor 的帮助下设置 undo 表空间的尺寸，来平衡这几者关系, 进入到 OEM:

```
[oracle@localhost ~]$ emctl start dbconsole
```

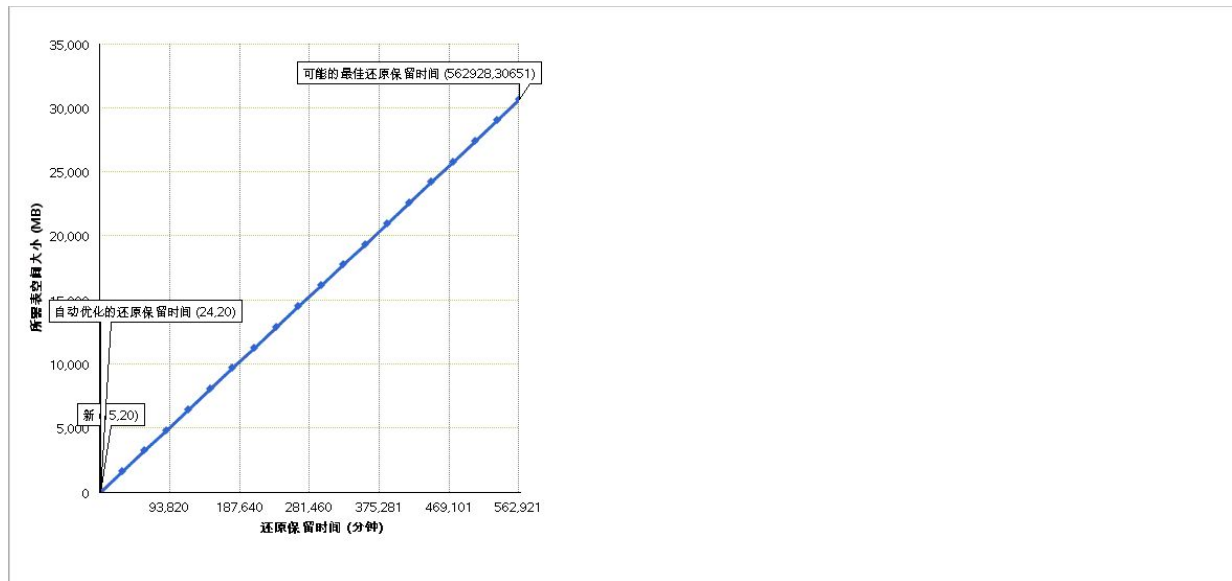
```
Oracle Enterprise Manager 11g Database Control Release 11.2.0.1.0
```

```
Copyright (c) 1996, 2009 Oracle Corporation. All rights reserved.
```

```
https://diy_os:1158/em/console/aboutApplication
```

```
Starting Oracle Enterprise Manager 11g Database Control .....
```

进入 em 后，在自动管理界面，提供了这一张图：



```
select * from v$undostat;
```

	BEGIN_TIME	END_TIME	UNDOTSN	UNDOBLKS	TXNCOUNT	MAXQUERYLEN	MAXQUERYID	MAXCONCURRENCY	UNXPSTALCNT	UNXPBLKRELCNT	UNXPBLKREUCNT	EXPSTALCNT	EXPBLKRELCNT
1	2015/4/20 18:35:16	2015/4/20 18:42:22	2	62	469	526	0rc4km05kgrb9	4	0	0	0	0	0
2	2015/4/20 18:25:16	2015/4/20 18:35:16	2	31	163	222	0rc4km05kgrb9	4	0	0	0	0	0
3	2015/4/20 18:15:16	2015/4/20 18:25:16	2	347	1099	822	0rc4km05kgrb9	4	0	0	0	0	0
4	2015/4/20 18:05:16	2015/4/20 18:15:16	2	12	248	1316	0rc4km05kgrb9	3	0	0	0	0	0
5	2015/4/20 17:55:16	2015/4/20 18:05:16	2	96	146	659	0rc4km05kgrb9	4	0	0	0	0	0
6	2015/4/20 17:45:16	2015/4/20 17:55:16	2	7	76	348	0rc4km05kgrb9	2	0	0	0	0	0
7	2015/4/20 17:35:16	2015/4/20 17:45:16	11	9	57	950	0rc4km05kgrb9	2	0	0	0	0	0
8	2015/4/20 17:25:16	2015/4/20 17:35:16	11	6	92	297	0rc4km05kgrb9	3	0	0	0	0	0
9	2015/4/20 17:15:16	2015/4/20 17:25:16	11	3	32	876	0rc4km05kgrb9	2	0	0	0	0	0
10	2015/4/20 17:05:16	2015/4/20 17:15:16	11	11	164	260	0rc4km05kgrb9	2	0	0	0	0	0
11	2015/4/20 16:55:16	2015/4/20 17:05:16	11	101	149	842	0rc4km05kgrb9	4	0	0	0	0	0
12	2015/4/20 16:45:16	2015/4/20 16:55:16	11	18	85	213	0rc4km05kgrb9	2	0	0	0	0	0
13	2015/4/20 16:35:16	2015/4/20 16:45:16	2	8	101	800	0rc4km05kgrb9	2	0	0	0	0	0
14	2015/4/20 16:25:16	2015/4/20 16:35:16	2	7	59	176	0rc4km05kgrb9	1	0	0	0	0	0
15	2015/4/20 16:15:16	2015/4/20 16:25:16	2	12	66	765	0rc4km05kgrb9	3	0	0	0	0	0
16	2015/4/20 16:05:16	2015/4/20 16:15:16	2	5	77	436	0rc4km05kgrb9	3	0	0	0	0	0
17	2015/4/20 15:55:16	2015/4/20 16:05:16	2	84	223	1020	0rc4km05kgrb9	4	0	0	0	0	0
18	2015/4/20 15:45:16	2015/4/20 15:55:16	2	8	59	390	0rc4km05kgrb9	5	0	0	0	0	0
19	2015/4/20 15:35:16	2015/4/20 15:45:16	2	9	61	965	0rc4km05kgrb9	3	0	0	0	0	0
20	2015/4/20 15:25:16	2015/4/20 15:35:16	2	7	56	363	0rc4km05kgrb9	2	0	0	0	0	0
21	2015/4/20 15:15:16	2015/4/20 15:25:16	2	8	156	927	0rc4km05kgrb9	2	0	0	0	0	0
22	2015/4/20 15:05:16	2015/4/20 15:15:16	2	7	65	314	0rc4km05kgrb9	2	0	0	0	0	0
23	2015/4/20 14:55:16	2015/4/20 15:05:16	2	69	138	902	0rc4km05kgrb9	4	0	0	0	0	0
24	2015/4/20 14:45:16	2015/4/20 14:55:16	2	4	70	261	0rc4km05kgrb9	2	0	0	0	0	0
25	2015/4/20 14:35:16	2015/4/20 14:45:16	2	15	56	861	0rc4km05kgrb9	4	0	0	0	0	0
26	2015/4/20 14:25:16	2015/4/20 14:35:16	2	6	86	250	0rc4km05kgrb9	2	0	0	0	0	0
27	2015/4/20 14:15:16	2015/4/20 14:25:16	2	7	56	811	0rc4km05kgrb9	3	0	0	0	0	0
28	2015/4/20 14:05:16	2015/4/20 14:15:16	2	13	192	504	0rc4km05kgrb9	4	0	0	0	0	0
29	2015/4/20 13:55:16	2015/4/20 14:05:16	2	109	191	0		4	0	0	0	0	0
30	2015/4/20 13:45:16	2015/4/20 13:55:16	2	6	52	468	0rc4km05kgrb9	2	0	0	0	0	0
31	2015/4/20 13:35:16	2015/4/20 13:45:16	2	6	115	695	0rc4km05kgrb9	2	0	0	0	0	0

每十分钟计算 undo 的数量。可以根据这个视图可以画出上述图形。