12

# Managing Undo Data

# Objectives

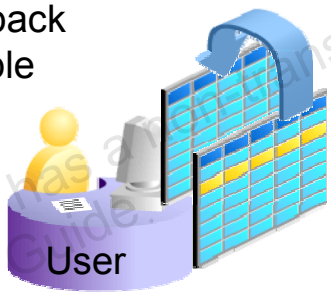
After completing this lesson, you should be able to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo
- Use the Undo Advisor

# Undo Data: Overview

Undo data is:

- A record of the action of a transaction
- Captured for *every* transaction that changes data
- Retained at least until the transaction is ended
- Used to support:
  - Rollback operations
  - Read-consistent queries
  - Oracle Flashback Query, Oracle Flashback Transaction, and Oracle Flashback Table
  - Recovery from failed transactions










The Oracle Database server saves the old value (undo data) when a process changes data in a database. It stores the data as it exists before modifications. Capturing undo data enables you to roll back your uncommitted data. Undo supports read-consistent and flashback queries. Undo can also be used to “rewind” (flash back) transactions and tables.

Read-consistent queries provide results that are consistent with the data as of the time a query started. For a read-consistent query to succeed, the original information must still exist as undo information. If the original data is no longer available, you receive a “Snapshot too old” error (`ORA-01555`). As long as the undo information is retained, the Oracle Database server can reconstruct data to satisfy read-consistent queries.

Flashback queries purposely ask for a version of the data as it existed at some time in the past. As long as undo information for that past time still exists, flashback queries can complete successfully. Oracle Flashback Transaction uses undo to create compensating transactions, to back out a transaction and its dependent transactions. With Oracle Flashback Table, you can recover a table to a specific point in time.

Undo data is also used to recover from failed transactions. A failed transaction occurs when a user session ends abnormally (possibly because of network errors or a failure on the client computer) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes or you issue the `SHUTDOWN ABORT` command.

In case of a failed transaction, the safest behavior is chosen, and the Oracle Database server reverses all changes made by a user, thereby restoring the original data.

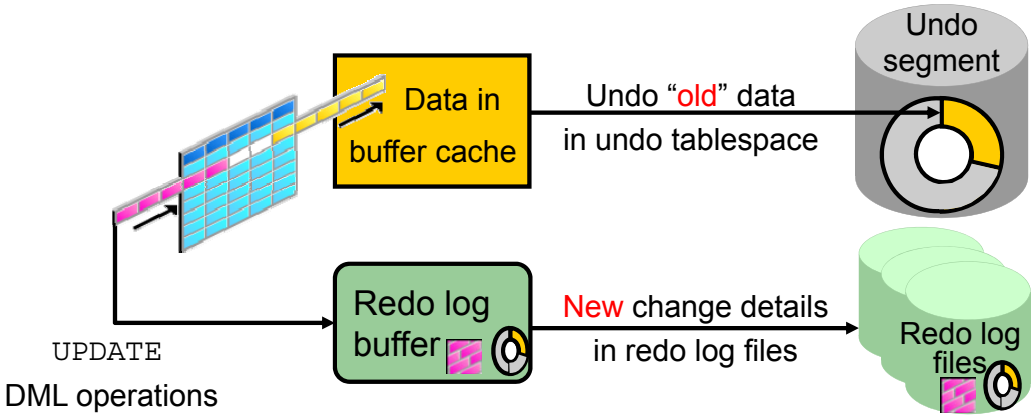
Undo information is retained for all transactions, at least until the transaction is ended by one of the following:

- User undoes a transaction (transaction rolls back).
- User ends a transaction (transaction commits).
- User executes a DDL statement, such as a `CREATE`, `DROP`, `RENAME`, or `ALTER` statement. If the current transaction contains any DML statements, the database server first commits the transaction and then executes and commits the DDL as a new transaction.
- User session terminates abnormally (transaction rolls back).
- User session terminates normally with an exit (transaction commits).

The amount of undo data that is retained and the time for which it is retained depend on the amount of database activity and the database configuration.

**Note:** Oracle Flashback Transaction leverages the online redo logs to mine the appropriate undo SQL for execution. It only uses undo as an artificial time boundary, to determine a redo mining start time for the target transaction, if a transaction start time is not supplied in the flashback transaction invocation.

# Transactions and Undo Data




- Each transaction is assigned to only one undo segment.
- An undo segment can service more than one transaction at a time.



When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `V$TRANSACTION` dynamic performance view.

Undo segments are specialized segments that are automatically created by the database server as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

**Note:** Parallel DML and DDL operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see the *Oracle Database Administrator's Guide*.

# Storing Undo Information

- Undo information is stored in undo segments, which are stored in an undo tablespace.
- Undo tablespaces:
  - Are used only for undo segments
  - Have special recovery considerations
  - May be associated with only a single instance
  - Require that only one of them be the current writable undo tablespace for a given instance at any given time







Undo segments can exist only in a specialized form of tablespace called an *undo tablespace*. (You cannot create other segment types, such as tables, in the undo tablespace.)

The Database Configuration Assistant (DBCA) automatically creates a smallfile undo tablespace. You can also create a bigfile undo tablespace. However, in a high-volume online transaction processing (OLTP) environment with many short concurrent transactions, contention could occur on the file header. An undo tablespace, stored in multiple data files, can resolve this potential issue.

Although a database may have many undo tablespaces, only one of them at a time can be designated as the current undo tablespace for any instance in the database.

Undo segments are automatically created and always owned by `SYS`. Because the undo segments act as a circular buffer, each segment has a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 KB block size).

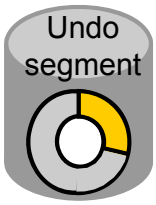
Undo tablespaces are permanent, locally managed tablespaces with automatic extent allocation. They are automatically managed by the database.

Because undo data is required to recover from failed transactions (such as those that may occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the `MOUNT` state.

# Comparing Undo Data and Redo Data



| | Undo | Redo |
|---|---|---|
| Record of | How to undo a change | How to reproduce a change |
| Used for | Rollback, read consistency, flashback | Rolling forward database changes |
| Stored in | Undo segments | Redo log files |









Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed if there is the need to undo a change, and this occurs for read consistency and rollback. Redo data is needed if there is the need to perform the changes again, in cases where they are lost for some reason. Undo block changes are also written to the redo log.

The process of committing entails a verification that the changes in the transaction have been written to the redo log file, which is persistent storage on the disk, as opposed to memory. In addition, the redo log file is typically multiplexed. As a result, there are multiple copies of the redo data on the disk. Although the changes may not yet have been written to the data files where the table's blocks are actually stored, writing to the persistent redo log is enough to guarantee consistency of the database.

Suppose that a power outage occurs just before committed changes have been reflected in the data files. This situation does not cause a problem because the transaction has been committed. When the system starts up again, it is able to roll forward any redo records that are not yet reflected in data files at the time of the outage.

# Managing Undo

Automatic undo management:

- Fully automated management of undo data and space in a dedicated undo tablespace
- For all sessions
- Self-tuning in `AUTOEXTEND` tablespaces to satisfy long-running queries
- Self-tuning in fixed-size tablespaces for best retention

DBA tasks in support of Flashback operations:

- Configuring undo retention
- Changing undo tablespace to a fixed size
- Avoiding space and “snapshot too old” errors

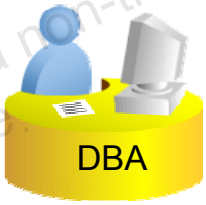


The Oracle Database server provides *automatic undo management*, which is a fully automated mechanism for managing undo information and space in a dedicated undo tablespace for all sessions. The system automatically tunes itself to provide the best possible retention of undo information. More precisely, the undo retention period for auto-extending tablespaces is tuned to be slightly longer than the longest-running active query. For fixed-size undo tablespaces, the database dynamically tunes for best possible retention.

Although by default the Oracle Database server manages undo data and space automatically, you may need to perform some tasks if your database is using Flashback operations. The administration of undo should prevent space errors, the use of too much space, and “Snapshot too old” errors.

# Configuring Undo Retention

- UNDO_RETENTION specifies (in seconds) how long already committed undo information is to be retained.
- Set this parameter when:
  - The undo tablespace has the AUTOEXTEND option enabled
  - You want to set undo retention for LOBs
  - You want to guarantee retention






The UNDO_RETENTION initialization parameter specifies (in seconds) the low threshold value of undo retention. Set the minimum undo retention period for the auto-extending undo tablespace to be as long as the longest expected Flashback operation. For auto-extending undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to meet the undo requirements of the queries. But this autotuned retention period may be insufficient for your Flashback operations.

For fixed-size undo tablespaces, the system automatically tunes for the best possible undo retention period on the basis of undo tablespace size and usage history; it ignores UNDO_RETENTION unless retention guarantee is enabled. So for automatic undo management, the UNDO_RETENTION setting is used for the three cases listed in the slide. In cases other than these three, this parameter is ignored.

# Categories of Undo

| Category | Description |
|---|---|
| Active: Uncommitted undo information | Supports an active transaction and is never overwritten |
| Unexpired: Committed undo information | Required to meet the undo retention interval |
| Expired: Expired undo information | Overwritten when space is required for an active transaction |

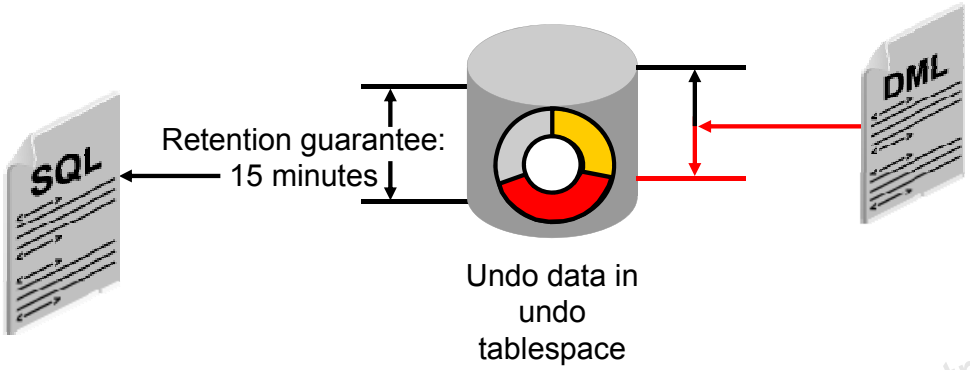


Undo information is divided into three categories:

- Uncommitted undo information (Active): Supports a currently running transaction, and is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.
- Committed undo information (Unexpired): Is no longer needed to support a running transaction but is still needed to meet the undo retention interval. It is also known as “unexpired” undo information. Committed undo information is retained when possible without causing an active transaction to fail because of lack of space.
- Expired undo information (Expired): Is no longer needed to support a running transaction. Expired undo information is overwritten when space is required by an active transaction.

# Guaranteeing Undo Retention

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```




`SELECT` statements running 15 minutes or less are always satisfied.

A transaction will fail if it generates more undo than there is space.

**Note:** This example is based on an `UNDO_RETENTION` setting of 900 seconds (15 minutes).



The default undo behavior is to overwrite the undo information of committed transactions that has not yet expired rather than to allow an active transaction to fail because of lack of undo space.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail.

`RETENTION GUARANTEE` is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements. The syntax to change an undo tablespace to guarantee retention is:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

To return a guaranteed undo tablespace to its normal setting, use the following command:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in the following error:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for undo tablespace
```

# Changing an Undo Tablespace to a Fixed Size

- Rationale:
  - Supporting Flashback operations
  - Limiting tablespace growth
- Steps:
  1. Run regular workload.
  2. Self-tuning mechanism establishes minimum required size.
  3. (Optional) Use the Enterprise Manager Cloud Control Undo Advisor, which calculates required size for future growth.
  4. (Optional) Change undo tablespace to a fixed size.







You might have two reasons for changing the undo tablespace to a fixed size: to support Flashback operations (where you expect future use of the undo) or to prevent the tablespace from growing too large.

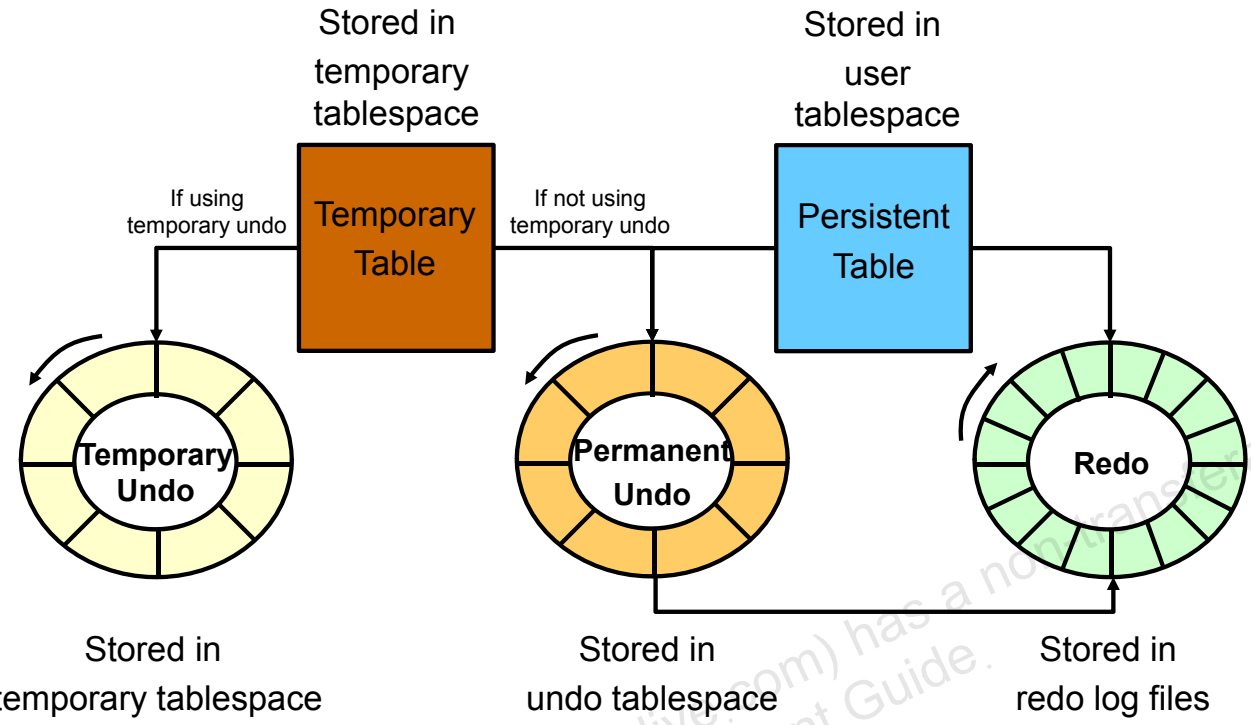
If you decide to change the undo tablespace to a fixed size, you must choose a large enough size to avoid the following two errors:

- DML failures (because there is not enough space to create the undo for new transactions)
- “Snapshot too old” errors (because there was insufficient undo data for read consistency)

Oracle recommends that you run a regular, full workload allowing the undo tablespace to grow to its minimum required size. The automatically gathered statistics include the duration of the longest-running query and the undo generation rate. Computing the minimum undo tablespace size based on these statistics is advisable for a system without Flashback operations, and for a system for which you do not expect longer-running queries in the future.

You can use the Enterprise Manager Cloud Control Undo Advisor to enter your desired duration for the undo period for longer-running queries and flashback.

# Temporary Undo: Overview






Temporary tables are widely used as scratch areas for staging intermediate results. This is because changing those tables is much faster than with non-temporary tables. The performance gain is mainly because no redo entries are directly generated for changes on temporary tables. However, the undo for operations on temporary tables (and indices) is still logged to the redo log.

Undo for temporary tables is useful for consistent reads and transaction rollbacks during the life of that temporary object. Beyond this scope the undo is superfluous. Hence it need not be persisted in the redo stream. For instance, transaction recovery just discards undo for temporary objects.

Starting with Oracle Database 12*c* it is possible for undo generated by temporary tables' transactions to be stored in a separate undo stream directly in the temporary tablespace to avoid for that undo to be logged in the redo stream. This mode is called temporary undo.

**Note:** A temporary undo segment is session private. It stores undo for the changes to temporary tables (temporary objects in general) belonging to the corresponding session.

# Temporary Undo: Benefits

- Temporary undo reduces the amount of undo stored in the undo tablespaces.
- Temporary undo reduces the size of the redo log.
- Temporary undo enables DML operations on temporary tables in a physical standby database with the Oracle Active Data Guard option.







Enabling temporary undo provides the following benefits:

- Temporary undo reduces the amount of undo stored in the undo tablespaces. Less undo in the undo tablespaces can result in more realistic undo retention period requirements for undo records.
- Temporary undo reduces the size of the redo log. Performance is improved because less data is written to the redo log, and components that parse redo log records, such as LogMiner, perform better because there is less redo data to parse.
- Temporary undo enables data manipulation language (DML) operations on temporary tables in a physical standby database with the Oracle Active Data Guard option. However, data definition language (DDL) operations that create temporary tables must be issued on the primary database.

# Enabling Temporary Undo

- Enable temporary undo for a session:

```
SQL> ALTER session SET temp_undo_enabled = true;
```

- Enable temporary undo for the database instance:

```
SQL> ALTER system SET temp_undo_enabled = true;
```

- Temporary undo mode is selected when a session first uses a temporary object.







You can enable temporary undo for a specific session or for the entire database. When you enable temporary undo for a session using an `ALTER SESSION` statement, the session creates temporary undo without affecting other sessions. When you enable temporary undo for the system using an `ALTER SYSTEM` statement, all existing sessions and new sessions create temporary undo.

When a session uses temporary objects for the first time, the current value of the `TEMP_UNDO_ENABLED` initialization parameter is set for the rest of the session. Therefore, if temporary undo is enabled for a session and the session uses temporary objects, then temporary undo cannot be disabled for the session. Similarly, if temporary undo is disabled for a session and the session uses temporary objects, then temporary undo cannot be enabled for the session.

The feature of temporary undo is available for databases with the `COMPATIBLE` initialization parameter set to at least `12.1.0.0.0`.

**Note:** Temporary undo is enabled by default for a physical standby database with the Oracle Active Data Guard option. The `TEMP_UNDO_ENABLED` initialization parameter has no effect on a physical standby database with Active Data Guard option because of the default setting.

# Monitoring Temporary Undo

```
SELECT to_char(BEGIN_TIME,'dd/mm/yy hh24:mi:ss'),
       TXNCOUNT,MAXCONCURRENCY,UNDOBLKCNT,USCOUNT,NOSPACEERRCNT
FROM   V$TEMPUNDOSTAT;

TO_CHAR(BEGIN_TIM TXNCOUNT MAXCONCURRENCY UNDOBLKCNT USCOUNT NOSPACEERRCNT
----------------- -------- -------------- ---------- ------- -------------
…
19/08/12 22:19:44         0              0          0       0             0
19/08/12 22:09:44         0              0          0       0             0
…
19/08/12 13:09:44         0              0          0       0             0
19/08/12 12:59:44         3              1         24       1             0

576 rows selected.

SQL>
```







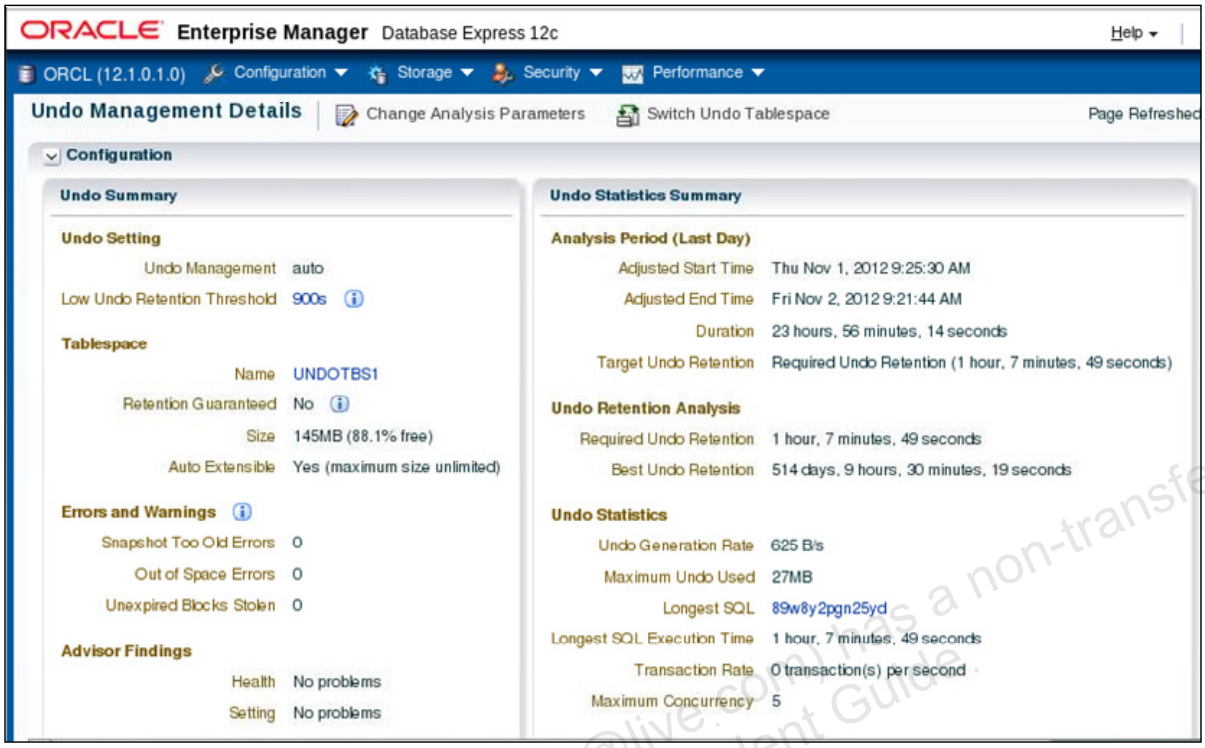
V$TEMPUNDOSTAT shows various statistics related to the temporary undo log for this database instance. It displays a histogram of statistical data to show how the system is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval. The rows are in descending order of the BEGIN_TIME column value. This view contains a total of 576 rows, spanning a 4-day cycle. This view is similar to the V$UNDOSTAT view.

The example shows you some of the important columns of the V$TEMPUNDOSTAT view:

- BEGIN_TIME: Identifies the beginning of the time interval
- TXNCOUNT: Total Number of transactions that have bound to temp undo segment within the corresponding time interval
- MAXCONCURRENCY: Highest number of transactions executed concurrently, which modified temporary objects within the corresponding time interval
- UNDOBLKCNT: Total number of temporary undo blocks consumed during the corresponding time interval
- USCOUNT: Temp undo segments created during the corresponding time interval
- NOSPACEERRCNT: Total number of times the error *no space left for temporary undo* was raised during the corresponding time interval

**Note:** For more information about V$TEMPUNDOSTAT, refer to the *Oracle Database Reference Guide*.

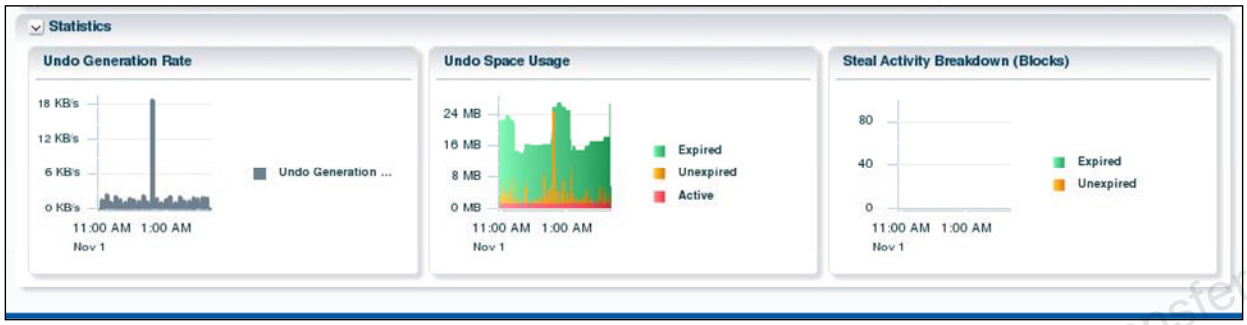
# Viewing Undo Information






You can view undo information on the Undo Management Details page in Enterprise Manager Database Express.

# Viewing Undo Activity

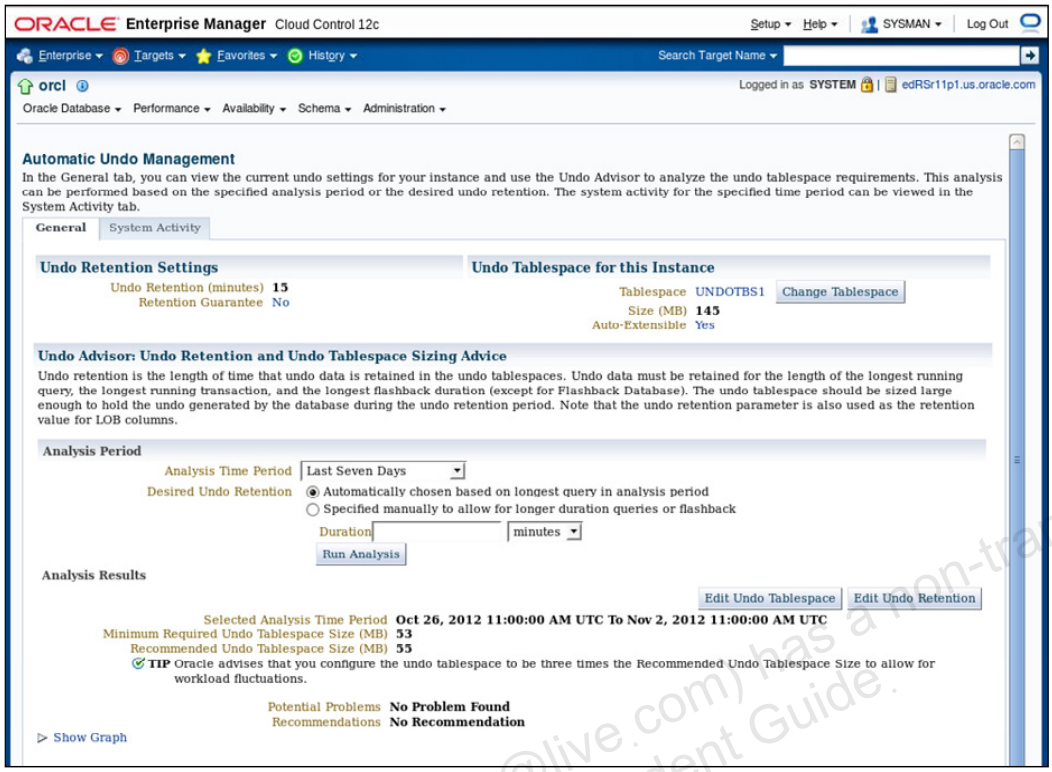





At the bottom of the Undo Management Details page, you can view additional statistical information:

- **Undo Generation Rate:** Displays the undo generation (in KB per second)
- **Undo Space Usage:** Shows the use of space in the tablespace
- **Steal Activity Breakdown:** Shows the number of attempts to steal expired undo blocks from other undo segments and attempts to obtain undo space by stealing unexpired extents from other transactions

# Using the Undo Advisor






You can access the Undo Advisor in Enterprise Manager Cloud Control by Performance > Advisors Home > Automatic Undo Management.

The middle part of the General tab contains the Undo Advisor analysis. It provides an estimate of the undo tablespace size required to satisfy a given undo retention.

Click Show Graph to see a graphical representation of the required tablespace size. You can click a point on the graph to see the tablespace size required to support the selected period.

To change the undo tablespace to a fixed size, click Edit Undo Tablespace and then click Edit in the Datafile section.

# Quiz

All you need to do to guarantee that all queries under 15 minutes will find the undo data needed for read consistency, is set the `UNDO_RETENTION` parameter to 15 minutes.

a. True
b. False



**Answer: b**

# Summary

In this lesson, you should have learned how to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo
- Use the Undo Advisor

# Practice 12

## 12-1: Managing Undo Data