UNDO AND REDO IN ORACLE

INTRODUCTION TO UNDO

When a transaction tries to modify your data, immediately oracle copies the original data in undo segments before to modify it. The copy of the modified data collectively known as 'UNDO DATA'.

Why Oracle copies original data in undo segments ?

To provide read consistency and to roll back uncommitted transactions.

Oracle provides two methods of undo management.
Manual undo management & Automatic undo management.

Oracle uses undo tablespace or rollback segments to store old values.
The purpose of undo segment & rollback segment is same except method of creation and maintenance.

Oracle strongly recommends that run your database in **AUTOMATIC UNDO MANAGEMENT** mode to manage undo instead of using rollback segments. Space management for rollback segment is complex.

Earlier releases of oracle database used rollback segments to store undo information.
In manual undo management mode, Oracle is managing undo data through rollback segments.

1. Rollback segments allow users to perform rollback for DML operations.
2. It keeps the database to maintain read consistency among multiple transactions.
3. A single rollback segment can hold multiple transactions.
4. Transaction may assigned to rollback segments in a round robin fashion.

Oracle 9i has introduced automatic undo management which simplifies undo space management. To change your database to automatic undo management, you must create an undo tablespace and then change the UNDO_MANAGEMENT initialization parameter to AUTO.

```
SYS> show parameter undo;

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------
undo_management                      string      MANUAL
undo_retention                       integer     900
undo_tablespace                      string
```

```
SYS> create undo tablespace undotbs1 datafile
'/u01/app/oracle/oradata/crms/undotbs01.dbf' size 1g;

Tablespace created.

SQL> alter system set undo_management=AUTO scope=spfile;
System altered.

SQL> alter system set undo_tablespace=UNDOTBS1 scope=spfile;
System altered.

SQL> shut immediate;
...

SQL> startup;
...
```

UNDO AND REDO IN ORACLE

```
SYS> show parameter undo;

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

An active undo tablespace cannot be dropped.

You cannot create any other segments types (tables, index) in undo tablespaces.

A database can contain more than one undo tablespace but **only one can be active** at any time.

UNDO tablespaces normally used for following purposes.

```
Rollback transactions explicitly using ROLLBACK command.
Rollback transactions implicitly – automatic instance recovery.
Providing read consistency for SQL queries.
Recovering from logical corruptions using flashback features.
```

AUTOMATIC UNDO MANAGEMENT

Oracle provides a fully automated mechanism **(AUM)** to manage undo information and space. To enable AUM, you need to set **UNDO_MANAGEMENT=AUTO**. A default undo tablespace is creation at that time of Database creation. In My Opinion, an undo tablespace is an alternative to a rollback segment.

The undo tablespace is used for several features ROLLBACK, READ CONSISTENCY & FLASHBACK TECHNOLOGY. Starting from 10g the flashback feature using undo.

```
Flashback Query – (based on time).
Flashback Versions Query – (based on scn).
Flashback Transaction Query – (based on period)
Flashback Table – (based on time)
```

As I told, purpose of undo is placing copy of the original data in the undo tablespace as it existed before the modification. When a user performs an update or deletes operation, the earlier data will be placed in undo segments and then actual data is modified to a new value.

Purpose is to maintain read consistency by maintaining the past image of the data who are accessing the data at the same time that another user is changing it. When a transaction is rolled back, Oracle restores the earlier data from undo segments.

Oracle saves undo data at least until the transaction has been committed. Until this time the undo data is in **"active state"**. When active undo data is stored in the undo tablespace, Oracle will not allow to overwrite until the corresponding transaction has been committed.

```
ACTIVE      - No one can overwrite because it is supporting active transactions.
EXPIRED     - "OLD". Older than the UNDO RETENTION, and eligible to be overwritten as required.
UNEXPIRED   - "OLD". Old but within the UNDO RETENTION period as is eligible to be overwritten.
```

After a transaction is committed undo data is no longer needed for rollback or transaction recovery Purposes. However, for consistent read purposes, long running queries may require this old undo producing older images of data blocks.

UNDO AND REDO IN ORACLE

Flashback features also depend upon availability of older undo information. So it is recommended to retain the old undo information as long as possible.

> Flashback queries retrieve information from a specific point in time. As long as the undo data still exists for that time, the query can retrieve information exactly.

As we know committed undo data can be either expired or unexpired. Expired data can be overwritten by new transactions. When there is no space in the undo tablespace for new transactions, Oracle will overwrite the unexpired data, depending on how you configure the UNDO_RETENTION parameter.

UNDO RETENTION PERIOD

Once you enabled automatic undo management, there is always a current **undo retention period,** which means Oracle database attempts to retain old undo information before overwriting it.
The undo retention **default threshold value is 900** seconds.

EXPIRED – Undo information is older than the undo retention period.
UNEXPIRED – Undo information is less than the current retention period.
UNEXPIRED is usually retained for consistent read and flashback operations).

Oracle database automatically tunes the undo retention period based on undo tablespace size and system activity. You too optionally specify minimum undo retention period (in seconds) by setting the UNDO_RETENTION initialization parameter.

```
SQL> alter system set undo_retention=2800;
System altered.
```

The database makes its best effort to honor the specified minimum undo retention period. When available becomes short for new transactions, the oracle database begins to overwrite expired undo. Suppose all expired undo is overwritten, still undo tablespace has no space for new transactions then the database may begin to overwrite unexpired undo information.

HOW ORACLE IS USING UNDO RETENTION PERIOD

The undo retention parameter is ignored for a fixed size undo tablespace.
The database may overwrite unexpired undo information when space becomes low.

The database always tunes the undo retention period (to set best possible retention) based on system activity and undo tablespace size. ***(automatic tuning of undo retention).***

If the undo tablespace with **AUTOEXTEND** option enabled, the database attempts to honor the minimum undo retention period specified by UNDO_RETENTION.

> **The undo_retention parameter can be honored, if the undo tablespace has enough space.**
> **- It does not matter if the undo tablespace is AUTOEXTEND or FIXED.**

When available space is low in undo tablespace, instead of overwriting unexpired undo information, the tablespace auto extends. If **MAXSIZE** clause is specified for an auto-extending undo tablespace, once the max size is reached, the database may begin overwrite unexpired undo information.

In auto tuning undo retention, we discuss **undo period is set with & without automatic extension.** Oracle automatically tunes the undo retention period based on how the undo tablespace is configured.

UNDO AND REDO IN ORACLE

WITHOUT AUTO EXTEDABLE (FIXED SIZE UNDO TABLESPACE)

If the undo tablespace is fixed size, the database tunes the undo retention period (to set best possible retention) for the tablespace size and current system load. The tuned retention period can be significantly greater than the specified minimum retention period.

```
SYS> select tablespace_name, file_name, autoextensible from dba_data_files
where tablespace_name like 'UNDO%';

TABLESPACE_NAME                FILE_NAME                                AUT
------------------------------ ---------------------- -------------- -------
UNDOTBS                        /u01/app/oracle/oradata/undotbs01.dbf      NO

SYS> show parameter undo;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
undo_management                      string      AUTO
undo_retention                       integer     2800
undo_tablespace                      string      UNDOTBS

SYS> select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time, 'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------
26-SEP-15 12:22 26-SEP-15 12:22                2800
```

```
SQL> create table t1(no number, string_val varchar2(100));
Table created.

SQL> insert into tab1 select rownum,'ORACLE' from dual connect by level <= 4000000;
4000000 rows created.

SQL> update tab1 set string_val= 'ORACLE_DATABASE';
...
```

```
SYS> select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time,'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION  (for fixed size tablespace
--------------- --------------- -------------------
26-SEP-15 12:02 26-SEP-15 12:04               14244

SYS> / bears

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------
26-SEP-15 12:02 26-SEP-15 12:04               28488
```

AUTO EXTENDABLE UNDO TABLESPACE

If undo tablespace is configured with auto extend option, the database tunes the undo retention period to be somewhat longer than the longest query on the system at that time. Again this tuned retention can be greater than specified retention period.

```
SYS> select tablespace_name, file_name, autoextensible from dba_data_files
where tablespace_name like 'UNDO%';

TABLESPACE_NAME                FILE_NAME                                AUT
------------------------------ ---------------------------------------  -------
UNDOTBS1                        /u03/app/oracle/oradata/undotbs01.dbf    YES

SYS> show parameter undo;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
undo_management                      string      AUTO
undo_retention                       integer     2800
undo_tablespace                      string      UNDOTBS1
```

```
SYS> select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time, 'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------
26-SEP-15 20:34 26-SEP-15 20:34                2800

SQL> update t1 set string_val= 'ORACLE' where no >= 1 and no <= 1000000;
...

SQL> update t1 set string_val= 'ORACLE_DATABASE' where no >= 1000001 and no <= 2000000;
...

...

SQL> update t1 set string_val= 'ORACLE_RAC_DATABASE' where no >= 2000001 and no <= 4000000;
...
```

```
SYS> select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time,'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------
26-SEP-15 20:34 26-SEP-15 20:44                2800
26-SEP-15 20:44 26-SEP-15 20:54                2800
26-SEP-15 20:54 26-SEP-15 21:04                3015
26-SEP-15 21:04 26-SEP-15 21:14                3389
26-SEP-15 21:14 26-SEP-15 21:24                3765
...
```

EXAMPLE 2

```
SYS> update statement 1;
...

SYS> update statement 2;
...

SYS> update statement 3;
...

SYS> select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time,'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;

BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------

26-SEP-15 21:34 26-SEP-15 21:44               2800
26-SEP-15 21:44 26-SEP-15 21:54               2800
26-SEP-15 21:54 29-SEP-15 22:04               2800
26-SEP-15 22:22 26-SEP-15 22:32               1169
26-SEP-15 22:32 26-SEP-15 22:42               1095
26-SEP-15 21:42 26-SEP-15 22:52               1062
```

The conclusion is that the value of TUNED_UNDORETENTION can be equal, lower, or higher than the value of the UNDO_RETENTION initialization parameter and it depends on the duration of the active transactions as well as on your undo settings (UNDO_RETENTION, UNDO Tablespace size, and also the UNDO TABLESPACE MAXSIZE). You can see TUNED_UNDORETENTION > UNDO_RETENTION parameter value.

```
SYS> select count(*) from v$undostat where tuned_undoretention > 2800;
...
```

Values in V$UNDOSTAT, depends of workload and also UNCOMMITED transactions for long running queries. The tuned retention value may increase/decrease depending on the distribution of DML activity on your system. You can use the MIN, MAX, and AVG tuned retention to get a quick picture of how sufficient my fixed undo tablespace allocation is for your system.

```
SYS> select
     round(avg(tuned_undoretention) /60,2)  as "AVG MINUTES"
,    round(min(tuned_undoretention) /60,2)  as "MIN MINUTES"
,    round(max(tuned_undoretention) /60,2)  as "MAX MINUTES"
from v$undostat;
```

RETENTION GUARANTEE

Oracle 10g has introduced new feature (retention guarantee). It leads to get ORA-30036 error i.e. failing to extend Undo Rollback segment – Why so?

Once you specify the RETENTION GUARANTEE clause for the undo tablespace, then the database will never overwrite undo data whose age is less than the undo retention period. When space is low this leads to DML operation to become fail. A column named **RETENTION** from **DBA_TABLESPACES** view contains a value of guarantee for the undo tablespace.

```
SYS> select tablespace_name, retention from dba_tablespaces where tablespace_name like 'UNDO%';

TABLESPACE_NAME                RETENTION
------------------------------ -----------
UNDOTBS1                       NOGUARANTEE
```

You can enable **retention guarantee** by specifying **retention guarantee clause** at that time of database creation or create undo tablespace statement. Using alter tablespace statement you can done it.

```
SYS> select tablespace_name, retention from dba_tablespaces where tablespace_name like 'UNDO%';

TABLESPACE_NAME                RETENTION
------------------------------ -----------
UNDOTBS1                       GUARANTEE
```

To guarantee the success of long running queries or flashback operations. If retention guarantee is not enabled, then the database can overwrite unexpired undo when space is low.

WARNING: Enabling retention guarantee can cause multiple DML operations to fail.

ORA-30036 error occurs 'when no more space is left to store active undo'.
NOSPACEERRCNT column in V$UNDOSTAT is a good indication how many times this has occurred.

V$ UNDOSTAT

The **V$UNDOSTAT** data dictionary view provides system generated statistics, collected every 10 mins. Using this view you can monitor and tune UNDO space. **You can track the tuned undo retention period by querying the TUNED_UNDORETENTION column of the V$UNDOSTAT view.** It is also useful to determine whether you have allocated sufficient space to the UNDO tablespace for the current workload.

If an active transaction requires undo space and the undo tablespace does not have available space, then the system starts reusing unexpired undo space. This action can potentially cause some queries to fail with a **"snapshot too old"** message. How do we see O1555 error details?

We can identify if any "snapshot too old errors" were generated query the **SSOLDERRCNT** column.

```
SYS> select begin_time, ssolderrcnt from v$undostat;

BEGIN_TIME          SSOLDERRCNT
------------------- -----------
18-SEP-2015 14:30:40           3
18-SEP-2015 14:20:40           0
18-SEP-2015 14:10:40           0
18-SEP-2015 14:00:40           5
18-SEP-2015 13:50:40           3
18-SEP-2015 13:40:40           0
18-SEP-2015 13:30:40           0
18-SEP-2015 13:20:40           0
```

You can see a few of the ten minute intervals received more than zero i.e. "snapshot too old" errors. To avoid this, increase the value of UNDO_RENTENTION.

SSOLDERRCNT – Identifies the number of times the error ORA-01555 occurred.

NOSPACEERRCNT: Identifies the number of times space was requested in the undo tablespace and no free space available; all of the space in the undo tablespace was in use by active transactions.

UNXPSTEALCNT: Number of attempts to obtain undo space by stealing unexpired extents from other transactions.

```
SYS> select begin_time, ssolderrcnt, nospaceerrcnt from v$undostat;

BEGIN_TIME           SSOLDERRCNT    NOSPACEERRCNT   UNXPSTEALCNT
-------------------- -----------  ----------------- -------------
18-SEP-2015 14:30:40           3              0              0
18-SEP-2015 14:20:40           0              0              0
18-SEP-2015 14:10:40           0              0              0
18-SEP-2015 14:00:40           5              2              8
18-SEP-2015 13:50:40           3              0              1
18-SEP-2015 13:40:40           0              0              0
18-SEP-2015 13:30:40           0              0              0
18-SEP-2015 13:20:40           0              0              0
```

Once you find NON ZERO VALUES in these columns ***(ssolderrcnt, nospaceerrcnt)***, immediately increase the size of your undo tablespace. Whenever a transaction needs some more space to grow in the undo_tablespace it uses following options.

1) First it will try to allocate new extent in the free space or extend the tablespace, If AUTOEXTEND is on.

2) If it fails in step1 due to insufficient free space or auotextend off, then it will try to take the space from the expired extents in its own undo segments. Once it finds an expired extent, it will steal it and update the EXPSTEALCNT and EXPBLKREUCNT columns in V$UNDOSTAT.

3) If it fails in step2, then it will try to take the space from the expired extents in other undo segments. Once it finds an expired extent, it will steal it and update the EXPSTEALCNT and EXPBLKRELCNT columns in V$UNDOSTAT.

4) If it fails in step3, then it looks for unexpired extents in own undo segments. It must be the oldest one. If it finds unexpired extents, it reuses and updates UNXPSTEALCNT and UNXPBLKREUCNT columns in V$UNDOSTAT.

5) If it fails in step4, then it looks for unexpired extents in other undo segments. Probably, the oldest one. If it finds unexpired extents, it reuses and updates UNXPSTEALCNT and UNXPBLKRELCNT columns in V$UNDOSTAT.

6) If it fails in step5, then the transaction will fail with space error and updates NOSPACEERRCNT column in V$UNDOSTAT.

DATA DICTIONARY VIEWS

V$ROLLNAME	DBA_ROLLBACK_SEGS
V$ROLLSTAT	DBA_TABLESPACES
V$TRANSACTION	DBA_UNDO_EXTENTS
V$UNDOSTAT	DBA_HIST_UNDOSTAT

V$SESSION – Lists session info for each current session.
V$ROLLNAME – Lists the name of the online rollback segments.
V$TRANSACTION – Lists all active transactions in the database.

UNDO_SEGMENT

By default initially oracle creates 10 undo segments. In order to store more undo records the number of undo segments will be increased.

```
SYS> select tablespace_name, segment_name, segment_type from dba_segments
where tablespace_name like 'UNDO%';

TABLESPACE_NAME                SEGMENT_NAME                   SEGMENT_TYPE
------------------------------ ------------------------------ ------------------
UNDOTBS1                       _SYSSMU10_4131489474$          TYPE2 UNDO
UNDOTBS1                       _SYSSMU9_1735643689$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU8_3901294357$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU7_3517345427$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU6_2897970769$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU5_538557934$            TYPE2 UNDO
UNDOTBS1                       _SYSSMU4_1003442803$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU3_1204390606$           TYPE2 UNDO
UNDOTBS1                       _SYSSMU2_967517682$            TYPE2 UNDO
UNDOTBS1                       _SYSSMU1_592353410$            TYPE2 UNDO

10 rows selected.

SYS> select count(*) from dba_segments where tablespace_name='UNDOTBS1';
...

SYS> select * from v$rollname;

       USN NAME
---------- ------------------------------
         0   SYSTEM                    # related to system tablespace
         1   _SYSSMU1_592353410$
         2   _SYSSMU2_967517682$
         3   _SYSSMU3_1204390606$
         4   _SYSSMU4_1003442803$
         5   _SYSSMU5_538557934$
         6   _SYSSMU6_2897970769$
         7   _SYSSMU7_3517345427$
         8   _SYSSMU8_3901294357$
         9   _SYSSMU9_1735643689$
        10   _SYSSMU10_4131489474$

11 rows selected.
```

More segments are created as the number of concurrent transaction increases.
Automatic undo management attempts to have one transaction per undo segment.
We can verify XACTS column from V$ROLLSTAT. **XACTS** – number of active transactions.

UNDO AND REDO IN ORACLE

EXAMPLE 1

```
SYS> select * from v$rollstat where xacts=1;
no rows selected

SYS> select a.usn, a.xacts, b.name from v$rollstat a, v$rollname b
where a.usn = b.usn and a.xacts !=0;

no rows selected
```

```
SYSTEM> update scott.emp set ename='SONY' where empno=7996;
1 row updated.
```

```
SYS> select a.usn, a.xacts, b.name from v$rollstat a, v$rollname b
where a.usn = b.usn and a.xacts !=0;

USN        XACTS          NAME
---------- ----------  ----------------------
5          1           _SYSSMU5_538557934$

SYS> select * from v$rollstat where xacts=1;
...
```

You can see **XACTS=1,** which means undo segment 5 has allocated for my transaction.
Automatic undo management attempts to have one transaction per undo segment. i.e.
Transaction has used rollback **segment number 5,** and data segment name is **_SYSSMU5_4189375384$**

```
SYS> select t.xidusn, r.name, t.status, s.username, s.sid, s.serial#, s.status
  2  from v$transaction t, v$session s, v$rollname r
  3  where t.addr=s.taddr and t.xidusn=r.usn;

XIDUSN     NAME                   STATUS     USERNAME     SID    SERIAL#    STATUS
-------- ----------------------  ---------- ---------- ------- --------  ------
5          _SYSSMU5_538557934$     ACTIVE       SYSTEM      30      4       INACTIVE
```

COLUMN	DESCRIPTION	VIEW
XACTS	Number of active transactions	V$ROLLSTAT
USN	Rollback (UNDO) segment number	V$ROLLNAME
TADDR	Address of transaction state object	V$SESSION
STATUS	Status of the session (active, inactive)	V$SESSION
ADDR	Address of the transaction state object	V$TRANSACTION
STATUS	Status	V$TRANSACTION
XIDUSN	Undo segment number	V$TRANSACTION
OPTSIZE	Optimal size of the rollback segment.	V$ROLLSTAT
HWMSIZE	High watermark of rollback segment size.	V$ROLLSTAT
SHRINKS	Number of times the size of rollback segment decreases.	V$ROLLSTAT
WRAPS	Number of times rollback segment is wrapped.	V$ROLLSTAT
EXTENDS	Number of times rollback segment size is extended.	V$ROLLSTAT
STATUS	Rollback segment status.	V$ROLLSTAT
RSSIZE	Size in bytes of the rollback segment.	V$ROLLSTAT

You can get addition information using following Queries.

```
SYS> select a.name, b.optsize "Optimal_Size_for_Shrink", b.hwmsize HWM,
b.shrinks "Num_Shrinks", b.wraps "Num_wraps", b.extends "Num_Extends",
b.status FROM  v$rollname a, v$rollstat b WHERE a.usn = b.usn;
...
```

```
SYS> select a.name, b.extents, b.rssize, b.writes, b.xacts, b.gets, b.waits,
FROM v$rollname a, v$rollstat b WHERE a.usn = b.usn;
...
```

```
# TO GET UNDO SEGMENTS STATUS DETAILS

SYS> select status,
  round(sum_bytes / (1024*1024), 0) as MB,
  round((sum_bytes / undo_size) * 100, 0) as PERC
from
(
  select status, sum(bytes) sum_bytes
  from dba_undo_extents
  group by status
),
(
  select sum(a.bytes) undo_size
  from dba_tablespaces c
    join v$tablespace b on b.name = c.tablespace_name
    join v$datafile a on a.ts# = b.ts#
  where c.contents = 'UNDO'
    and c.status = 'ONLINE'
);

STATUS                              MB       PERC
------------------------------ ---------- ----------
ACTIVE                                596         38
EXPIRED                                25          2
UNEXPIRED                             864         58
```

MONITOR WHO IS USING UNDO

```
SYS> select
   s.sid, s.serial#, osuser, terminal, program, sql_id, module,
   s.username, r.name "RBS_NAME",
   t.start_time, t.used_ublk  "Undo_Blocks",
   t.used_urec  "Undo_Recs"
FROM
   v$session s, v$transaction t, v$rollname r
WHERE
t.addr = s.taddr and
r.usn  = t.xidusn;
...
```

UNDO AND REDO IN ORACLE

V$TRANSACTION links with V$SESSION will show current used undo blocks for ongoing transactions.

```
# VERIFY SEGMENT AND UNDO BLOCKS

SYS> SELECT a.sid, a.username, b.used_urec, b.used_ublk
FROM v$session a, v$transaction b
WHERE a.saddr = b.ses_addr
ORDER BY b.used_ublk DESC;                                    Or

SQL> select substr(a.os_user_name,1,8) "OS_User",
substr(a.oracle_username,1,8) "DB_User",
substr(b.owner,1,8) "Schema",
         substr(b.object_name,1,25)   "Object_Name",
substr(b.object_type,1,10) "Type",
substr(c.segment_name,1,5) "RBS",
substr(d.used_urec,1,20) "# of Records",
substr(d.used_ublk,1,20) "# of undo blocks used"
FROM
     v$locked_object    a,
     dba_objects        b,
     dba_rollback_segs  c,
     v$transaction      d,
     v$session          e
WHERE
    a.object_id     = b.object_id
    And a.xidusn    = c.segment_id
    And a.xidusn    = d.xidusn
    And a.xidslot   = d.xidslot
    And d.addr      = e.taddr;
..
```

In my opinion, space for undo segments is dynamically allocated, consumed, freed, and reused – controlled by an Oracle Database.

WHAT IS REDO

The REDO LOG can consist of two parts. The **Online redo log & Archived redo log.**

RLBC - committed transactions that are not yet written to the redo log files.

When a transaction is **committed**, the transaction's details in the **redo log buffer** **(RLBC)** is written to a online redo log file by LGWR background process. **LGWR** writes redo info at following situations.

When a user commits a transaction.
When Redo log buffer becomes 1/3 full.
If Redo buffer contains 1 MB of change records.
When log switch occurs.

Each individual redo log is assigned to a group. Oracle writes to only one online redo log group at a time. Once the online redo log(s) in that group are filled then Oracle starts to write next online redo log group and so on in a circular fashion. Oracle uses these online redo log group only for **recovery**. An Oracle database must have at least two redolog files. Logfiles can be multiplexed on multiple disks.

UNDO AND REDO IN ORACLE

Undo is stored into the undo tablespace (undo segments) and is accessible to the transaction. Redo is stored outside of the database and is inaccessible to the transaction but redo's are required in case if you do database recovery operations.

USES OF THE REDOLOG

Protection against the data loss. After an instance failure, online redo log files are used to recover commited data yet not written to the datafiles.

```
# REDO GENERATED BY USER SESSIONS

SYS> select v$session.sid, username, value redo_size
from v$sesstat, v$statname, v$session
where v$sesstat.STATISTIC# = v$statname.STATISTIC#
and v$session.sid = v$sesstat.sid
and name = 'redo size'
and value > 0
and username is not null
order by value;
...

# WHO GENERATES MORE REDO

SELECT s.sid, s.username, s.program, t.value "redo blocks written"
  FROM v$session s, v$sesstat t
 WHERE s.sid = t.sid
   AND t.value != 0
   AND t.statistic# = (select statistic# from v$statname WHERE name = 'redo size')
 ORDER BY 4;
 ...

# CHECK REDO GENERATED ONLY FOR MY SESSION

SCOTT> select value redo_size
 from v$mystat, v$statname
 where v$mystat.STATISTIC# = v$statname.STATISTIC#
 and name = 'redo size';
 ...

# CURRENT STATUS OF REDOLOG

SYS> select thread#, group#, sequence#, bytes, members, archived, status,
first_change#, to_char(first_time,'dd-mon-yyyy hh24:mi:ss') first_time
from sys.v_$log order by thread#, group#;
...

# REDOLOG WRITER PROCESS

SYS> select spid from v$process where program like '%LGWR%';
...

SYS> ! ps -ef | grep lgwr
...
```

UNDO AND REDO IN ORACLE

```
# LISTS EACH MEMBER OF EACH ONLINE REDOLOG GROUP

SYS> select * from v$log;

...

# REDOLOG FILE_NAME INFO

SYS> select * from v$logfile;

...
```

STATUS FROM V$LOG & V$LOGFILE

A log group can be in one of four status.

CURRENT	Current redo log, the redo log is active.
ACTIVE	Log is active, but it is NOT current log. Required for instance recovery.
INACTIVE	Log is no longer needed for instance recovery and can be overwritten.
UNUSED	Redo log just added or just after a RESETLOGS.

A log file can be in one of four status.

INVALID	File is corrupt or missing. - (inaccessible)
STALE	File is never been used.
DELETED	File is no longer used.

VIEWS FOR REDOLOG

V$LOG	Displays the redo file info from the control file
V$LOGFILE	Identifies redo log groups and members and member status.
V$LOG_HISTORY	Contains log history information.
V$archived_log	Contains archived logs information.

CLEARING THE REDOLOG FILE/GROUPS

When database is open, a redo log file might be corrupted then the database will stop responding. In this situation the ***alter database clear logfile...*** **SQL** statement can be used to reinitialize the file without shutting down the database.

```
SYS> alter database clear logfile group <group_number>;
SYS> alter database clear logfile group 1;
```

In some other situation, there may be a chance that the current redo log file got corrupt before it was successfully archived. In those situations we can clear the unarchived redo log group using following statement. This statement clears the corrupted redo logs and avoids archiving them.

```
SYS> alter database clear unarchived logfile group <group_number>;
SYS> alter database clear unarchived logfile group 1;
```

Once you clear an unarchived redo log file, you should make another backup of the database.