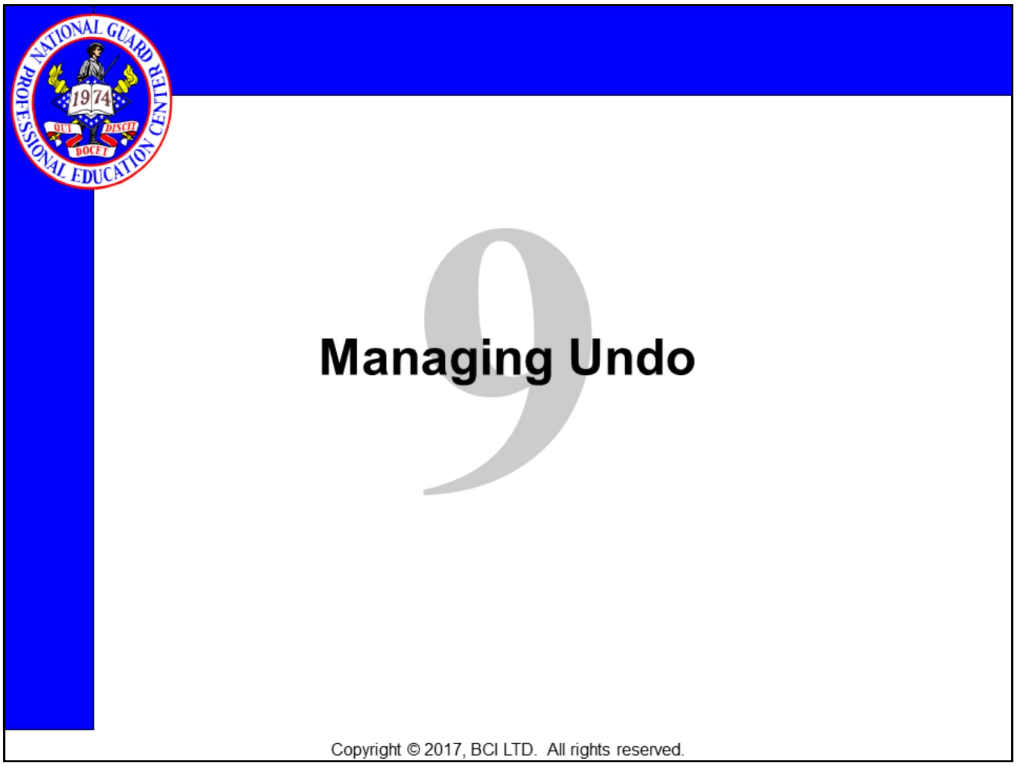
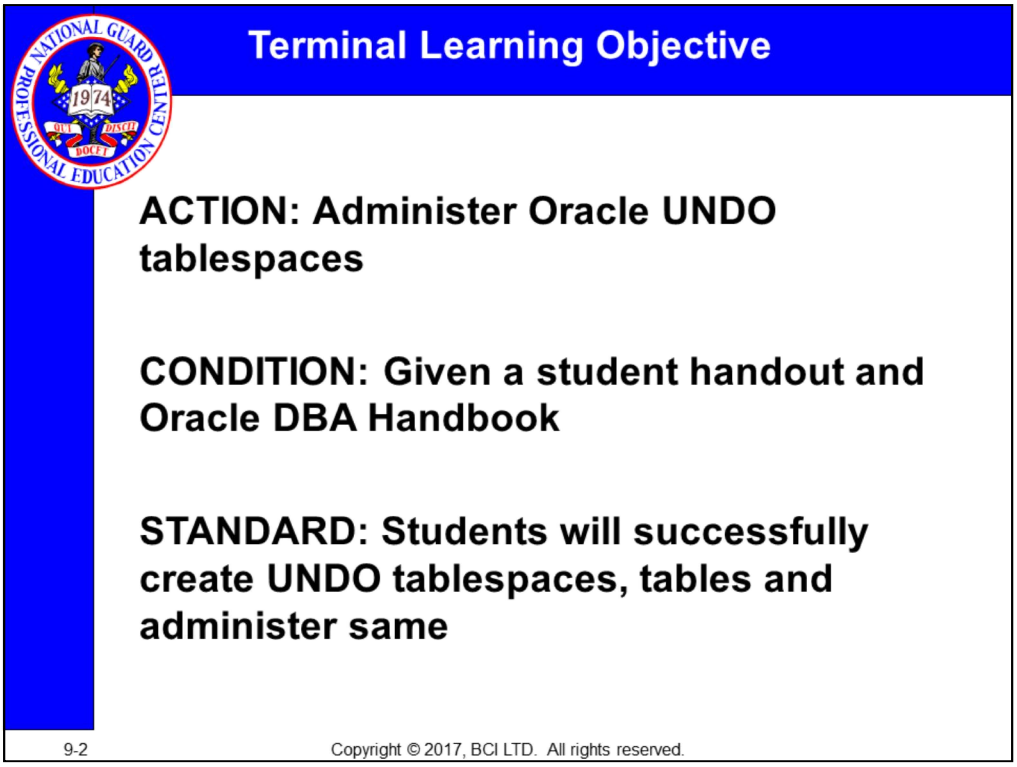
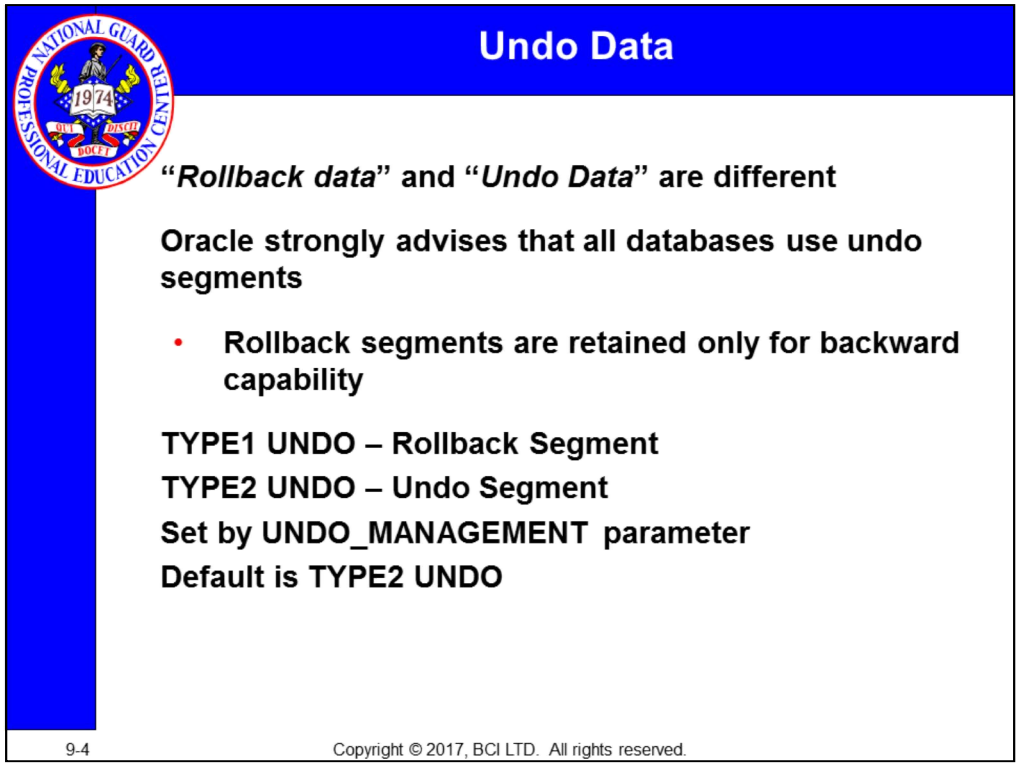

9
Managing Undo
Copyright © 2017, BCI LTD. All rights reserved.


Terminal Learning Objective
ACTION: Administer Oracle UNDO tablespaces
CONDITION: Given a student handout and Oracle DBA Handbook
STANDARD: Students will successfully create UNDO tablespaces, tables and administer same
9-2
Copyright © 2017, BCI LTD. All rights reserved.

Learning Objectives

After completing this lesson, you should be able to do the following:

- Undo Data
- Undo Tablespaces
- Transactions and Undo Segments
- Managing Undo
- Error Conditions Related to Undo
- Parameters for Undo Management
- Sizing and Monitoring the Undo Tablespace
- Dropping and Shrinking Undo Segments



Undo data is information needed to reverse the effects of DML statements. It is often referred to as "rollback data," but try to avoid that term. In earlier releases of Oracle, the terms "rollback data" and "undo data" were used interchangeably, but from 9i onward, they are different.

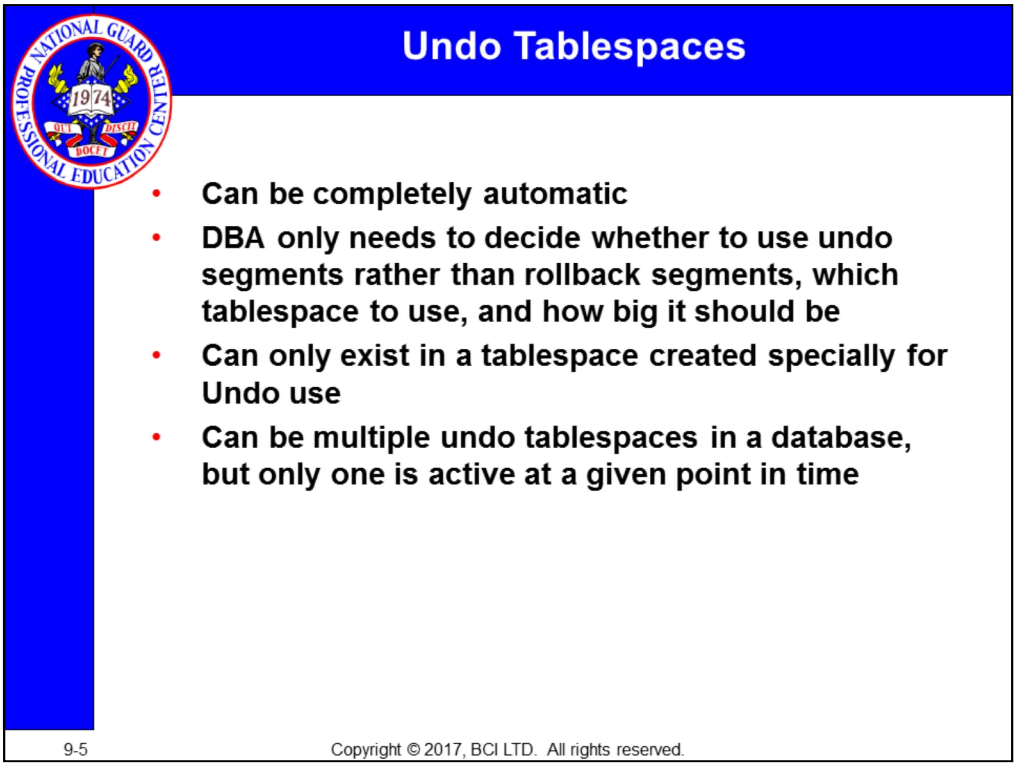
Whenever a transaction changes data, the pre-update version of the data is written out to a rollback segment or to an undo segment. The difference is crucial. Rollback segments can still exist, but with release 9i of the database, Oracle introduced the undo segment as an alternative.

To roll back a transaction means to use data from the undo segments to construct an image of the data as it was before the transaction occurred. This is usually done automatically, but the flashback capability leverages the power of the undo mechanism by giving you the option of querying the database as it was at some time in the past.

Oracle must keep pre-update versions of data in order that incomplete transactions can be reversed, either automatically in the case of an error or on demand through the use of the rollback command. This type of rollback is permanent and published to all users. Second, for read consistency, the database must be able to present a query with a version of the database as it was at the time the query started. The server process running the query will go to the undo segments and construct what is called a "read consistent" image of the blocks being queried, if they were changed after the query started. This type of rollback is temporary and visible only to the session running the query. Third, undo segments are also used for transaction isolation. This is perhaps the most complex use of undo data.

From 9i onward, undo data can also be used for flashback queries. This is a completely optional tool.

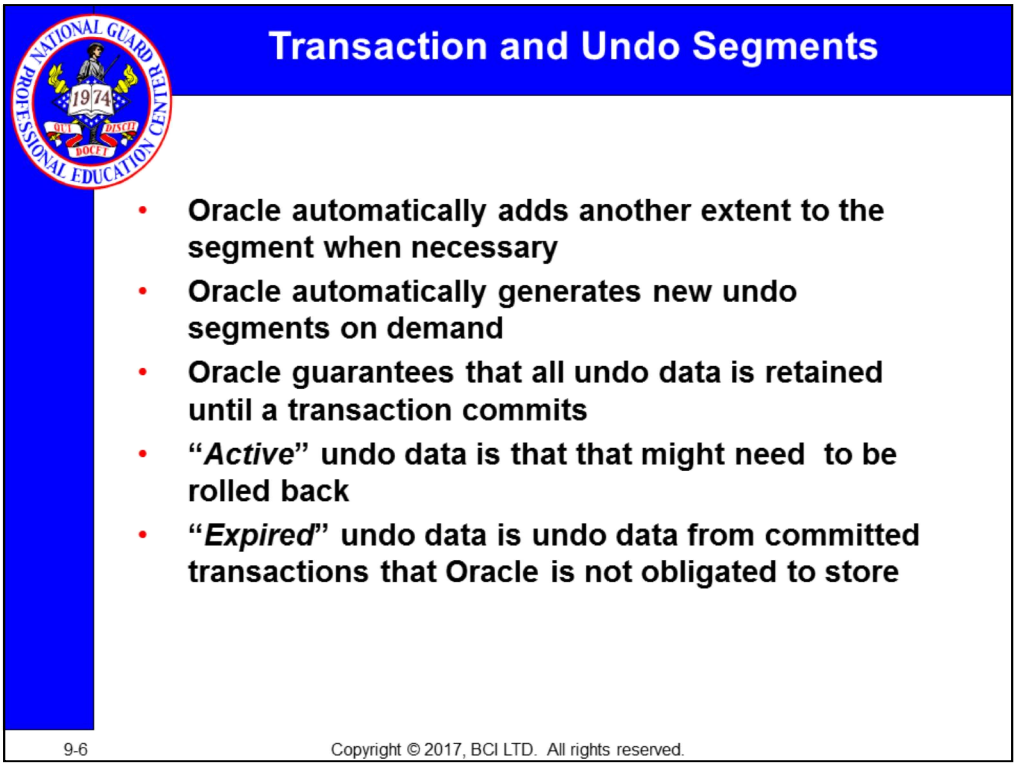
As a final word on "rollback" as opposed to "undo," when you query the data dictionary and see TYPE2 UNDO, this is an undo segment. So, what type of segment is TYPE1 UNDO? A rollback segment.



Starting with Oracle 9i release, the management of undo data, by means of the new segment type “undo,” can be completely automatic. The only control the DBA has is whether to use the undo segments rather than rollback segments and, if so, which tablespace to keep them in, and how big this tablespace should be. The actually size will be discussed later in this module.

One feature of undo segments is that they can exist only in a tablespace created specially for that purpose, whereas rollback segments could be created in any tablespace. You will always create an undo tablespace, possibly at database creation time. You may well create more than one, but if you do, only one of them will be used at any given moment. You can create undo tablespace from the SQL*Plus command line, from iSQL*Plus, or from Database Control.

An undo tablespace is in many ways a tablespace like any other, but there are some limitations. It must be created as permanent, locally managed, with automatic extent allocation. After creation, management commands are generally limited to physical operations. You can add, move, or resize datafiles in an undo tablespace, but you cannot, for example, make an undo tablespace read-only. If it is the active undo tablespace, you cannot take it offline or drop it.

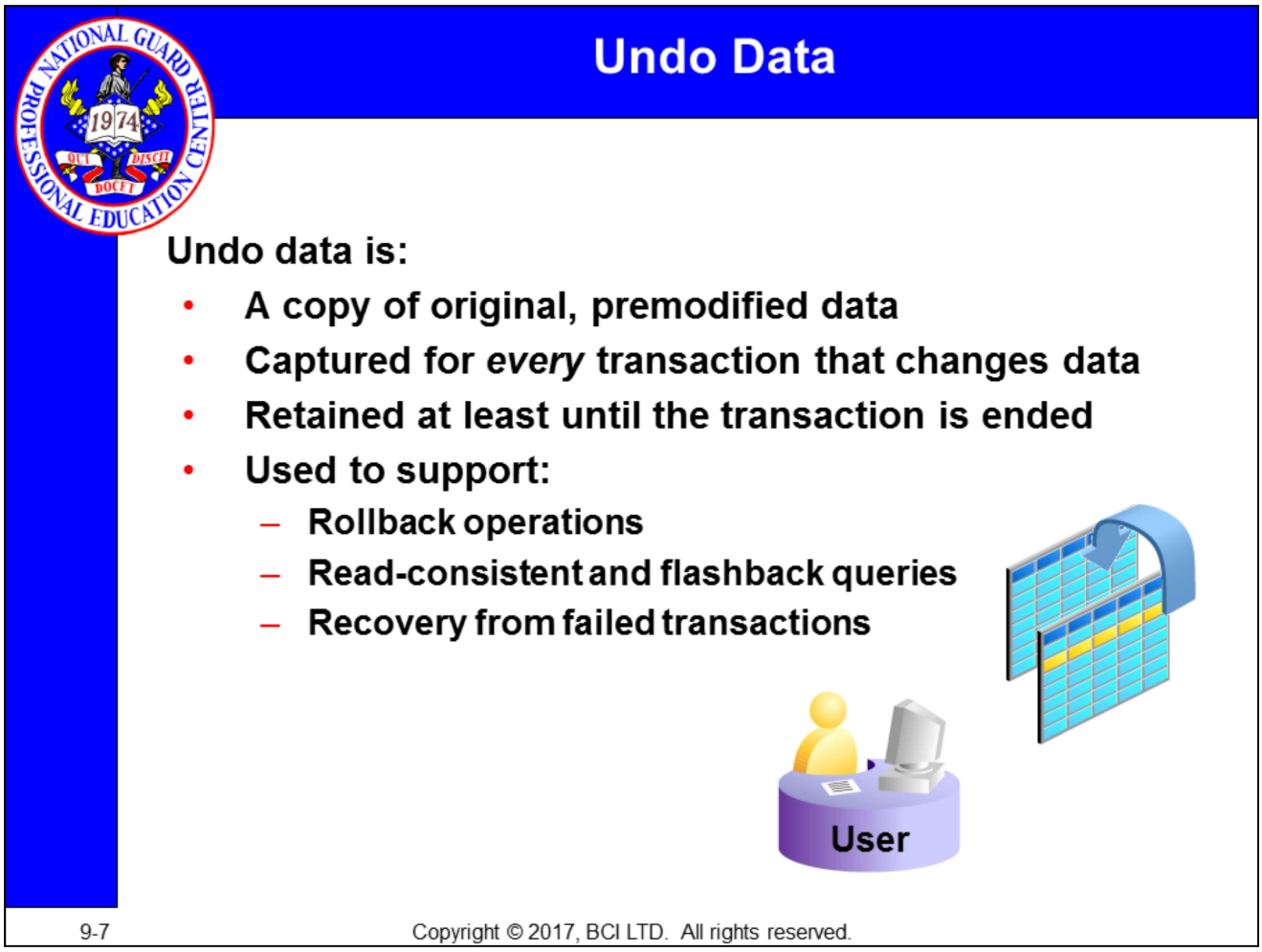


When a transaction starts, Oracle will assign it to one (and only one) undo segment. Any one transaction can be protected by only one undo segment; it is not possible for the undo data generated by one transaction to cut across multiple undo segments. This is not a problem, because undo segments are not fixed in size. So if a transaction does manage to fill its undo segment, Oracle will automatically add another extent to the segment so that the transaction can continue. It is possible for multiple transactions to share one undo segment, but in normal running this should not occur. A tuning problem common with rollback segments was estimating how many rollback segments would be needed to avoid excessive interleaving of transactions within rollback segments without creating so many as to waste space. One feature of undo management is that Oracle will automatically generate new undo segments on demand, in an attempt to ensure that it is never necessary for transactions to share undo segments. If Oracle has found it necessary to extend its undo segments or to generate additional segments, when the workload drops Oracle will shrink and drop the segments, again automatically.

As a transaction updates table or index data blocks, the information needed to roll back the changes is written out to blocks of the assigned undo segment. Oracle guarantees that all the undo data must be retained until a transaction commits. Oracle guarantees consistency to the extent that if a query succeeds, the results will be consistent with the state of the database at the time the query started – but it does not guarantee that the query will actually succeed. This means that undo data can be divided into two parts. "Active" undo is undo data that might be needed to roll back transactions in progress. This data can never be overwritten, until the transaction completes. "Expired" undo is undo data from committed transactions that Oracle is no longer obliged to store – though it might be needed for consistent reads, if there are any long-running queries in progress. This data can be overwritten if Oracle needs the space for another active transaction.

The fact that undo information expires on commit means that undo segments can be used in a circular fashion. Eventually, the whole of the undo tablespace will be filled with undo data, so when a new transaction starts, or a running transaction generates some more undo, the undo segment will "wrap" around, and the oldest undo data within it will be overwritten – always assuming that this oldest data is not part of a long-running uncommitted transaction, in which case it would be necessary to extend the undo segment instead.

With the old, manually managed rollback segments, a critical part of tuning was to control which transactions were protected by which rollback segments. A rollback segment might even be created and brought online specifically for one transaction. Automatically managed undo segments make all of that unnecessary, because the DBA has no control over which undo segment will protect any one transaction. Oracle does a better job than the DBA could!



Undo Data

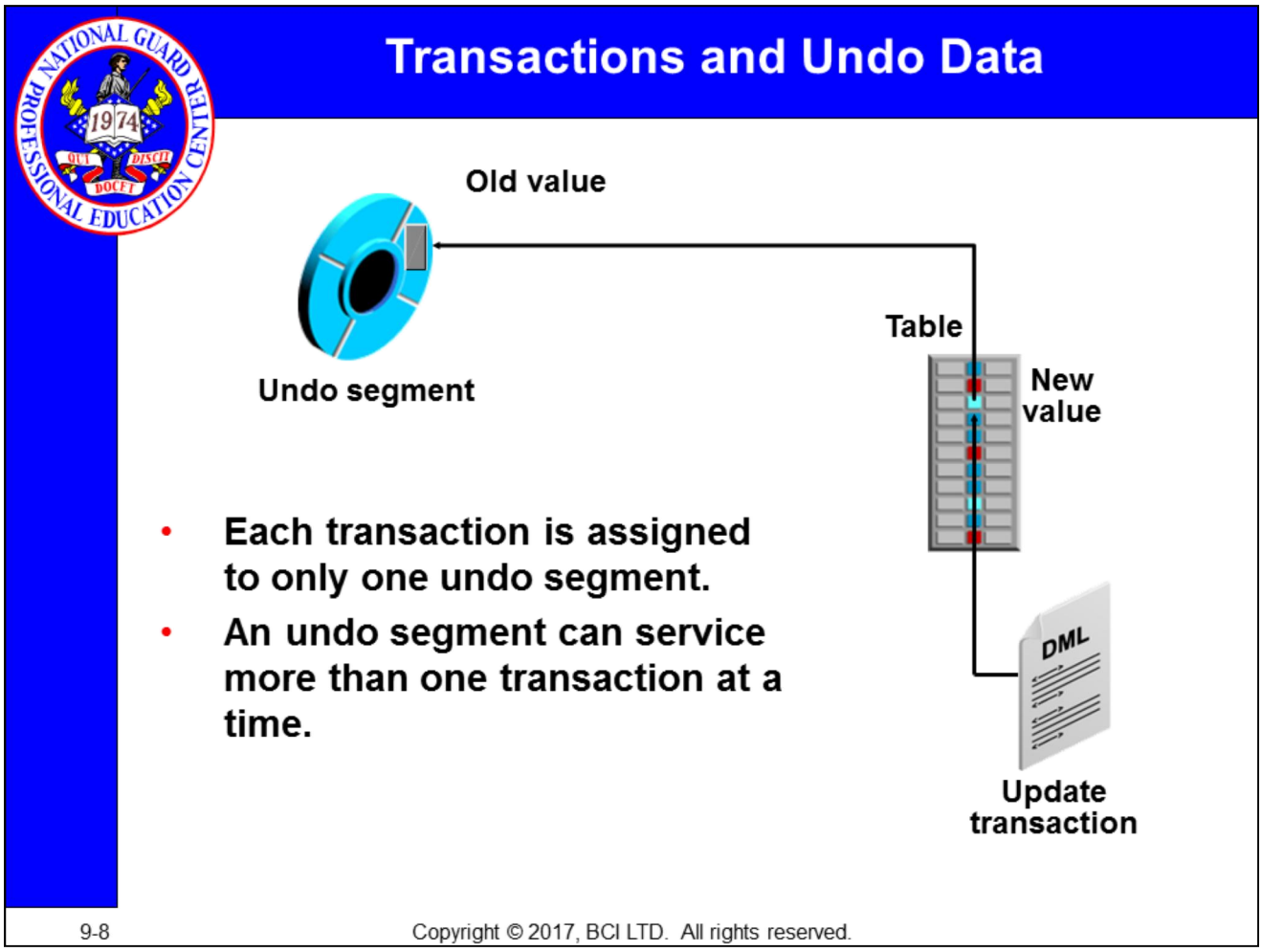
The Oracle database saves the old value (undo data) when a process changes data in a database. It stores the data as it existed before being modified. Capturing undo data enables you to roll back your uncommitted data. Undo also supports read-consistent and flashback queries.

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. As long as the undo information is retained, the Oracle database can reconstruct data to satisfy read-consistent queries.

Flashback queries are queries that 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.

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.



Transactions and Undo Data

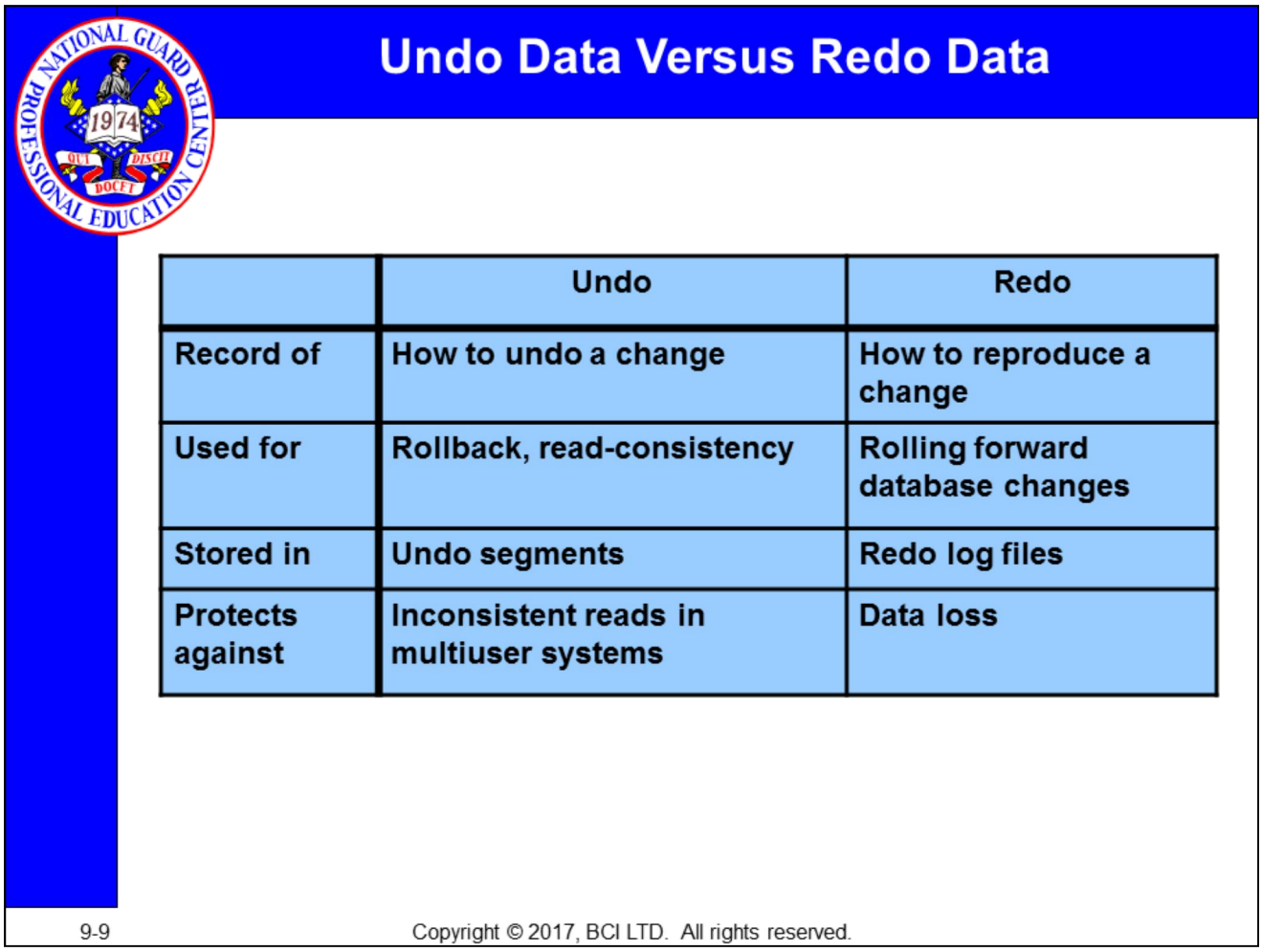
When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `v$transaction` dynamic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

Undo segments are specialized segments that are automatically created by the instance 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 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 10g*.

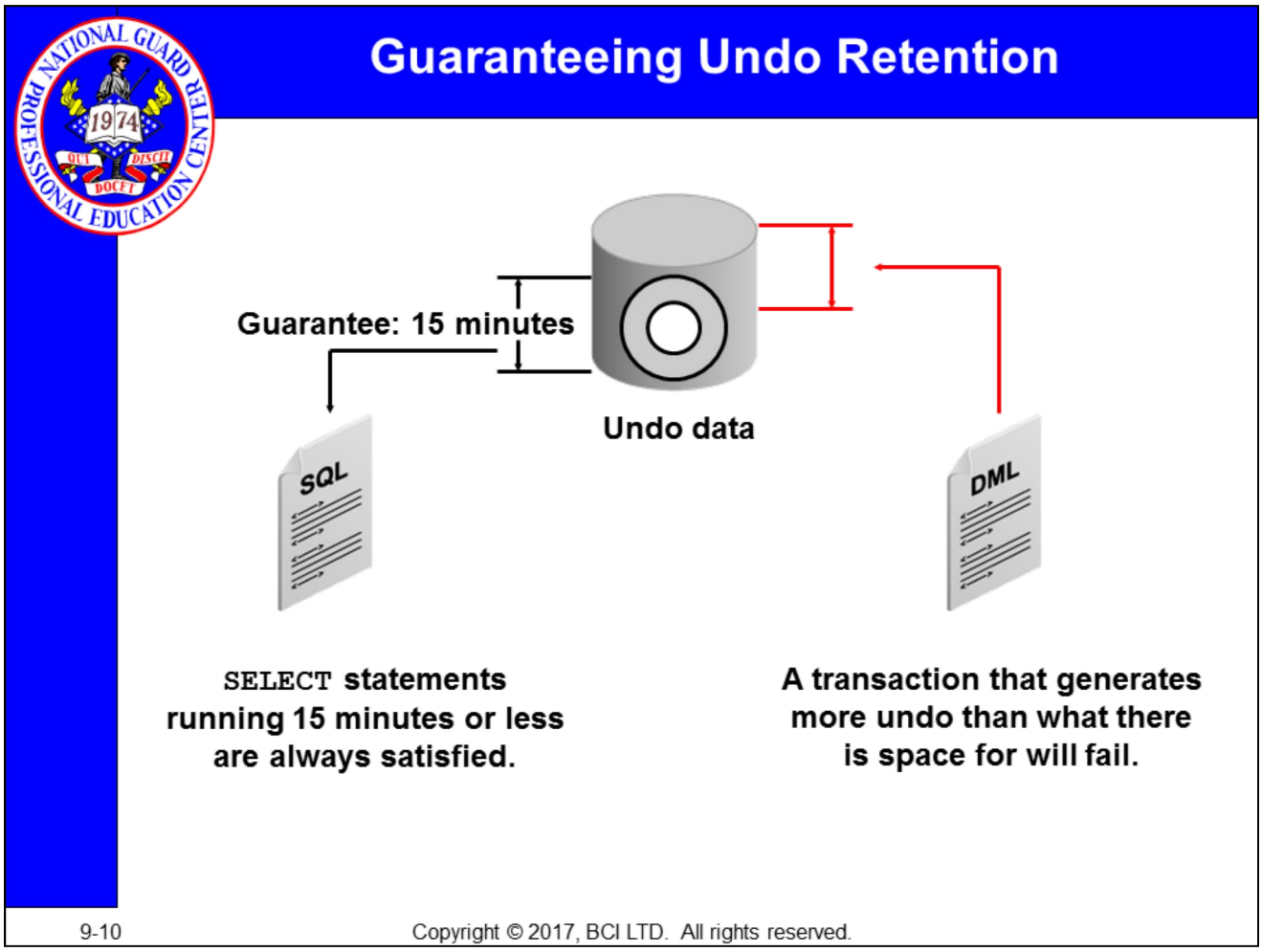


Undo Data Versus Redo Data

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

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, it is typically multiplexed. So, there are multiple copies of the redo data on the disk. Even though the changes may not have yet been written to the data files where the table's blocks are actually stored, guaranteeing that the changes have been written to the redo log file is enough.

A power outage that occurs just before committed changes have been reflected into the data files does not cause a problem because the transaction has been committed. So, 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.



Guaranteeing Undo Retention

The default undo behavior is to overwrite committed transactions that have 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:

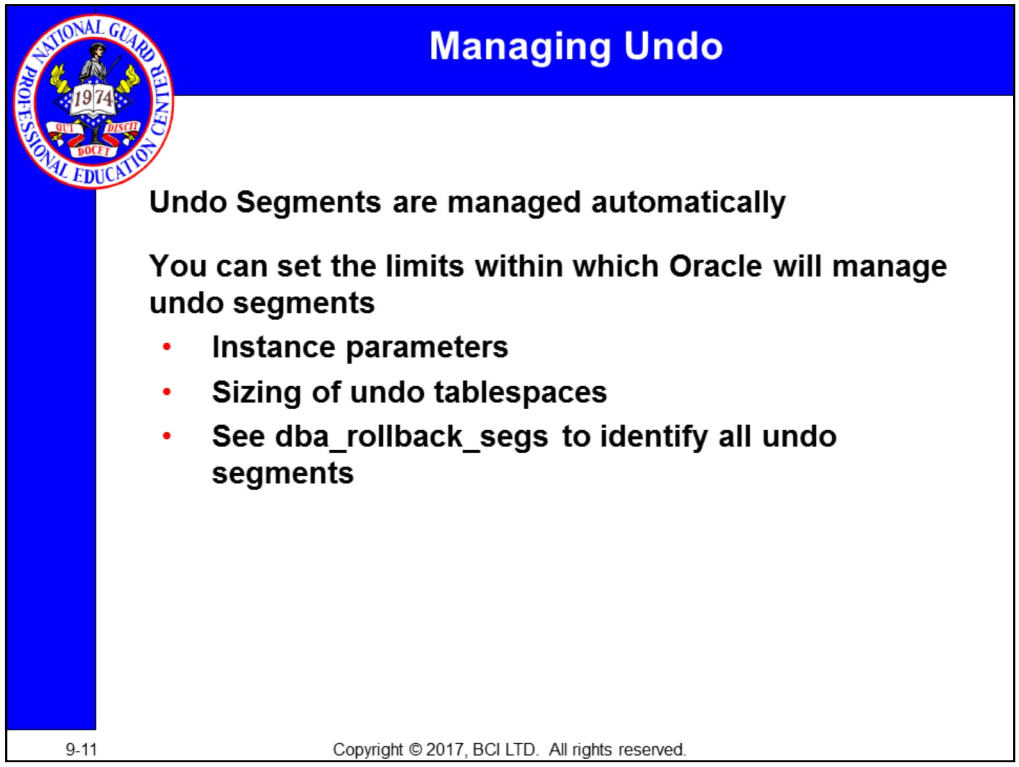
```
SQL> ALTER TABLESPACE undotbs1 RETENTION
NOGUARANTEE;
```

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

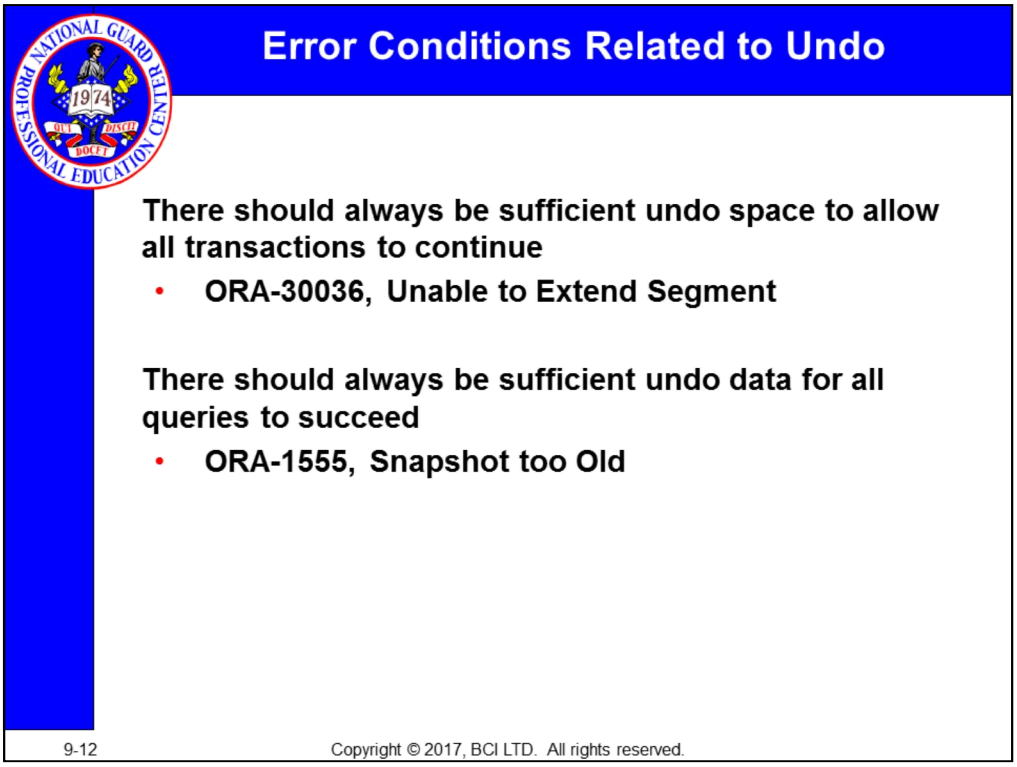
```
SQL> ALTER TABLESPACE example RETENTION
GUARANTEE;

ERROR at line 1:

ORA-30044: 'Retention' can only specified for undo
tablespace
```

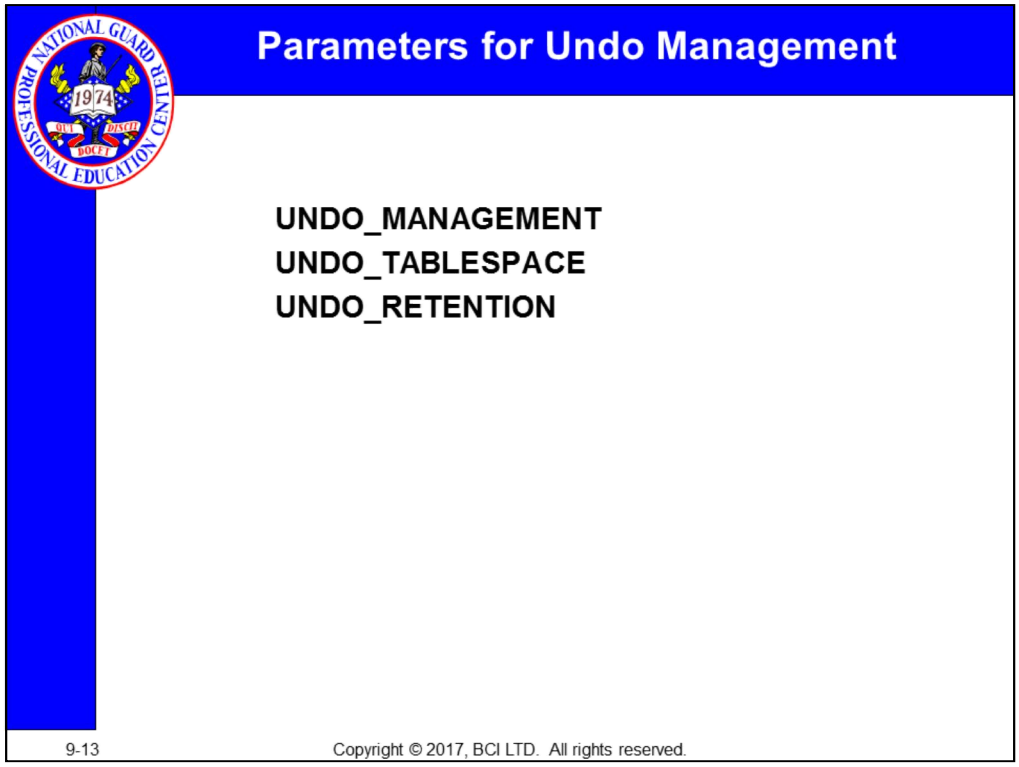


A major feature of undo segments it that they are managed automatically, but you must set the limits within which Oracle will do its management. After considering the nature and volume of activity in your database, you set certain instance parameters and adjust the size of your undo tablespace in order to achieve your objectives.



The principles are simple: first, there should always be sufficient undo space to allow all transactions to continue, and second, there should always be sufficient undo data for all queries to succeed. The first principle requires that your undo tablespace be large enough to accommodate the worst case for undo demand. It should have enough space allocated for the worst case – the peak usage of active, or unexpired, undo data generated by your transaction workload. Note that this might not be during the peak number of concurrent transactions; it could be that during normal running you have many small transactions, but the total undo they generate might be less than that generated by a single end-of-month batch job. The second principle requires that there be additional space in the undo tablespace to store expired undo data that might be needed for read consistency, so that long-running queries will not fail with a famous Oracle error ORA-1555, snapshot too old.

If a transaction runs out of undo space, it will fail with the error ORA-30036, unable to extend segment in undo tablespace. The statement that his the problem is rolled back, but the rest of the transaction remains intact and uncommitted. The algorithm that assigns space within the undo tablespace to undo segments means that this error condition will arise only if the undo tablespace is absolutely full of unexpired undo data. If a query fails on consistent read with “snapshot too old”, it means that the query hit a block that has been changed since the query started, but when it went to the undo segment to find the pre-update version of the data, that bit of undo data had been overwritten.



There are three initialization parameters controlling undo: UNDO_MANAGEMENT, UNDO_TABLESPACE, and UNDO_RETENTION.

UNDO_MANAGEMENT defaults to “AUTO,” meaning that Oracle will use undo segments. Most DBAs set this parameter at database creation and, therefore, it is set to “auto.” but can be set to ‘manual’ should the DBA suffer from a bout of insanity. This parameter is static, meaning that if it is changed, the change will not come into effect until the instance is restarted. The other parameters are dynamic – they can be changed while the instance is running.

Finally, UNDO_RETENTION, set in seconds, is usually optional. It specifies a target for keeping expired undo data. If, for example, your longest running query is thirty minutes, you would set this parameter to 1800. Oracle will then attempt to keep all undo data for at l your query should therefore never fail with ORA-1555. The algorithm controlling which expires undo data is overwritten first will always choose to overwrite the oldest bit of data; With Oracle 10g, there is an option to guarantee undo retention. This means that undo data will never be overwritten until the time specified by the UNDO_RETENTION parameters has passed. Retention guarantee is enabled at the tablespace level via the Retention Guarantee clause.

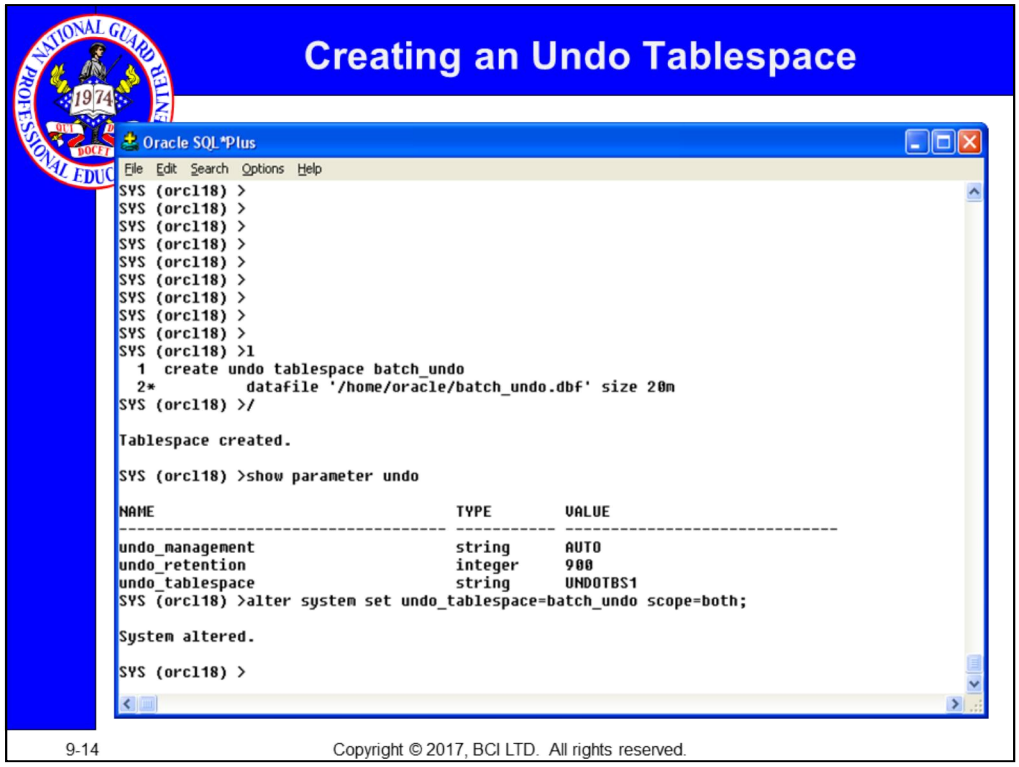
A database might have one tablespace used in normal operations where undo retention is not guaranteed, and another used during month-end reporting where retention is guaranteed.

You can alter the tablespace while online with the following syntax:

```
alter tablespace undo_guarantee retention guarantee;
```

-- then for normal running

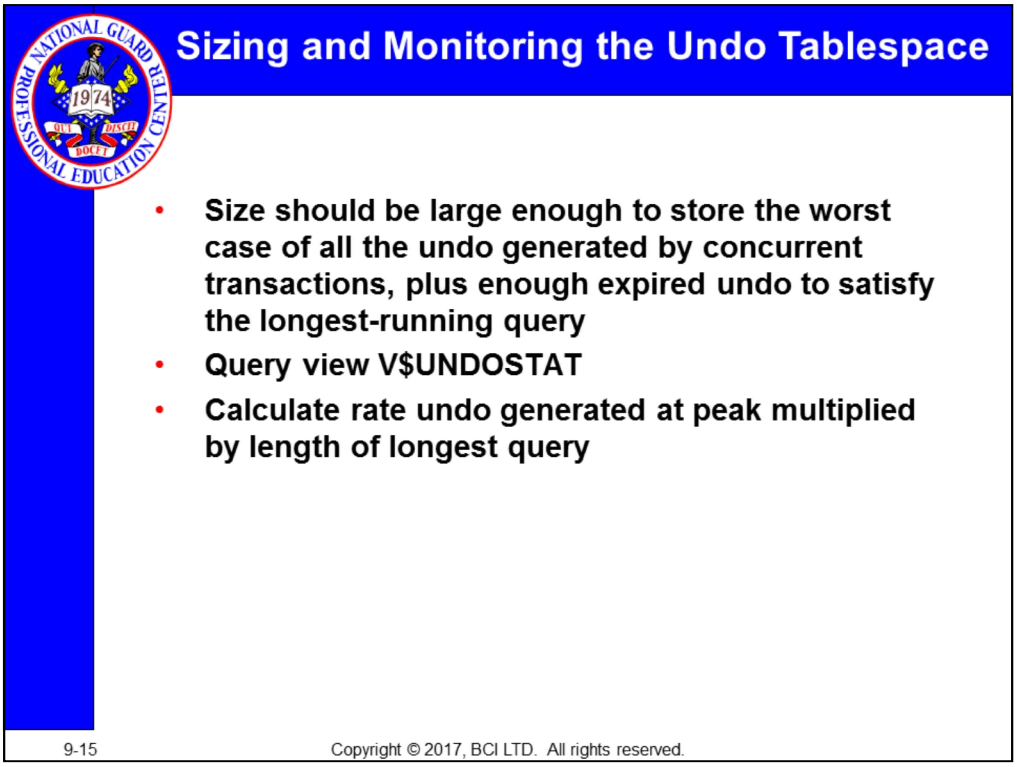
```
alter system set undo_tablespace = undo_noguarantee;
```



You may create a second undo tablespace to handle much larger batch applications which change massive amounts of data. First, you would create

The new undo tablespace with the appropriate size. You can go to the Undo Size Advisor in Enterprise Manager to see how big this needs to be based upon current applications.

Secondly, you would issue the command to swap undo tablespaces. Keep in mind you can only have one (1) active undo tablespace at one time.

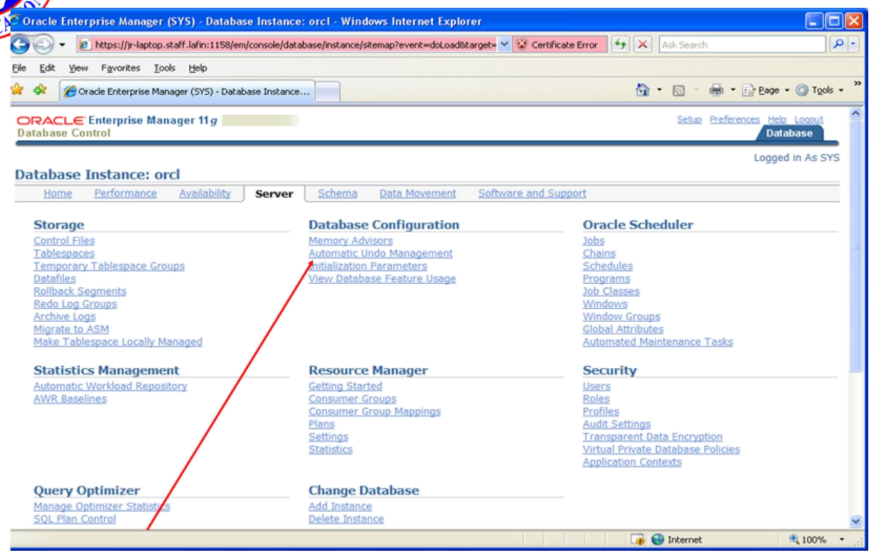


The undo tablespace should be large enough to store the worst case of all the undo generated by concurrent transactions, plus enough expired undo to satisfy the longest-running query. In an advanced environment, you may also have to add space to allow for flashback queries as well. The algorithm is simple: calculate the rate at which undo is being generated at your peak workload, and multiply by the length of your longest query.

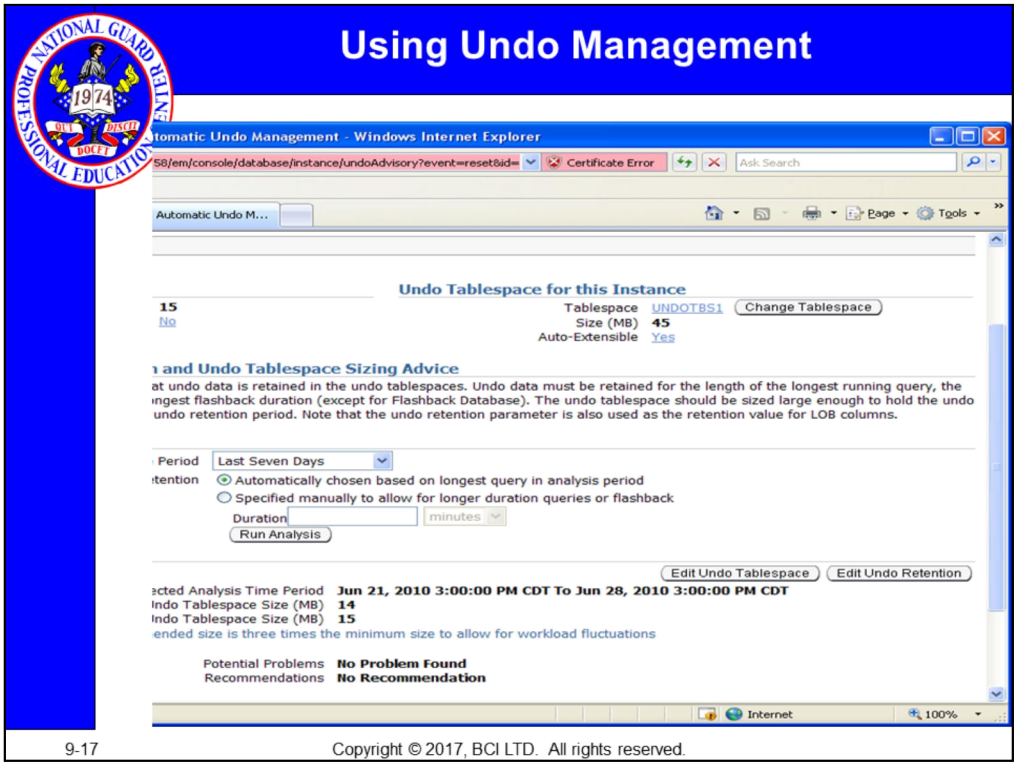
There is a view, V$UNDOSTAT, that will tell you all you need to know. There is also an advisor within Database Control that will present the information in an immediately comprehensible way. To reach the undo management screen of Database Control, take the Administration tab from the database home page, and then the Undo Management link in the Instance section. The Change Tablespace button will issue an ALTER SYSTEM command to activate an alternative undo Tablespace.

The System Activity and Tablespace Usage section of the screen tells you what the peak rate for undo generation is. It will also tell you about the longest running query. To view the undo advisor, click Advisor.

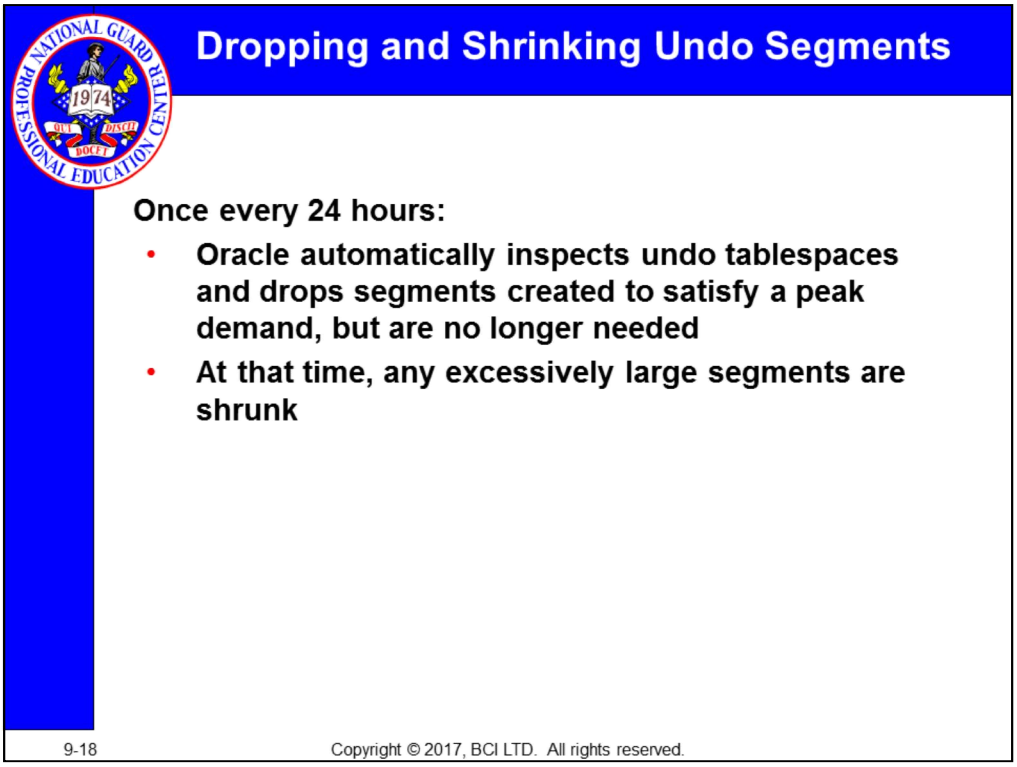
E/M Undo Management



In Enterprise Manager, go to the home page and click on Undo Management.



One of the major features of the E/M Undo Management screen is the ability to see the current amount of work and throughput along with using the Undo Advisor to see if the Undo Tablespace needs to be adjusted.



When an undo tablespace is created, Oracle will create a pool of undo segments within it. If the number of concurrent transactions exceeds the number of segments in this pool, Oracle will create more. Also, if a transaction's undo data exceeds the size of its undo segment, the segment will be extended. In normal running, if the undo tablespace is under space pressure, Oracle will automatically transfer extents of expired undo data from one segment to another if necessary, to ensure that undo segments do have enough space for the active undo being generated by currently running transactions. Additional mechanism for resizing, and indeed dropping, undo segments is driven by the system monitor process, SMON. Once every twenty-four hours, SMON will inspect the undo tablespaces and drop segments that were created to satisfy a peak demand but are no longer needed. At the same time, any excessively large segments will be shrunk.

Practical Exercise

Lab 10: Managing Undo

9-19

Summary

In this lesson, you should have learned how to:

- Undo Data
- Undo Tablespaces
- Transactions and Undo Segments
- Managing Undo
- Error Conditions Related to Undo
- Parameters for Undo Management
- Sizing and Monitoring the Undo Tablespace
- Dropping and Shrinking Undo Segments

Terminal Learning Objective

ACTION: Administer an Oracle UNDO tablespaces

CONDITION: Given a student handout and Oracle DBA Handbook

STANDARD: Students will successfully create UNDO tablespaces, tables and administer same