9 Flashback Technology Enhancements

CERTIFICATION OBJECTIVES

This chapter reviews Oracle's flashback technology, which operates at many levels, serving a varied list of objectives. The flashback features of Oracle let you undo logical corruptions or user errors by retrieving data from a past time period. The rationale behind flashback technology is to enable quick recovery from logical errors without having to resort to time-consuming point-in-time recovery strategies using backups and archived redo logs. Several of the flashback recovery features rely on the familiar undo data, since they all need information from the past to recover from logical errors. In Oracle Database 10*g*, a new type of recovery log, called a flashback log, enables the flashback database feature. You can use the flashback features of Oracle Database 10*g* to perform queries that return past data, to perform queries that show the history of changes to table data, to undo undesirable changes to table data, and to recover table(s) or even an entire database to a previous point in time.

Flashback technologies are superior to traditional recovery methods like point-in-time recovery, which have become prohibitively slow in larger databases. Flashback techniques hold an advantage over the use of the LogMiner tool as well. Flashback techniques are faster because they focus purely on specific items of the changed data—there is no need to parse entire redo log files to glean information about a single bad transaction. Oracle indexes all the changes, both on a row as well as transaction basis, for fast access. The flashback commands are simple as well. Thus, flashback techniques provide you with a fast and efficient way to recover from logical (or user made) data errors. In Oracle Database 10*g*, you can use the flashback technology at the database, table, and transaction levels, as summarized here:

- Flashback database enables you to take the entire database to a past point in time (using flashback logs). Example: useful when you drop a user accidentally, truncate a large table, or a batch job applies partial changes only.
- Flashback drop lets you retrieve accidentally dropped tables and indexes (using the recycle bin). Example: useful when you accidentally drop a table. This feature also lets you restore a table to an earlier state in the event of an error.
- Flashback table lets you recover a table to a time in the past (using undo data). Example: useful in situations when you update a table with the wrong WHERE clause.
- Flashback query lets you query and restore data rows to a point in time (using undo data). Examples: you want to compare current data against past data, or you want to undo the effect of incorrectly run DML statements.

exam watch

Most flashback features (flashback transaction query, flashback versions query, flashback table) rely on undo data. The flashback database feature relies on the new flashback logs. The flashback drop (table) feature relies on the new concept of a recycle bin.

on the job

Although all the recovery features in this chapter are called flashback features, not all of them rely on the flash recovery area or the new flashback logs. Only the flashback database feature uses the flash recovery area and flashback logs. The flashback drop feature relies on the new recycle bin concept. All the flashback query features rely on undo data.

We'll look at the various Oracle Database 10*g* flashback features in detail in this chapter. Let's start with a review of the building blocks that enable Oracle to offer you all the amazing and painless recovery techniques.

CERTIFICATION OBJECTIVE 9.01

General Flashback Technology Considerations

Before we delve into the various flashback features, let's review Oracle's undo management, which is at the heart of several of the new features. Let's see how the concept of guaranteed undo retention turns out to be critical for the functioning of the flashback technology.

on the job

If you have a damaged disk drive, or if there is physical corruption (not logical corruption due to application or user errors) in your database, you must still use the traditional methods of restoring backups and using archived redo logs to perform the recovery.

Guaranteed Undo Retention

Oracle automatically collects undo data and stores it in the undo segments. Undo data records the effects of individual transactions in changing table row data. Traditionally, Oracle has used undo data to provide read consistency for queries, to roll back unneeded transactions, and to recover terminated transactions. Starting with the Oracle9*i* version, Oracle has been using undo data for even farther-reaching purposes—to query past data and recover from logical errors in the data. In Oracle Database 10*g*, use of undo data has been extended to flashback entire tables to a past time, as well as perform transaction auditing.

The initialization parameter UNDO_RETENTION enables you to specify the length of time Oracle must retain undo information in the undo segments. Oracle Database 10*g* automatically tunes undo information by collecting statistics on the longest-running queries and the undo generation rate in your database. If you don't set the UNDO_RETENTION parameter, or you specify a zero value for the parameter, Oracle automatically tunes undo, using 900 seconds as the default value for the parameter. By setting a much higher value than the default of 900 seconds, you can go back further in the past. Since many flashback features in Oracle Database 10*g* depend on having enough undo data, you should set the UNDO_RETENTION parameter much higher than the default value. (In addition to enabling more effective flashback features, this will also reduce the occurrence of the old snapshot too old errors.)

In previous chapters, you learned about the new guaranteed undo retention concept in Oracle Database 10*g*. Guaranteed undo retention simply means that Oracle will keep undo information in the undo segments for the entire length of the undo retention period you specify (using the UNDO_RETENTION parameter), no matter what. That is, if you specify half an hour as the undo retention interval, Oracle will retain all undo segments for half an hour in the undo tablespace, even if there is space pressure in the undo segments. If there were no room for saving undo information for the new transactions, any new DML transactions would fail, since Oracle won't able to store the undo information for those changes. Thus, if you don't size your undo tablespace correctly, there is a trade-off between guaranteeing undo information and the potential failure of some DML statements.

exam watch

By default, Oracle doesn't guarantee undo retention. When you do choose to guarantee undo retention, the default interval is 900 seconds (15 minutes).

You can specify undo guarantee in several ways: You can specify undo guarantee for the undo tablespace when you create the database. You can also specify guaranteed undo retention by using the RETENTION GUARANTEE clause when you create a new undo tablespace, as shown here:

```
SQL> create undo tablespace test1
     datafile 'c:\oracle\product\10.1.0\oradata\undotbs_01.dbf'
     size 100M autoextend on
     retention guarantee;
Tablespace created.
SQL>
```

You can also use the ALTER TABLESPACE command to tell Oracle to guarantee undo retention in your database, as shown here:

```
 SQL> alter tablespace test1
      retention guarantee;

Tablespace altered.
SQL>
```

You can use the RETENTION NOGUARANTEE clause to turn off the guaranteed retention of undo information.

on the job

Use Oracle's Undo Advisor to get approximate undo parameter values as well as suggestions regarding the sizing of your undo tablespace to successfully support flashback for a specified time.

Time Mapping Granularity

Oracle gives you a choice between using either clock time or the system change number (SCN) to specify exactly what time in the past you want to go back to. Oracle uses an internal mapping between clock times and SCNs. Thus if you specify a certain clock time, Oracle will pick an SCN time that's within three seconds of the clock time you specify.

Oracle retains the mapping between your clock time and SCNs for a period that is as long as your UNDO_RETENTION initialization parameter.

Two new SQL functions convert SCNs to a corresponding timestamp value and vice versa. Let's review these functions in the following sections.

SCN_TO_TIMESTAMP

The SCN_TO_TIMESTAMP SQL function lets you convert an SCN to a calendar time (TIMESTAMP) value. Here's an example:

```
SQL> SELECT current_scn, SCN_TO_TIMESTAMP(current_scn)
  2  FROM v$database;
```

```
         CURRENT_SCN            SCN_TO_TIMESTAMP(CURRENT_SCN)
        -------------         -----------------------------------
           5956956              03-JUL-04 09.37.16.000000000 AM
SQL>
```

TIMESTAMP_TO_SCN

The `TIMESTAMP_TO_SCN` function is the inverse of the `SCN_TO_TIMESTAMP` function. It converts a timestamp to its corresponding SCN.

CERTIFICATION OBJECTIVE 9.02

Flashback Database

When you discover a major logical corruption in your database, the usual recourse is to perform a point-in-time recovery. This type of recovery involving the use of datafile backup copies and archived redo logs is very cumbersome and time consuming. Flashback database does the same job as a traditional point-in-time recovery: it takes the database back to a specific point in time or system change number (SCN). The big difference, of course, is that you don't need to restore any backup datafiles, and you may need, at the most, just a fraction of the archived redo log information. Consequently, a flashback database operation lets you recover from logical corruptions much faster than with the usual point-in-time incomplete database recoveries.

You must understand that flashing back a database is possible only when there is no media failure. If you lose a datafile, you'll have to recover using a restored datafile from backups. You must have all the datafiles and they must all be uncorrupted in order to conduct a flashback database operation.

The critical thing to note is that no matter what the extent of the logical corruption, traditional point-in-time recoveries require the restoration of datafiles and application of archived redo logs. With flashback database, the extent of time taken for recovery directly depends on the amount of changes that you need to undo. Thus, the size of the error, not the size of the database, determines the time it takes to recover.

How Flashback Database Works

Once you enable the flashback database feature (I show how to do this in upcoming sections), at regular intervals, the database copies images of each altered block in the datafiles from memory (flashback buffer) to the new flashback logs. Oracle logs flashback data at infrequent intervals to reduce the I/O and CPU overhead. Oracle stores these flashback logs in the flashback recovery area. That's why the first step in turning the flashback database feature on is to make sure you have configured a flash recovery area.

The new memory buffer, flashback buffer, logs images of all altered data blocks in the database. You can use these *before images* of data blocks to reconstruct a datafile so you can back out any changes made after a specified target time and turn the file back to a time in the past. In reality, however, the flashback database logs are used to recover to a time just before the target time. Oracle uses traditional archive logs to write changes for the short gap in time between the target recovery point in time and the actual recovery time. A new background process, RVWR (Recovery Writer), starts up automatically when you enable the flashback database feature. The RVWR process writes the contents of the flashback buffer to the flashback database logs. The flashback database logs are somewhat similar to the traditional Oracle redo logs. Indeed, both logs are written to from a buffer area. However, the big difference is that there are no *archived* flashback database logs! Oracle stores all the flashback database logs in the flash recovery area.

When you want to restore a database to a past target time using the flashback database feature, Oracle will restore each data block to its state at the time immediately before the target time that the flashback logs were written. For example, if you want to flashback to 8:00 A.M., it may turn out that the flashback logs nearest to the target time were written at 7:56 A.M. To cover this gap, you must apply the changes from archived or online redo log files pertaining to that period. This is the reason why, although you aren't doing a traditional point-in-time recovery using backup files and archived redo log files, you must have redo logs available for the entire time period spanned by the flashback logs.

Traditional point-in-time incomplete recovery involves the restoration of backup data files first. After the restore, you apply the archive redo logs to advance the database forward. In flashback database, you don't use any backup data files. You simply apply the flashback database logs to take the database back in time.

on the job

Oracle recommends that you set your redo log buffer for large databases to at least 8MB. The flashback buffer size will be at least log_buffer times two, thus ensuring that Oracle allocates the typical maximum memory possible for your flashback buffer, which is about 16MB.

exam watch

The time taken to flashback a database strictly depends on how far back you want to flashback and the amount of data block changes in the meantime. If you have a heavy DML-based database, you'll have more data block changes than if the database was mainly supporting queries.

Always remember that Oracle doesn't guarantee that you can flashback your database to the flashback retention target that you set by using the `FLASHBACK_RETENTION_TARGET` init parameter. Oracle stores the flashback logs you need for a flashback database operation in the flash recovery area. If Oracle is running low on free space in the flash recovery area for newly arriving archived redo log files, it will remove some flashback logs to make room. That's why it is absolutely essential to monitor the flash recovery area size to ensure you don't lose any critical flashback logs necessary to recover your database.

Flashback Database Considerations

There are several considerations that you must be aware of when you use the flashback Database feature. Here's a summary:

- All commands of the Flashback Database feature work the same in RMAN as well as SQL*Plus. The only difference is that RMAN will automatically retrieve the necessary archived redo logs.
- If you've lost a datafile or for some reason can't use a datafile, you can't use Flashback Database to recover. You must have all the datafiles to flashback your database to a past period.
- If a datafile was resized during the time span covered by the Flashback Database operation, you can't flashback that datafile. Instead, you must offline that particular datafile before you start the flashback database operation.
- If a control file has been restored or re-created during the time span you want to flashback over, you can't use the Flashback Database feature.
- For good performance, Oracle recommends using fast file systems that don't use operating system file caching. Large flashback database logs add to your overhead when using operating system file caching.

- To increase disk throughput, Oracle recommends using multiple disk spindles, with small stripe sizes (128K) for flash recovery areas.

exam watch

If you want to undo the effects of a Flashback Database operation, you must recover the database by rolling it forward. Flashback Database rolls back the database. To undo it, you therefore must roll forward the database.

- You can't flashback a database to before a RESETLOGS operation.
- You can't flashback a datafile that was dropped during the time span covered by the flashback table operation.
- You can't flashback a datafile that was shrunk during the time span covered by the flashback table operation.

Configuring Flashback Database

You need to perform a set of operations before you can use the Flashback Database feature. Let's review these operations here:

1. Ensure that your database is in the archivelog mode.
2. Your database must be using the flash recovery area. (See Chapter 8 for details on configuring the flash recovery area.) If you haven't configured the flash recovery area, you must do so before you can proceed further.
3. You must use the new initialization parameter `DB_FLASHBACK_RETENTION_TARGET` to set your flashback retention target. The flashback retention target specifies how far back you can flashback your database. The value of the `DB_FLASHBACK_RETENTION_TARGET` parameter determines how much Flashback Database log data your database will store in the flashback recovery area. Of course, you must have enough space in your flash recovery area to contain these Flash Database logs. Here's an example that shows how to set your flashback target to 1 day (1440 minutes):

```
SQL> ALTER SYSTEM SET
  2  DB_FLASHBACK_RETENTION_TARGET=1440;
System altered.
SQL>
```

4. Shut down the database and restart in the `MOUNT EXCLUSIVE` mode. (If you are using a single instance, the simpler `MOUNT` command will do.)

```
SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup mount;
ORACLE instance started.
Total System Global Area  117440512 bytes
Fixed Size                   787728 bytes
Variable Size              95419120 bytes
Database Buffers           20971520 bytes
Redo Buffers                 262144 bytes
Database mounted.
SQL>
```

5. Turn the flashback database feature on with the following command:

```
SQL> alter database flashback on;
Database altered.
SQL>
```

6. Use the `ALTER DATABASE OPEN` command to open the database. To confirm whether the flashback database feature is enabled, issue the following query:

```
SQL> select flashback_on from v$database;
FLA
---
YES
SQL>
```

exam watch

You can find out if the flashback feature is enabled by querying the `V$DATABASE` view. The three things you must do to configure the flashback database are as follows: configure the flash recovery area, set the `DB_FLASHBACK_RETENTION_TARGET` parameter, and finally, use the `ALTER DATABASE FLASHBACK ON` command. You can turn the feature off by using the `ALTER DATABASE FLASHBACK OFF` command. Make sure you restart the database in the `MOUNT EXCLUSIVE` mode before using this command.

on the job

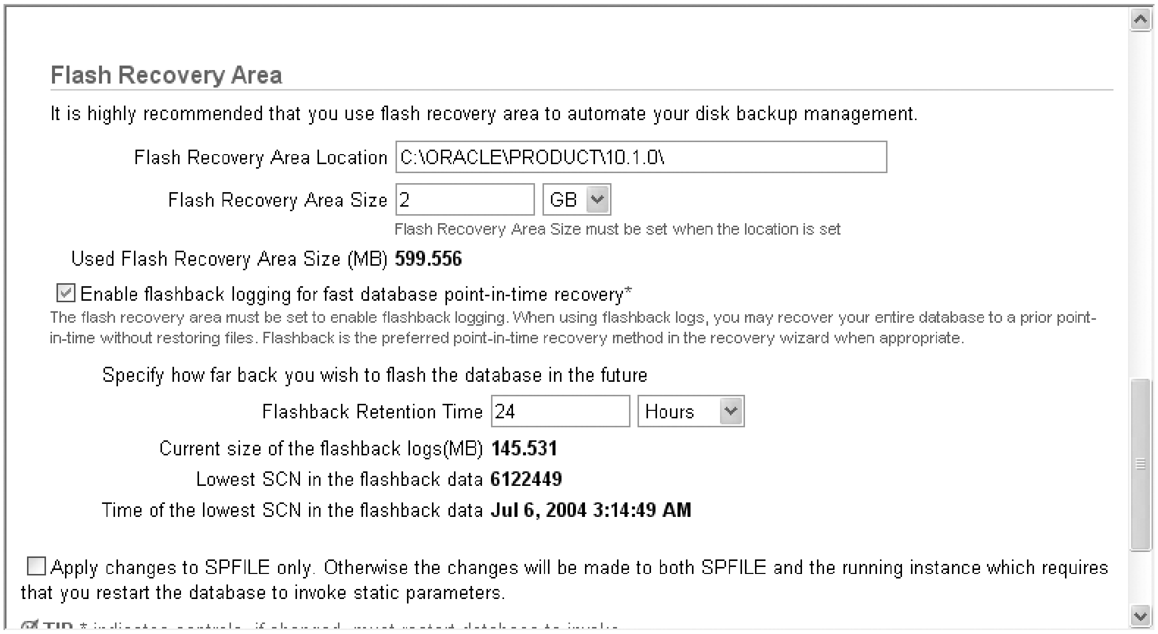
When you disable the flashback database feature, Oracle deletes all flashback database logs in the flash recovery area.

Configuring Flashback Database Using Database Control

If you want to take the easy way out, you can use the OEM Database Control tool to configure flashback logging in your database. Here are the steps you must follow. (Make sure you have archive logging turned on.)

1. From the Home page, click on the Maintenance tab.
2. Go to the Backup/Recovery section and click on Configure Recovery Settings.
3. Under the Flash Recovery Area section, first specify the flash recovery area destination and size.
4. Check the box next to "Enable flashback logging for fast database point-in-time recovery," as shown in Figure 9-1.

FIGURE 9-1 Configuring Flashback Database using Database Control



Flash Recovery Area

It is highly recommended that you use flash recovery area to automate your disk backup management.

Flash Recovery Area Location: C:\ORACLE\PRODUCT\10.1.0\

Flash Recovery Area Size: 2 GB

Flash Recovery Area Size must be set when the location is set

Used Flash Recovery Area Size (MB) **599.556**

✓ Enable flashback logging for fast database point-in-time recovery*

The flash recovery area must be set to enable flashback logging. When using flashback logs, you may recover your entire database to a prior point-in-time without restoring files. Flashback is the preferred point-in-time recovery method in the recovery wizard when appropriate.

Specify how far back you wish to flash the database in the future

Flashback Retention Time: 24 Hours

Current size of the flashback logs(MB) **145.531**

Lowest SCN in the flashback data **6122449**

Time of the lowest SCN in the flashback data **Jul 6, 2004 3:14:49 AM**

☐ Apply changes to SPFILE only. Otherwise the changes will be made to both SPFILE and the running instance which requires that you restart the database to invoke static parameters.

TIP * indicates controls, if changed, must restart database to invoke

Flashback Storage Issues

Flashback database logs (stored in the flash recovery area) make it possible for you to perform a flashback database operation. The flash recovery area is primarily meant as a storage location for recovery-related files like datafile copies, incremental backups, and archived redo logs. The database accords priority to storing these recovery-related files over retaining flashback database logs. Accordingly, the database will delete flashback database logs if it needs the flash recovery area space to accommodate new recovery-related files. Therefore, you must ensure that you have allocated sufficient space to your flash recovery area to hold all your flashback database logs.

The value of the DB_FLASHBACK_RETENTION_TARGET parameter (1440 minutes in our example) determines how much flashback data the database should retain in the form of flashback database logs in the flash recovery area. In order to estimate the space you need to add to your flash recovery area for accommodating the flashback database logs, first run the database for a while with the flashback database feature turned on. Then run the following query:

```
SQL> select estimated_flashback_size, retention_target,
flashback_size from  v$flashback_database_log;
ESTIMATED_FLASHBACK_SIZE    RETENTION_TARGET    FLASHBACK_SIZE
------------------------ ---------------- ----------------------
         126418944                 1440              152600576
SQL>
```

on the job

If there isn't sufficient data to perform a flashback all the way back to where you want to take the database, you can use standard recovery procedures to recover the database.

The V$FLASHBACK_DATABASE_LOG view provides information that helps you estimate the amount of flashback data needed for your current database workload level of DML activity (since queries don't produce any flashback data). Let's understand the three key columns in the V$FLASHBACK_DATABASE_LOG view:

- RETENTION_TARGET shows you the target retention time in minutes.
- ESTIMATED_FLASHBACK_SIZE shows you the estimated size of flashback data needed to satisfy the value you specified for the RETENTION_TARGET parameter. Oracle bases its flashback data estimates on either the DML changes in the database since the instance started, or the most recent interval of time equal to your retention target. Oracle chooses the shorter of these two choices to prepare its flashback data estimate.
- FLASHBACK_SIZE column shows the current size, in bytes, of the flashback data.

Although the query on the V$FLASHBACK_DATABASE_LOG view enables you to come up with an estimate of the disk space for the flashback database logs, that isn't a guarantee that the space will meet your needs. To really know how far back you can flashback your database at any given time, you must query the V$FLASHBACK_DATABASE_LOG in the following manner:

```
SQL> select oldest_flashback_scn,
     oldest_flashback_time
    from v$flashback_database_log;
OLDEST_FLASHBACK_SCN OLDEST_FLASHBACK_
-------------------- -----------------
             5964669 07-03-04 12:22:37
SQL>
```

If the query results indicate that you can't wind back your database as far as you would like, you must increase the size of your flash recovery area. You do this so the flash recovery area can hold a larger amount of flashback database logs.

exam watch

Just because you have set your retention target at a high value doesn't ensure that Oracle will guarantee the retaining of flashback database logs to satisfy the retention target. If the flashback recovery area gets full, Oracle will remove the flashback database logs to make room for new recovery files.

The database incurs an I/O overhead while logging the flashback data. The view V$FLASHBACK_DATABASE_STATS helps you monitor the I/O overhead of logging flashback data. Here's the structure of the V$FLASHBACK_DATABASE_STATS view:

```
SQL> desc v$flashback_database_stat;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
 BEGIN_TIME                                         DATE
 END_TIME                                           DATE
 FLASHBACK_DATA                                     NUMBER
 DB_DATA                                            NUMBER
 REDO_DATA                                          NUMBER
 ESTIMATED_FLASHBACK_SIZE                           NUMBER
SQL>
```

Let's take a closer look at the elements in the V$FLASHBACK_DATABASE_STAT view:

- BEGIN_TIME and END_TIME stand for the beginning and ending hourly time intervals for which the view's statistics were collected. Oracle collects flashback data on an hourly basis for a maximum of 24 hours. If you issue a query on the table, however, it may return 25 rows, the 25th row being for the most recent fraction of time after the last (24th) row was logged in the view.
- FLASHBACK_DATA stands for the number of bytes of flashback data written during the interval.
- DB_DATA stands for the number of bytes of database data read and written during the interval.
- REDO_DATA stands for the number of bytes of redo data written during the interval.
- ESTIMATED_FLASHBACK_SIZE is identical to the value of the ESTIMATED_FLASHBACK_SIZE column in the V$FLASHBACK_DATABASE_LOG view.

The V$FLASHBACK_DATABASE_STAT view helps you monitor how your flashback data generation is changing across a day's time. You can adjust either or both of your retention target and flash recovery area size based on the statistics provided by this view.

If you don't want certain tablespaces to be part of your flashback operations, you can specify that Oracle not log any flashback database data for these tablespaces. You can specify turning off the flashback database feature at the tablespace level when creating the tablespace itself, or later on, by using the ALTER TABLESPACE command. Here is an example:

```
SQL> alter tablespace users flashback off;
Tablespace altered.
SQL>
```

exam watch

When you turn the flashback database feature off for a tablespace, as shown in the previous example, make sure that you take the tablespace offline. Otherwise, Oracle will issue an error when you try to use the FLASHBACK DATABASE command.

Flashback Database Examples

When you issue the FLASHBACK DATABASE statement, Oracle will first check that all the archived and online redo log files spanning the entire flashback period are available. It automatically reverts all the currently online datafiles to the SCN or time you specify in the FLASHBACK DATABASE statement.

Let's look at a few examples to illustrate the use of the FLASHBACK DATABASE command. In the first example, I first create a table called persons and load it with some test data. I then use the flashback database feature to effortlessly get back to a specified SCN. The example follows.

First, I check for the number of rows in the persons table.

```
SQL> select count (*) from persons;
     COUNT(*)
----------
     32768
```

I note what the current SCN of the database is with the following SQL command:

```
SQL> select current_scn from v$database;
CURRENT_SCN
-----------
    5965123
```

I then double the number of rows in our test table, persons, I check the current number of rows, and I get the following answer:

```
SQL> select count (*) from persons;
     COUNT(*)
----------
     65536
```

My goal is to get back to the time when the row count in the persons table was 32768. I can do this easily by flashing back to the SCN 5965123. To use any flashback feature, I must restart the database in the MOUNT (exclusive) mode, as shown here:

```
SQL> startup mount;
ORACLE instance started.
…
Database mounted.
```

I can now turn the flashback database feature on by using the following command:

```
SQL> flashback database to  SCN 5964663;
Flashback complete.
```

Note that TO SCN takes the database back to its state at that SCN. You can also take a database back to its state just before an SCN by using the TO BEFORE SCN clause. Alternatively, you may use the TO_TIMESTAMP or TO BEFORE TIMESTAMP to revert the database to a specified timestamp, or one second before the specified timestamp.

In order to query the persons table, I must first open the database, which I try to do here:

```
SQL> alter database open;
alter database open
*
ERROR at line 1:
ORA-01589: must use RESETLOGS or NORESETLOGS option for database open
```

I then use the ALTER DATABASE OPEN RESETLOGS command to open the database. In order to have a write access to the database that I just flashed back, I must reopen the database with an ALTER DATABASE OPEN RESETLOGS statement. To make sure that you have indeed used the current SCN or target time, you may first want to use the ALTER DATABASE OPEN READ ONLY statement. Once you confirm that you have flashed the database back to the correct point in time or the correct SCN, you can finalize matters by using the ALTER DATABASE OPEN RESETLOGS statement. If after the initial check (in the read-only mode), you decide to go back further in time, you can flashback the database again. If you determine that you flashed back too far into the past, you can use redo logs to roll forward. The ALTER DATABASE OPEN RESETLOGS statement should be used only after you are finally satisfied about the correctness (timewise)of the data.

```
SQL> alter database open resetlogs;
Database altered.
```

You can verify that your database has been flashed back by using the following query on the persons table:

```
SQL> select count(*) from persons;
     COUNT(*)
    -----------
     32768
```

In the previous example, I used an SCN to tell the database how far back it should flashback. However, you may use an archived log sequence, or a prior time as well, to specify the flashback point. Here are some examples using time and log sequence numbers:

```
SQL> flashback database to sequence 12345; /* will flashback
the database to the log sequence numbered 1234) */
SQL> flashback database to timestamp (SYSDATE -1/24); /* will flashback the database to an hour ago */
```

exam watch

Once you flashback the database to a certain point in time, you can flashback the database to a different time if you think you used the wrong flashback target time the first time around. If you want, you can perform a recovery to roll the database forward, after the flashback operation. If you want to completely undo the effects of the flashback database operation, just use the command `RECOVER DATABASE` to perform a complete recovery of the database.

EXERCISE 9-1

Using the Flashback Database Feature

List the contents of your flash recovery area. Configure the flashback database feature in your database. Check the current SCN of your database. What is the oldest SCN you can flashback your database to?

CERTIFICATION OBJECTIVE 9.03

Flashback Drop

One of the biggest occupational hazards of a DBA's job is the fact that you could inadvertently drop a key production table. There are countless stories of how DBAs have dropped or truncated the wrong table in the wrong database, with disastrous consequences, at an organizational as well as a personal level. Well, Oracle Database 10*g* finally takes this hazard away, since you can't mistakenly drop a table anymore! When you drop a table by using the command `DROP TABLE PERSONS`, for example, the persons table doesn't go away immediately, as before. If you change your mind or realize you've made a mistake, you don't have to perform a database point-in-time recovery. You simply use the following magical command to get your "lost" table back:

```
SQL> FLASHBACK TABLE persons TO BEFORE DROP;
```

In an Oracle 10*g* database, when you drop a table, Oracle doesn't automatically drop the space allocated to the table. It simply renames and stores the table and its dependent objects in the database's recycle bin. Using the Flashback Drop operation as shown in the previous example, you then recover the dropped table from the recycle bin.

How the Flashback Drop Feature Works

In previous versions of Oracle, when you used the `DROP TABLE` command, Oracle immediately dropped the table and all its dependent objects. The database also released all the space in that table segment back to the database. In Oracle Database 10*g*, when you drop a table, the table is gone in name only—the table itself, with all its rows, is merely renamed and stays in the same location as before. If you query the `DBA_FREE_SPACE` view, you'll see additional space, indicating that the table segment has given up its space. However, this additional free space is merely the potential free space you can possibly reclaim, if you *really* get rid of the table later on. Thus, when you drop a table or an index, the space occupied by the objects is not reclaimable immediately by the database, although you can see the space as free space in the `DB_FREE_SPACE` view. Oracle reclaims the space only if it experiences space pressure. Oracle's goal here is to retain the dropped objects for the maximum time possible.

As stated earlier, when you issue the `DROP TABLE` command, Oracle merely renames the table and moves it to a recycle bin. The recycle bin is not an area on disk, unlike your flash recovery area. The recycle bin is merely a data dictionary table that maintains information about dropped tables, such as their original names and the new names that Oracle gives them when it temporarily renames them and places them in the recycle bin.

All objects that you drop will first sit in the recycle bin and continue to take up their original space allocation in their tablespaces. Oracle will permanently remove them from the recycle bin—and therefore, from the database—when either a user purges them or Oracle automatically reclaims the space. To summarize:

- You remove a table permanently with a PURGE command. You can use the PURGE command as an option with the DROP TABLE (or INDEX) command. You may also use the command to drop an object currently saved in the recycle bin.

Or

- Oracle automatically removes the dropped objects in the recycle bin because of space pressure. *Space pressure* is said to occur when there isn't enough free space in a tablespace when you need to create a new object or extend more space to an existing object. Oracle first allocates all the available free space in the tablespace to the new object. When it runs out of available free space, Oracle will start automatically purging the occupants of the recycle bin to make room in the tablespace.

Querying the Recycle Bin

When you drop a table, Oracle lists the object in the recycle bin, which is a data dictionary table named RECYCLEBIN$. You can view the contents of the recycle bin by using either the DBA_RECYCLEBIN (database level) view or the USER_RECYCLEBIN (user level) view. The DBA_RECYCLEBIN shows all dropped database objects, provided they are still in the recycle bin. Here is an example that shows how to retrieve information about any dropped objects in your database:

```
SQL>  select owner,original_name,object_name,
  2   ts_name,droptime
  3*  from dba_recyclebin;
OWNER   ORIGINAL_NAME      OBJECT_NAME                          TS_NAME
-----------------------------------------------------------------------
sam        PERSONS     BIN$xTMPjHZ6SG+1xnDIaR9E+g==$0          USERS
```

exam watch

There could be several recycle bins, each owned by the person who dropped an object. The `USER_RECYCLEBIN` view displays information about the recycle bin owned by the current user. The `DBA_RECYCLEBIN` view displays information about all recycle bins in the database.

The DBA_RECYCLEBIN (and the USER_RECYCLEBIN) view shows details about objects in all recycle bins in the database. Two important columns of the view are the CAN_UNDROP and CAN_PURGE columns, which tell you whether you can undrop and purge an object, respectively.

You can also use the command SHOW RECYCLEBIN to view the contents of the recycle bin, as shown here:

```
SQL> show recyclebin
ORIGINAL NAME      RECYCLEBIN NAME           OBJECT TYPE     DROP TIME
---------------- ------------------------------ ------------ -------------
PERSONS        BIN$xTMPjHZ6SG+1xnDIaR9E+g==$0   TABLE    2004-07-
04:12:08:57
SQL>
```

on the job

The SHOW RECYCLEBIN command shows only those objects that you can undrop.

When Oracle moves a dropped table to the recycle bin it assigns the table a system-generated name, which is usually 30 characters long. If you look in the DBA_TABLES, DBA_OBJECTS, and DBA_SEGMENTS views, you are still listed as the owner of the table. The reason is that as long as the dropped table is still in the recycle bin, you can recover it by using the FLASHBACK TABLE command.

How can you tell by looking at the DBA_TABLES view which of your tables are in the recycle bin with system given names, and which are not? A simple query on DBA_TABLES will resolve this issue, as shown here:

```
SQL> select table_name, dropped from dba_tables;
```

exam watch

The recycle bin helps you recover dropped tables only. If you truncate a table, you can't recover it using the recycle bin. If you truncate a large table, you might want to use the flashback database feature.

Any table that was dropped and is still in the recycle bin will show a YES value for the DROPPED column, and NO otherwise.

If you wish, you can query tables in the recycle bin. Understand that the recycle bin knows your table by its system-generated name only—therefore, your query must include not the original name of the dropped table, but the new name given it while it is temporarily stored in the recycle bin. Here's an example that shows you how to query the dropped table persons. (Make sure you enclose the system name of the dropped table in double quotes.)

```
SQL>  select * from "BIN$ksisyyg0TxKnt18rqukpQA==$0";
NAME
--------------------

Valerie alapati
sam alapati
nina alapati
nicholas alapati
shannon alapati
SQL>
```

exam watch

You can use the SELECT command to query the objects in the recycle bin. You can't use INSERT, UPDATE, and DELETE commands on these objects.

exam watch

Oracle renames all objects in the recycle bin, including the dependent objects like indexes, triggers, and constraints. When you flashback a table, Oracle will recover the dependent objects as well, but they'll continue to have their cryptic system-generated names. You may want to rename these objects to names that follow your organization's object naming conventions after you complete the flashback table operation.

Restoring Dropped Tables

The amazing power of the Flashback Drop feature lies in its ability to effortlessly recover dropped tables and indexes. In order to restore a dropped table, you use the `FLASHBACK TABLE …TO BEFORE DROP` command, after making sure that your table is listed as part of the recycle bin. You can use either the original name of the table you dropped or the new system-generated name when you recover a table and its dependent objects from the recycle bin. Here is an example using the original name of the table to restore it. (In the example, I first dropped the persons table, which had 10240 rows.)

```
SQL>  flashback table persons
  2*  to before drop;
Flashback complete.
```

If you wish, you may use the system-generated name for the flashback table operation, as shown here:

```
SQL> flashback table "BIN$ksisyyg0TxKnt18rqukpQA==$0"
  2  to before drop;
Flashback complete.
SQL>
```

You can also use the flashback operation as an opportunity to rename your dropped table upon recovery, using the following command. (You may use the same command with the original name of the table in the first line, instead of the system-generated name.)

```
SQL> flashback table "BIN$ksisyyg0TxKnt18rqukpQA==$0"
     to before drop
     rename to  NEW_PERSONS;
```

exam watch

When you recover a table from the recycle bin using the `FLASHBACK TABLE... TO BEFORE DROP` command, Oracle removes the object from the recycle bin.

You may find this ability to rename a table within the flashback command very helpful when you've already created a new table with the same name as the dropped table.

If you drop and re-create a table with the same name, the recycle bin will have several versions of the dropped table, each with a unique system-generated table name. If you then issue a `FLASHBACK TABLE … TO BEFORE DROP` command, Oracle will simply recover the *latest version* of the table. If you don't want Oracle to do this, you have the following options:

- In the `FLASHBACK TABLE` command, provide the specific system-generated name of the table you want to recover.
- Keep issuing the `FLASHBACK TABLE` command until you recover the particular table you want.

Permanently Removing Tables

What do you do when you are sure that you want to get rid of a table permanently? When you issue the `DROP TABLE` command, you can specify the new `PURGE` option, which will ensure that Oracle removes the table permanently and immediately, without moving it to the recycle bin. The `PURGE` clause comes in especially handy when you want to drop a sensitive table and don't want it to appear in the recycle bin for security purposes. Here's how you use the PURGE command:

```
SQL> drop table persons purge;
Table dropped.
SQL>
```

on the job

The DROP TABLE …PURGE command is the equivalent of the DROP TABLE command in previous versions of Oracle. Using the PURGE clause is equivalent to first dropping the table and then purging it from the recycle bin.

exam watch

Once you remove an object from the recycle bin with the PURGE command, or if you drop an object with the PURGE command, you can't apply the flashback drop feature with those objects (or their dependent objects)—the purged objects are gone forever!

Let's say you drop the persons table from the tablespace USERS. Now the persons table becomes a part of the recycle bin. When a user creates a new table in the USERS tablespace, Oracle assigns free space that doesn't correspond to the dropped persons table. If there isn't enough free space in the USERS tablespace to create the new table, Oracle will automatically purge objects from the recycle bin. If there are several objects in the recycle bin, Oracle first removes the oldest objects that it had placed in the recycle bin. If the space pressure continues even after purging tables from the recycle bin, Oracle may extend the tablespace, provided it is autoextensible.

Here is how you can use the PURGE command to permanently remove various objects from your recycle bin:

- PURGE TABLE or INDEX will remove the table or index. You can also use the DROP TABLE … PURGE command, but that command applies to objects that aren't a part of the recycle bin. You may also drop an object by using the recycle bin name of the object, for example:

  ```
  SQL> purge table "BIN$Q1qZGCCMRsScbbRn9ivwfA==$0"
  Table purged.
  SQL>
  ```

on the job

If you have several tables of the same original name in the recycle bin, the PURGE command will drop the first table that you originally dropped.

- PURGE TABLESPACE removes all objects that are part of that tablespace. Of course, if you remove a table, all of its dependent objects, such as indexes, will be dropped as well. When you issue the PURGE TABLESPACE command, the dependent objects of a table that live in other tablespaces will be removed as well. If you want to remove all objects of a single user, scott, for example, from the recycle bin, you may use the following command:

```
SQL> purge tablespace  users user scott;
```

- If you drop a tablespace with the command DROP TABLESPACE …INCLUDING CONTENTS, all the tablespace's objects will be dropped permanently right away, without being placed in the recycle bin. In addition, any objects belonging to the tablespace that are part of the recycle bin are purged immediately. If you use the DROP TABLESPACE command by itself, without the INCLUDING CONTENTS clause, the tablespace must not have any objects in it. Otherwise, the command will fail. In addition, if there are any objects in the recycle bin that originally belonged to this tablespace, those objects will be purged from the recycle bin.

exam watch

Understand the difference between the DROP USER and DROP USER …CASCADE command. Similarly, understand the difference between a DROP TABLESPACE and a DROP TABLESPACE …INCLUDING CONTENTS command.

- If you use the command DROP USER … CASCADE, Oracle immediately drops the user and all objects owned by the user from the database. Any objects in the recycle bin that belong to that user are automatically purged.

exam watch

When you drop a user, Oracle drops all the user's objects permanently. In addition, Oracle permanently removes any of the user's objects that are in the recycle bin as well.

- PURGE RECYCLEBIN or PURGE USER_RECYCLEBIN will remove all objects belonging to the user issuing the command.
- PURGE_DBA_RECYCLEBIN will remove all objects in the recycle bin. You must have the SYSDBA privilege to purge the entire recycle bin.

EXERCISE 9-2

Using the Flashback Drop Feature

Create a test table with a couple of indexes. Drop the table with the traditional DROP TABLE command. Re-create the table and drop it again. Repeat this process a few times. Show the ways in which you can retrieve the very first table you dropped.

Restrictions on Flashback Drop

Certain restrictions apply to the use of the Flashback Drop feature. Let's look at the important restrictions here:

- You can use the Flashback Drop feature on any table that belongs to any non-SYSTEM, locally managed tablespace.
- Dependent objects can be in either a locally or dictionary managed tablespace, to be stored in the recycle bin.
- The following types of dependent objects aren't saved in the recycle bin when you drop a table:
  - Materialized view logs
  - Referential integrity constants
  - Bitmap join indexes
- You can't save a table that has fine-grained auditing (FGA) or Virtual Private Database policies defined on it.

exam watch

The table that you want to flashback must be in a non-SYSTEM, locally managed tablespace. However, the dependent objects of that table can be in a dictionary or locally managed tablespace.

CERTIFICATION OBJECTIVE 9.04

Flashback Table

Oracle's Flashback Table feature lets you perform an online, point-in-time recovery on one or more tables. The `FLASHBACK TABLE` statement lets you restore an earlier state of a table to recover from human or application errors. For example, you may need to perform a point-in-time recovery on a table when a user mistakenly applies an update statement with the wrong WHERE clause. The Flashback Table feature relies on undo information in the database undo segments to perform the point-in-time recovery without ever having to restore any data files or apply any archived redo

exam watch

All indexes and other dependent objects of a table, such as dependent on-commit materialized views, remain intact during a flashback table operation.

log files. Ensure that you set your UNDO_RETENTION parameter to a time interval large enough to cover the longest span of time for which you might want to recover a table. You can use this feature to roll back any changes made to a table to a past SCN or time. Oracle acquires exclusive DML locks on the table or tables that it is recovering, but the tables are online and available to other users.

How Flashback Table Works

Flashback table technology uses undo information to restore data rows in changed blocks of tables. The changed information is in the undo segments, and Oracle uses the undo data to undo or roll back the table data using DML statements like INSERT, UPDATE, and DELETE. Let's review the necessary steps to perform a flashback table operation.

on the job

The flashback features aren't supported for the user SYS.

First, make sure you have all the privileges necessary to perform the Flashback Table operation. You must have either the FLASHBACK ANY TABLE or the more specific FLASHBACK object privilege on the table you want to recover. In addition, you must have the SELECT, INSERT, DELETE, and ALTER privileges on the table.

Second, make sure you enable row movement in the table, since the flashback operation doesn't preserve Oracle ROWIDs when it restores rows in the changed data blocks of the table.

```
SQL> alter table persons enable row movement;
Table altered.
```

Once you enable row movement in the table(s), you are ready to flashback the table to any time in the past, or any SCN in the past. Here's an example that shows how to flashback a table to a past SCN:

```
SQL> flashback table  persons to scn 6039341;
Flashback complete.
```

You may also specify a time to flash back to, using the following command:

```
SQL> flashback table persons to timestamp to_timestamp
('2004-07-04 08:05:00', 'YYYY-MM-DD HH24:MI:SS');
```

The previous `FLASHBACK TABLE` command restores the persons table to 8.05 A.M. on July 4, 2004.

The persons table continues to be online and accessible to users for all queries. However, Oracle acquires exclusive DML locks on the table during the Flashback Table operation. The restore is an in-place, online operation and thus doesn't involve any offlining of the datafiles or tablespaces, a procedure that's common during traditional point-in-time recovery. However, Oracle disables all relevant triggers by default and reenables them upon completing the table recovery. You may simply append the `ENABLE TRIGGERS` clause to your `FLASHBACK TABLE` command if you want to override this default behavior. Here's an example:

```
SQL> FLASHBACK TABLE person TO TIMESTAMP TO_TIMESTAMP
     ('2004-07-04 08:05:00', 'YYYY-MM-DD HH24:MI:SS')
     ENABLE TRIGGERS;
```

Undoing a Flashback Table Operation

Can you undo a flashback operation? Yes! If it turns out that your flashback table results aren't satisfactory, you can use the `FLASHBACK TABLE` statement again to go back to just before you were when you issued the first statement. This is why it is important to note your current SCN before using a Flashback Table operation. That way, if you ever want to undo a Flashback Table operation, you simply use the `FLASHBACK TABLE … TO SCN` statement to roll forward the table to a more recent time.

Restrictions on Flashback Table

Several restrictions apply to the Flashback Table recovery feature. Let's look at the important restrictions in this section.

on the job

The entire Flashback Table operation executes as a single transaction.

- You can't flashback a system or remote table.
- You can't flashback a table back to a time preceding any DDL operation that changes the structure of a table (for example, adding or dropping a column).

- Either the flashback operation succeeds entirely or it fails; it is a single transaction. That is, if you have several tables in a `FLASHBACK TABLE` command, Oracle will recover all tables or none.
- If there are any constraint violations during the flashback recovery, Oracle abandons the recovery process and leaves the tables as they were before the commencement of the flashback operation.
- If you shrink a table or otherwise change any of the nonstorage attributes of a table (other than storage attributes such as `PCTFREE`, `INITTRANS`, and `MAXTRANS)`, you can't flashback to a time before you made these changes.
- Oracle doesn't flashback statistics of the recovered objects. You may, therefore, wish to collect new statistics for the recovered objects after you complete the flashback operation.

CERTIFICATION OBJECTIVE 9.05

Row Level Flashback Features

Thus far in this chapter, you've seen the application of flashback techniques at a table or the database level. You can also use the flashback techniques to perform very useful tasks at the row level. The row level flashback features provide ways to select information from the past, see all the versions of a row, and change necessary row data. All row level flashback features rely on undo data. The length of time you can go back for query purposes depends on the setting of your `UNDO_RETENTION` initialization parameter. If you wish to provide your users the ability to flashback queries for a length of two hours, for example, you must set the `UNDO_RETENTION` parameter to 7220. The default value for this parameter is only 900 seconds (15 minutes). There are three types of row level flashback query features—Flashback Query, Flashback Versions Query, and Flashback Transaction Query. Let's look at these features in detail in the following sections.

exam watch

The value of the `UNDO_RETENTION` parameter determines the length of time your users can flashback their queries.

Flashback Query (SELECT…AS OF)

The Flashback Query feature isn't new to Oracle Database 10g, but I'm reviewing it briefly here for completeness. This feature retrieves data from a time in the past using the AS OF clause of the SELECT statement. The Flashback Query feature enables you to perform queries on the database at a target time in the past. For example, if a user accidentally deletes data rows during data entry, you can query the table using Flashback Query to retrieve the lost rows and reinsert them in the table. All the changes are made using simple SQL statements, and you don't need to restore and recover any datafiles.

You may need to query past data in order to undo incorrect changes, verify the past numbers in a column, or compare present data in a column with its past values. To query data from the past, simply use the normal SELECT statement with an AS OF clause. You may use SCN numbers or timestamps for this purpose. The AS OF clause ensures that Oracle returns the committed data in the table as of a previous SCN or timestamp. If you omit the AS OF clause, Oracle will retrieve data as of the current SCN or a specific SCN or clock time if you have specified either one. Here's a simple example that illustrates how you can query the past state of a table:

```
SQL> select * from  persons as of timestamp
     to_timestamp('2004-07-04 08:05:00', 'YYYY-MM-DD HH:MI:SS')
     where name = 'ALAPATI';
```

After verifying that the rows shown by your SELECT statement will correct the data entry, you may undo the damage by simply inserting the correct rows, again with the help of the SELECT … AS OF construct, this time in a subquery, as shown here:

```
SQL> insert into  employee
     (SELECT * FROM employee AS OF TIMESTAMP
     TO_TIMESTAMP('2004-07-04 08:05:00', 'YYYY-MM-DD HH:MI:SS')
     where name = 'ALAPATI');
```

Flashback Versions Query

The Flashback Versions query feature enables users to query the history of a given row in a table. For a given interval, the feature enables you to retrieve different versions of specific rows that existed during that interval.When you issue a SELECT statement using the VERSIONS clause, Oracle will return the different committed versions of the same row between two SCNs or two timestamps. You may also use the VERSIONS clause in subqueries of DML and DDL statements.

exam watch

Flashback Versions query, like all the other Oracle flashback features, relies on undo data. You must set your UNDO_RETENTION initialization parameter to an adequate interval if you want to ensure that your SELECT statement returns all the versions you are interested in.

The Flashback Versions Query feature enables the retrieval of all committed versions of the same table row between two time points. If you've updated the row ten different times, the Flashback Versions Query will get you all ten of those rows. The syntax of the versions query is as follows:

```
VERSIONS {BETWEEN {SCN | TIMESTAMP} start AND end}
```

In the preceding syntax statement, you can use the VERSIONS clause as a part of your normal SELECT statement, with an optional BETWEEN clause appended to it. Optionally, you can specify the SCN or TIMESTAMP clauses. You must specify the start and end expressions, which indicate the start time and end time of the interval for which you are seeking the different row versions.

exam watch

The Flashback Versions Query feature retrieves only committed transactions. Remember that the query retrieves both deleted and subsequently reinserted rows.

The output of a Flashback Versions Query is unlike the output of your typical SELECT statement. In addition to the column values you specify in the SELECT statement, for each version of the row, Oracle will provide you with values for a set of pseudocolumns for each row version. It is these pseudocolumns that tell you the story of when exactly a row change took place, and what exactly was done to the row at that time. Here is a brief explanation of each pseudocolumn that will be part of the flashback versions query output:

- **VERSIONS_STARTSCN and VERSIONS_STARTTIME** This pseudocolumn tells you the SCN and timestamp when this particular row was first created.

exam watch

An index-organized table (IOT) will show an update operation as a separate insert and a delete operation. Your versions query would produce both the deleted and inserted rows as two independent versions. The first version would show a D for the delete operation under the VERSIONS_OPERATION pseudocolumn (explained below), and the subsequent insert column would show an I for the same pseudocolumn.

- **VERSIONS_ENDSCN and VERSIONS_ENDTIME** These pseudocolumns tell you when this particular row expired.

- **VERSIONS_OPERATION** This pseudocolumn provides you with information as to the type of DML activity that was performed on the particualr row. The DML activities are indicated by letters: *I* stands for insert, *D* for delete, and *U* for an update.
- **VERSIONS_XID** This pseudocolumn stands for the unique transaction identifier of the transaction that resulted in this row version.

exam watch

If the VERSIONS_STARTSCN and the VERSIONS_STARTTIME are NULL, then the row was created before the lower bound specified by your BETWEEN clause. If the VERSIONS_ENDSCN and the VERSIONS_ENDTIME are NULL, this means that this row version is current when you tried to use the Flashback Versions Query operation, or the row was part of a delete operation.

Flashback Versions Query Example

Here's an example of using the Flashback Versions Query feature:

```
SQL> select versions_xid XID, versions_startscn START_SCN,
     versions_endscn END_SCN, versions_operation OPERATION,
     empname, salary from hr.emp
     versions between scn minvalue and maxvalue
     as of scn 113900
     where empno = 111;
XID              START_SCN  END_SCN   OPERATION  EMPNAME    SALARY
---------------- ---------- --------- ---------- ---------- ----------
0004000700000058 113855               I          Tom        927
000200030000002D 113564               D          Mike       555
000200030000002E 112670     113564    I          Mike       555
```

The example retrieves three versions of a row for empno (employee number) 111. The AS OF SCN of the query is 113900. That is, you want to know what versions of the row existed at this SCN. The query asks for the empname, salary and othr information for emno 111. The empno 111 is orginally to a new employee and if that employee is deleted, may be reassigned to a different employee. Thus, you may see different employee names for the same empno valuye over time. Although you see three versions of rows for empno 111, only one of them still exists at SCN 113900. Which version is it?

Read the query output from top to bottom. Pay particular attention to the START_SCN and the END_SCN columns. As you know, the START_SCN pseudocolumn tells you when a row was created. The END_SCN tells you when a row expired. All rows will have a START_SCN, but may have a NULL value for the END_SCN if the version of the row still exists at the current SCN.

The Flashback Versions Query asks to bring all versions as of the SCN 113900. The first row, which inserted (operation = I) the first_name TOM at 113855, is the latest version of the row. Since the END_SCN is NULL for the first row, you know that this row still exists at the SCN 113900. The middle row doesn't have an END_SCN either (NULL), so why is this row not the current incarnation of the row? If you look under the OPERATION column, you see the letter D, indicating that the middle row was deleted. The bottom or third row has an END_SCN number, so clearly this row expired at SCN 113564.

Note that you could have substituted the `VERSIONS BETWEEN TIMESTAMP` … for the `VERSIONS BETWEEN SCN` *nn* AND *nn* clause, if you decided to use timestamps instead of SCNs to specify the time interval for retrieving the various versions of the row.

exam watch

You must know how to tell the current, or live, version of a row when you use the Flashback Versions Query feature. You must understand when a row expires and how an END_SCN of NULL for a deleted row means that the row no longer exists.

Restrictions and Observations on Flashback Versions Query

The following requirements and limitations apply to the Flashback Versions Query feature:

- You can't use this feature to query a view.
- Your `VERSIONS` clause can't be applied across DDL operations.
- The query will ignore changes in row data that are purely physical, as is the case during a segment shrink operation.
- You can't use this feature to query external temporary and fixed tables.

on the job

If you want to query past data at a precise time, you must use an SCN. If you use a timestamp, the actual time queried might be up to three seconds earlier than the time you specify. Internally, Oracle Database uses SCNs, which are mapped to timestamps, with a granularity of three seconds. This time-to-SCN mapping gap may throw you off if you're trying to flashback to an exact time in the past that's immediately after a DDL operation. Your entire table might be gone, and you'll get a major error instead of the Flashback Versions Query results!

exam watch

If you omit the `AS OF` clause in a Flashback Versions Query, data is retrieved as of the current SCN, or the clock time or SCN if you have specified either one.

Flashback Transaction Query

Oracle Database 10*g* provides a new view called Flashback Transaction Query that lets you identify which transaction or transactions were responsible for certain changes during a certain interval. A Flashback Transaction Query is a query on the `FLASHBACK_TRANSACTION_QUERY` view. Using a Flashback Transaction Query, you can obtain transaction information, including the necessary SQL statements to undo all the changes made by a transaction. This feature enables you not only to correct logical errors, but also conduct transaction audits in your database.

You are probably well aware of the use of Oracle's LogMiner tool to undo SQL statements. However, LogMiner has a serious drawback: Oracle must serially read entire redo log files to get to the necessary information. The Flashback Transaction Query feature lets you use an indexed access path to quickly get to the necessary undo data directly, instead of traversing through an entire redo log file. In addition, you can correct the effects of a single transaction or a set of bad transactions during an interval of time.

Using the Flashback Transaction Query Feature

The Flashback Transaction Query feature lets you view all database chages at the transaction level, in order to recover from suer errors or to audit transactions..A new Oracle 10*g* view called `FLASHBACK_TRANSACTION_QUERY` enables you to query past transactions and correct them, if necessary. Using the Flashback Transaction Query feature simply means querying the `FLASHBACK_TRANSACTION_QUERY` view, to gather information about changes made at the transaction level, as well as to retrieve the SQL code to undo the changes made by the unwanted transactions.

The `FLASHBACK_TRANSACTION_QUERY` view contains several useful columns that let you identify a transaction's timestamp, the identity of the user that made the transaction, the type of operations made during the transactions, as well the undo statements necessary to retrieve the original row. Here's the structure of the FLASHBACK_TRANSACTION_QUERY view:

```
SQL> desc FLASHBACK_TRANSACTION_QUERY
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
 XID                                                RAW(8)
 START_SCN                                          NUMBER
 START_TIMESTAMP                                    DATE
 COMMIT_SCN                                         NUMBER
 COMMIT_TIMESTAMP                                   DATE
 LOGON_USER                                         VARCHAR2(30)
 UNDO_CHANGE#                                       NUMBER
 OPERATION                                          VARCHAR2(32)
 TABLE_NAME
VARCHAR2(256)
 TABLE_OWNER                                        VARCHAR2(32)
 ROW_ID                                             VARCHAR2(19)
 UNDO_SQL
VARCHAR2(4000)
```

on the job

You must have the SELECT ANY TRANSACTION system privilege to query the FLASHBACK_TRANSACTION_QUERY view.

In the preceding description, the various columns stand for the following items:

- START_SCN and START_TIMESTAMP tell you when a certain row was created.
- COMMIT_SCN and COMMIT_TIMESTAMP provide information about when a transaction was committed.
- XID, ROW_ID, and the UNDO_CHANGE# help identify the transaction, the row, and the undo change number, respectively.
- OPERATION refers to the type of DML operation—update, insert, or delete .

exam watch

Sometimes you'll notice a value of UNKNOWN under the OPERATION column, instead of an INSERT, DELETE, or an UPDATE. This simply means that the transaction didn't have enough undo information to correctly identify its operation type.

exam watch ***The Flashback Transaction Query feature retrieves only committed transactions. The query retrieves the deleted and subsequently reinserted rows.***

- LOGON_USER, TABLE_NAME, TABLE_OWNER columns provide the name of the user who made the changes, the table's name, and its schema, respectively.
- UNDO_SQL gives you the exact SQL statement to undo a transaction. Here's an example of the type of data you would find under the UNDO_SQL column:

```
SQL>  select undo_sql from flashback_transaction_query
  2   where table_owner = 'SAMALAPATI';
UNDO_SQL
-----------------------------------------------------------------------
delete from "SAMALAPATI"."PERSONS" where ROWID = 'AAANZ+AAEAAAAIEAAG';
delete from "SAMALAPATI"."PERSONS" where ROWID = 'AAANZ+AAEAAAAIEAAF';
delete from "SAMALAPATI"."PERSONS" where ROWID = 'AAANZ+AAEAAAAIEAAE';
insert into "SAMALAPATI"."PERSONS"("NAME") values ('sam alapati');
insert into "SAMALAPATI"."PERSON"("NAME") values ('nina alapati');
```

The following Flashback Transaction Query uses the `FLASHBACK_VERSIONS_QUERY` view to derive the transaction ID (XID), operation type, the start and commit SCNs, the user's name, and the SQL statement to undo the transaction.

```
SQL> select opEration, start_scn, logon_user,
     undo_sql from flashback_transaction_query
     where XID = HEXTORAW('0A001A00AD020000');
 OPERATION START_SCN LOGON_USER UNDO_SQL
-------------------------------------------------------------------
DELETE     6039310 SAMALAPATI insert into
                   "SAMALAPATI"."PERSONS"("NAME") Values ('nina alapati');
SQL>
```

In our simple example, there is only a single delete operation that you need to perform, in case you wish to undo the changes made by this transaction. However, transactions usually contain several DML statements, in which case you have to apply the undo changes in the sequence that the query returns them in order to correctly recover the data to its original state.

Using Flashback Transaction Query and Flashback Versions Query Together

You've learned how you can derive a version history of the changes made to a table using the Flashback Versions Query feature. This feature provides the various versions (the *what*) of a row, along with their unique version IDs and other information such as the timestamps and SCNs when the row versions were created. However, this feature doesn't have all the necessary information to correct the changed row data using appropriate SQL statements to undo an undesirable change. The Flashback Transaction Query feature provides you with the necessary information to identify not only the type of operations performed on each version of a row, but also the necessary undo SQL (the *how*) to put the rows back in their original state.

Let's use a simple example to demonstrate how you can combine the Flashback Versions Query and the Flashback Transaction Query features to undo undesirable changes to your data.

First, let's use the Flashback Versions Query feature to identify all the row versions in a certain table that have changed in a certain time period. The following query lets you do this:

```
SQL> select versions_xid XID, versions_startscn START_SCN,
     versions_endscn END_SCN, versions_operation OPERATION,
     empname, salary FROM hr.emp
versions between SCN MINVALUE and MAXVALUE
     where empno = 111;
XID              START_SCN  END_SCN   OPERATION  EMPNAME    SALARY
---------------- ---------- --------- ---------- ---------- ----------
0004000700000058 113855               I          Tom        927
000200030000002D 113564               D          Mike       555
000200030000002E 112670     113564    I          Mike       555
3 rows selected
SQL>
```

In the previous query, let's say we identified the second row, which indicates a delete operation (D) as the culprit. By mistake, one of our users has incorrectly deleted the row. All you need to do in order to extract the correct SQL to undo this delete operation is to take the transaction ID (XID) from the Flashback Versions Query shown in the previous section, and search for it in the FLASHBACK_TRANSACTION_QUERY view. Here's the query you'll need to execute:

```
SQL> select xid, start_scn START, commit_scn commit,
     operation op, logon_user user,
     undo_sql from  flashback_transaction_query
     WHERE xid = HEXTORAW('000200030000002D');
XID               START   COMMIT  OP       USER   UNDO_SQL
----------------  -----   ------  --       ----   ------------------------
000200030000002D  195243  195244  DELETE   HR     insert into "HR"."EMP"
("EMPNO","EMPNAME","SALARY") values ('111','Mike','655');
000200030000002D  195243  195244  INSERT   HR     delete from "HR"."DEPT"
where ROWID = 'AAAKD4AABAAAJ3BAAB';
```

```
000200030000002D  195243  195244  UPDATE   HR     update "HR"."EMP"
set "SALARY" = '555' where ROWID = 'AAAKD2AABAAAJ29AAA';
000200030000002D  195243  113565  BEGIN  HR
4 rows selected
SQL>
```

Flashback Transaction Query Considerations

Here are a few cautions and restrictions involving the use of the Flashback Transaction Query feature:

- You may want to turn on minimal supplemental logging in order to support operations involving chained rows and special storage structures such as clustered tables.
- If you query a transaction involving an IOT, an update operation is always shown as a two-step delete/insert operation.
- If your Flashback Transaction Query involves a dropped table or a dropped user, it returns object numbers and user IDs instead of the object names and usernames, respectively.

INSIDE THE EXAM

It is important for you to understand the mechanism behind the various flashback features. Understand that you rely on the flash recovery area (Flashback Database), undo data (Flashback table, Flashback Transaction Query, and Flashback Versions Query). What feature enables you to recover a dropped table (the recycle bin)?

The test will contain questions relating to the flash recovery area. How do you configure the flash recovery area? How do you monitor and size the flash recovery area?

You must know when to use each of the flashback techniques. For example, what flashback technique would you use when you drop a table accidentally? Which technique is best when you truncate a large table or apply partial changes in a large table? How about when you use the wrong WHERE clause to update a table? Know how and why you can use the Flashback Transaction Query and the Flashback Versions Query features together.

The exam will contain questions relating to the recycle bin. Know the difference between the `DROP TABLE` and `DROP TABLE …PURGE` commands. The test will probe your knowledge of the recycle bin object naming conventions. How can you flashback a table

INSIDE THE EXAM

more than once? When does the database reclaim the space belonging to a dropped object? (Either use the `PURGE` command, or the database must suffer space pressure.) What happens to the recycle bin contents when you issue commands to drop a tablespace or a user? Know how to query objects in the recycle bin. What commands help you bypass the recycle bin?

You must understand why you need to enable row movement before a certain flashback operation. Can you roll back a flashback operation?

The test will contain questions on flashback-related views like `V$FLASHBACK_DATABASE_LOG` and `V$FLASHBACK_DATABASE_STAT`. What do these two views help you do? (They help you size the flashback area and set the flashback retention target.) How do you find out the flashback status for a database?

CHAPTER SUMMARY

Oracle offers you the point-in-time recovery technologies to help you go back to a past point in time in the event of a logical error. However, performing a point-in-time recovery is time consuming, and the new flashback technology offers ways to undo logical errors by simply "rewinding" the database or a table to a previous time. This chapter introduced you to the new Oracle Database 10*g* flashback techniques that enable you to perform effortless recovery at a row, table, and even the database level. You first learned how the undo data forms the bedrock of the flashback technology.You learned how to flashback a database as well as an individual table. You reviewed the new recycle bin concept, which helps you recover dropped database tables and their dependent objects.

You also learned how to use the Flashback Versions Query feature to discover all the versions of a given row. You then learned how to use the Flashback Transaction Query feature, which enables you to see the changes made by particular transactions and also to undo any undesirable changes. You can use this feature to audit certain transactions or to recover from logical errors.

✓ TWO-MINUTE DRILL

General Flashback Technology Considerations

- ❑ Flashback technology offers you ways of undoing logical errors at the database, table, and row levels.
- ❑ The guaranteed undo retention feature is critical to the functioning of several flashback features that rely on undo data.
- ❑ If you have a damaged disk or some physical corruption, you must still use the traditional recovery techniques.
- ❑ Flashback technology uses undo data to query past data as well as to recover from logical errors.
- ❑ Undo retention is the length of time Oracle retains undo data.
- ❑ By default, Oracle doesn't guarantee undo retention.
- ❑ The default undo interval under guaranteed undo retention is 900 seconds.
- ❑ You can use the `RETENTION GUARANTEE` clause to institute guaranteed undo retention, either when you create a tablespace, or later, by using the `ALTER TABLESPACE` command.
- ❑ You can use either a system change number or regular clock time to tell Oracle the time in the past that you want it to go back to.
- ❑ Oracle picks an SCN that's within three seconds of the clock time that you specify.
- ❑ The `SCN_TO_TIMESTAMP` SQL function converts an SCN to a calendar time.
- ❑ The `TIMESTAMP_TO_SCN` function converts a timestamp to its corresponding SCN.

Flashback Database

- ❑ The Flashback Database feature lets you take a database to a past point in time.
- ❑ The flashback Database feature enables point-in-time recoveries without backup files and archived redo logs.
- ❑ The size of the logical error, not the size of the database, determines recovery time when you use the flashback database feature.

- ❑ The flashback buffer logs images of altered data blocks in the database.
- ❑ Oracle uses the before images of the data blocks to reconstruct a datafile.
- ❑ The new background process RVWR is in charge of writing the contents of the flashback buffer to the flashback database logs.
- ❑ Oracle stores all Flashback Database logs in the flashback recovery area, and you can't archive them.
- ❑ You can use all Flashback Database commands the same way in both RMAN and SQL*Plus.
- ❑ If you've lost a datafile, or if you have resized a datafile during the relevant period, you can't use the Flashback Database feature.
- ❑ Since flashing back a database involves the rolling back of data to undo its effects, you must roll forward the database using redo logs.
- ❑ To configure the Flashback Database feature, you must first set up your flash recovery area and then set your retention target using the `DB_FLASHBACK_RETENTION_TARGET` parameter. You can then turn the Flashback Database feature on, with the `ALTER DATABASE FLASHBACK ON` command.
- ❑ You can turn the Flashback Database feature off with the `ALTER DATABASE FLASHBACK OFF` command.
- ❑ If you disable the Flashback Database feature, Oracle deletes all the Flashback Database logs in the flash recovery area.
- ❑ Oracle gives priority to the storing of recovery-related files (datafile copies, incremental backups, and archived redo logs) over Flashback Database logs.
- ❑ The ESTIMATED_FLASHBACK_SIZE column of the `V$FLASHBACK_DATABASE_LOG` view provides you with the estimated size of flashback data needed to satisfy your retention target.
- ❑ The FLASHBACK_SIZE column of the `V$FLASHBACK_DATABASE_LOG` view shows the current size of flashback data.
- ❑ The `V$FLASHBACK_DATABASE_STATS` view helps you monitor the I/O overhead of logging flashback data.
- ❑ You can turn flashback data collection off for individual tablespaces after taking the tablespaces offline first.
- ❑ You can flashback a database using a log sequence number, an SCN, or a time specified by `TIMESTAMP`.

Flashback Drop

❑ The DROP TABLE command doesn't drop a table right away, as in previous versions.

❑ Oracle saves a table in the recycle bin after you issue the normal DROP TABLE command.

❑ You can retrieve a dropped table by using the Flashback Drop feature.

❑ The command to retrieve a table is FLASHBACK TABLE *table_name* TO BEFORE DROP.

❑ The DBA_FREE_SPACE view shows additional free space when you drop a table, but Oracle doesn't actually reclaim this space immediately.

❑ Oracle reclaims the space occupied by an object only after either the user or the database *purges* the object permanently from the recycle bin.

❑ Oracle will try to keep dropped objects in the recycle bin as long as possible. It purges objects automatically only when it's under space pressure.

❑ You can view the contents of the recycle bin by using the DBA_RECYCLEBIN and USER_RECYCLEBIN views, or the SHOW RECYCLEBIN command.

❑ The SHOW RECYCLEBIN command only shows those objects that you can *undrop*.

❑ You can't recover a truncated table.

❑ You can issue queries against a dropped table, provided you use the system-given names.

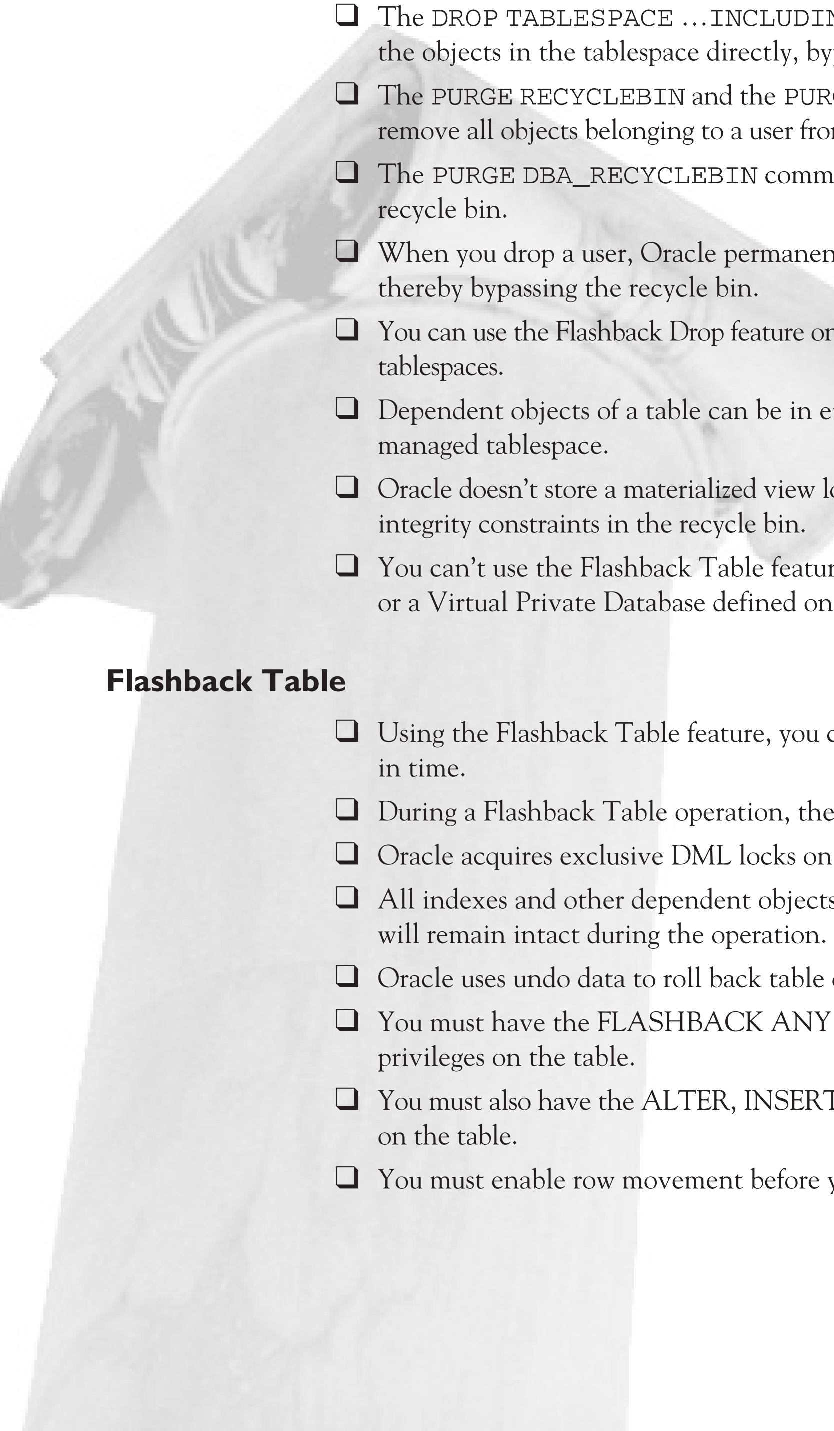
❑ When you drop any object, Oracle renames all dependent objects that it stores in the recycle bin.

❑ In the FLASHBACK TABLE … TO BEFORE DROP command, you can give the original name of the object or its system-given name after it is stored in the recycle bin.

❑ The PURGE option of the DROP TABLE command will remove a table permanently from the database or from the recycle bin, if the object is being stored there after renaming it.

❑ If there are several identically named tables in the recycle bin, the PURGE command will remove the first table that you dropped.

❑ The PURGE TABLESPACE command removes a tablespace from the recycle bin.



- The `PURGE TABLESPACE` *`tablespace_name`* `USER` *`user_name`* command will remove all objects of a user from the recycle bin.
- The `DROP TABLESPACE …INCLUDING CONTENTS` command drops all the objects in the tablespace directly, bypassing the recycle bin.
- The `PURGE RECYCLEBIN` and the `PURGE USER_RECYCLEBIN` commands remove all objects belonging to a user from the recycle bin.
- The `PURGE DBA_RECYCLEBIN` command will remove *all* objects from the recycle bin.
- When you drop a user, Oracle permanently drops all of the user's objects, thereby bypassing the recycle bin.
- You can use the Flashback Drop feature only on non-SYSTEM, locally managed tablespaces.
- Dependent objects of a table can be in either a locally managed or dictionary-managed tablespace.
- Oracle doesn't store a materialized view log, bitmap join indexes, or referential integrity constraints in the recycle bin.
- You can't use the Flashback Table feature if a table has fine-grained auditing or a Virtual Private Database defined on it.

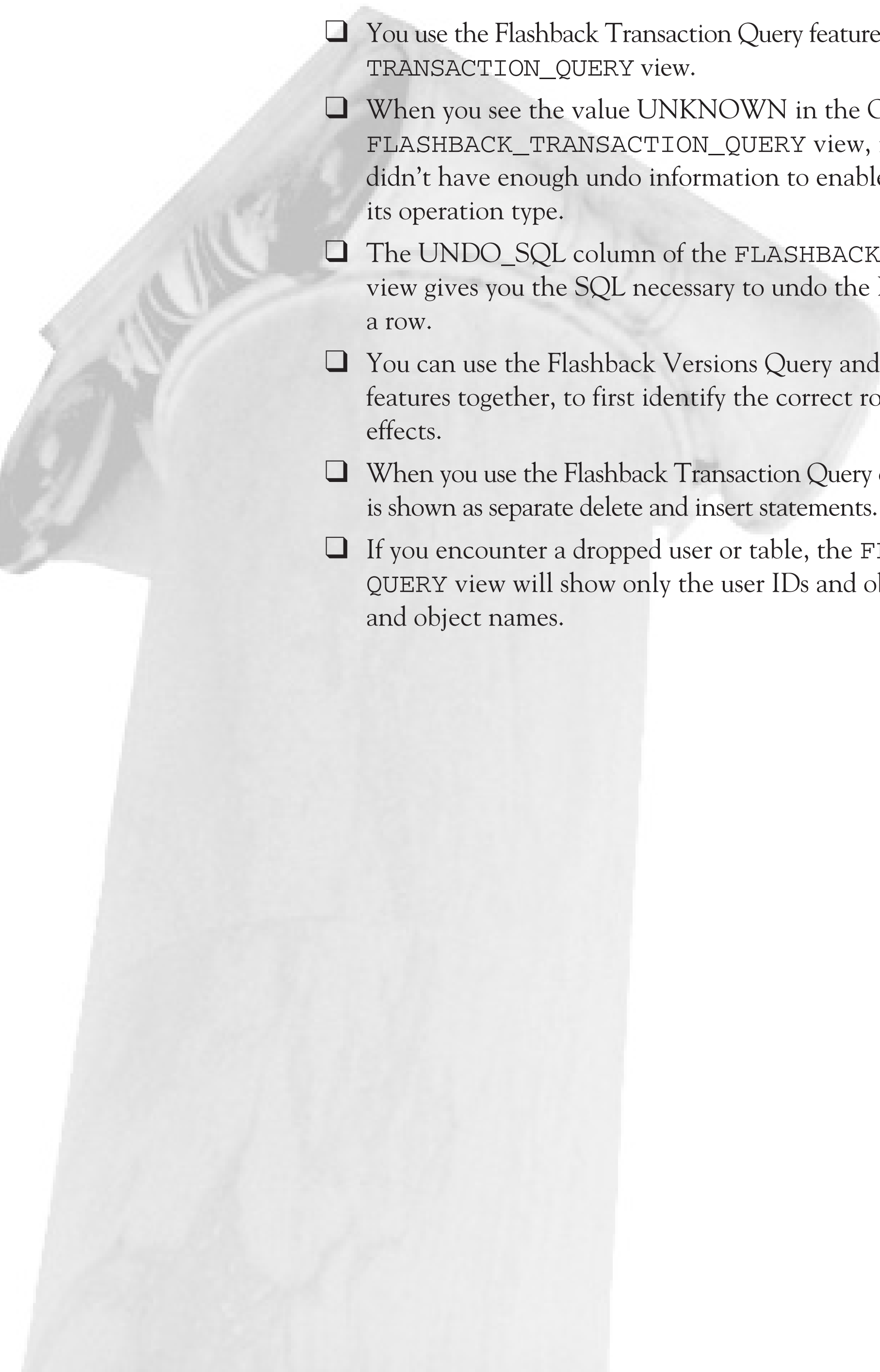
Flashback Table

- Using the Flashback Table feature, you can restore a table back to a point in time.
- During a Flashback Table operation, the tables are online and in place.
- Oracle acquires exclusive DML locks on the table it is flashing back.
- All indexes and other dependent objects of a table that is being flashed back will remain intact during the operation.
- Oracle uses undo data to roll back table data during the flashback operation.
- You must have the FLASHBACK ANY TABLE or FLASHBACK object privileges on the table.
- You must also have the ALTER, INSERT, UPDATE, and DELETE privileges on the table.
- You must enable row movement before you can flashback a table.

- You can flashback a table using a past SCN or a past time period, using the TIMESTAMP function.
- You can undo a Flashback Table operation by simply using the `FLASHBACK TABLE` command again.
- You can't flashback a system or remote table.
- The flashback operation is a single transaction, so either the entire operation succeeds, or the whole operation will fail.
- If you shrink a table or change any nonstorage attributes of a table, you can't flashback a table to a time before you made these changes.
- Oracle doesn't flashback statistics of recovered tables and indexes.

Row Level Flashback Features

- The Flashback Versions Query feature lets you query the history of a data row by selecting all versions of a row.
- Oracle returns only committed versions of a row.
- You can use SCNs or timestamps to indicate the interval for the flashback versions query feature.
- In addition to column values, Oracle will also provide values for the pseudocolumn for each row version. The VERSIONS_XID pseudocolumn gives you the unique transaction identifier for a row version.
- The VERSIONS_OPERATION pseudocolumn indicates the type of DML activity that was performed on a particular version of a row.
- You can't use the Flashback Versions Query feature to query a view, external tables, temporary tables, and fixed tables.
- You can't apply the `VERSIONS` clause across DDL operations.
- The Flashback Versions Query feature ignores purely physical changes like those that occur during a segment shrink operation.
- The Flashback Transaction Query feature focuses on the changes made to a row during a certain interval.
- The Flashback Transaction Query feature enables transaction audits and the correction of logical errors.



- ❑ The Flashback Transaction Query operation is faster than the LogMiner tool, because you use an indexed access path to quickly get to the appropriate undo data.
- ❑ You use the Flashback Transaction Query feature by querying the `FLASHBACK_TRANSACTION_QUERY` view.
- ❑ When you see the value UNKNOWN in the OPERATION column of the `FLASHBACK_TRANSACTION_QUERY` view, it means that the transaction didn't have enough undo information to enable Oracle to correctly identify its operation type.
- ❑ The UNDO_SQL column of the `FLASHBACK_TRANSACTION_QUERY` view gives you the SQL necessary to undo the DML statement that changed a row.
- ❑ You can use the Flashback Versions Query and Flashback Transaction Query features together, to first identify the correct row version and then undo its effects.
- ❑ When you use the Flashback Transaction Query on an IOT, an update operation is shown as separate delete and insert statements.
- ❑ If you encounter a dropped user or table, the `FLASHBACK_TRANSACTION_QUERY` view will show only the user IDs and object IDs, instead of usernames and object names.

SELF TEST

General Flashback Technology Considerations

1. What is the purpose of the flashback technology in Oracle Database 10*g*?

- A. Recovery from physical corruption of a data file
- B. Recovery from physical and logical corruption of a data file
- C. Recovery from logical corruption of a data file
- D. Recovery from a problem brought on by a damaged disk drive

2. The initialization parameter `UNDO_RETENTION` is used for what purpose?

- A. To specify the length of time Oracle must retain undo data in the undo tablespace
- B. To specify the length of time Oracle must retain undo data in the flash recovery area
- C. To specify the length of time Oracle must retain undo data in the data files
- D. To specify the length of time Oracle will retain undo data in the flashback database logs

3. Which one (or more) of the following is true?

- A. Oracle guarantees undo retention by default.
- B. Oracle doesn't guarantee undo retention by default.
- C. The default undo interval is 1440 minutes (1 day).
- D. The default undo interval is 900 seconds (15 minutes).

4. Which of the following statements is true? (Please choose more than one answer.)

- A. Oracle uses clock times internally, but it maps them to SCN times.
- B. Oracle uses SCNs internally, but it maps them to clock times.
- C. Oracle will pick an SCN within three seconds of the clock time you specify.
- D. Oracle will pick an SCN within one second of the clock time you specify.

Flashback Database

5. Which of the following statements is correct when you use the Flashback Database feature?

- A. You don't need to apply any archived logs at all.
- B. You need to apply only one archived log during recovery.
- C. You must apply all archived logs produced during the time interval covered by the flash database operation.
- D. You must apply archived logs only for a small amount of time, since the flashback database logs will recover to a point just before the target time.

6. What should you do before you can turn the flashback database feature on?
   - A. Start the database with the `STARTUP` command.
   - B. Start the database with the `STARTUP MOUNT` command.
   - C. Start the database with the `STARTUP MOUNT EXCLUSIVE` command.
   - D. Start the database with the `STARTUP NOMOUNT` command.
7. What happens to the flashback database logs when you disable the flashback database feature?
   - A. Oracle will delete all the flashback database logs from the flash recovery area.
   - B. Oracle will archive all the flashback database logs that are currently in the flash recovery area, and delete them afterwards.
   - C. Oracle will clear the flashback buffer area.
   - D. Oracle will leave the flashback database logs in the flash recovery area, and remove them only when it's under space pressure.
8. Which of the following views contains the ESTIMATED_FLASHBACK_SIZE column?
   - A. `V$FLASHBACK_DATABASE_LOG` view
   - B. `V$FLASHBACK_DATABASE_STATS` view
   - C. Both the `V$FLASHBACK_DATABASE_LOG` and the `V$FLASHBACK_DATABASE_STATS` views have this column.
   - D. `V$FLASHBACK_DATABASE` view

Flashback Drop

9. What is the equivalent of the old `DROP TABLE` command in Oracle Database 10*g*?
   - A. `DROP TABLE ... TO BEFORE DROP`
   - B. `DROP TABLE` works exactly the way it did in previous versions.
   - C. `DROP TABLE ... PURGE`
   - D. `DROP TABLE NO PURGE`
10. Which one (or more) of the following statements is true?
    - A. The `SHOW RECYCLEBIN` command shows all objects that you can undrop.
    - B. The `SHOW RECYCLEBIN` command shows all objects that you can undrop and purge.
    - C. The `DBA_RECYCLEBIN` view shows all objects that you can undrop and purge.
    - D. The `DBA_RECYCLEBIN` only shows items that you can undrop.
11. After you drop a table using the `DROP TABLE` command,

- A. You can query the table by using its original name only.
- B. You can query the table by using its new system-generated name only.
- C. You can query the table by using the original or system-generated name.
- D. You can't query a table once you drop it.

12. When you use the command `PURGE RECYCLEBIN`,

- A. Oracle will remove all the objects in the recycle bin that are owned by all users.
- B. Oracle will only remove objects from the recycle bin that belong to the current user.
- C. Oracle will remove only the tables belonging to the current user.
- D. You must ensure that you have the SYSDBA privilege.

Flashback Table

13. The Flashback Table feature is ideal in which of the following situations?

- A. When you drop a user
- B. When you truncate a table
- C. When you drop a table
- D. When you update a table with the wrong WHERE clause by mistake

14. What kinds of privilege(s) are necessary for you to use the flashback table feature?

- A. ALTER TABLE
- B. FLASHBACK ANY TABLE
- C. No special privileges are necessary—you must be the owner of the object.
- D. FLASHBACK TABLE privilege on the table you are flashing back

15. Which one of the following statements is correct?

- A. You must enable row movement in the table you are flashing back, because the flashback operation doesn't preserve Oracle ROWIDs.
- B. You must disable row movement in the table you are flashing back, because the flashback operation doesn't preserve Oracle ROWIDs.
- C. You must disable row movement in the table you are flashing back, because the flashback operation preserves Oracle ROWIDs.
- D. You must enable row movement in the table you are flashing back, because the flashback operation preserves Oracle ROWIDs.

16. What is the most likely outcome if Oracle violates any constraints during a flashback table operation?

A. The database continues to flashback the database, after logging a warning in the alert log.
B. The database abandons the flashback process and leaves the table in the state it is at that point in time.
C. The database abandons the flashback process and leaves the tables as they were before the start of the flashback operation.
D. You have to disable all constraints before you start a flashback database operation. Therefore, there is no chance of your violating any table constraints.

Row Level Flashback Features

17. Which row or rows does the flashback versions query generate?

A. All committed versions of a row
B. All committed and uncommitted versions of a row
C. Only the most recent committed version of a row
D. The oldest committed row

18. What do the pseudocolumns of a flashback versions query tell you?

A. When a row version was first created
B. What kind of DML operation was performed on a row
C. The UNDO statement(s) necessary to undo the results of a transaction
D. The REDO statement(s) necessary to undo the results of a transaction

19. Which of the following statement (or statements) is true?

A. The flashback transaction query relies on the `FLASHBACK_TRANSACTION_QUERY` view.
B. The flashback transaction query relies on the `FLASHBACK_TRANSACTION_QUERY` procedure.
C. The flashback transaction query cannot be used together with the flashback versions query feature.
D. The flashback transaction query can be used together with the flashback versions query feature.

20. Which of the following statements is true?

A. The VERSIONS_XID column of the `FLASHBACK_TRANSACTION_QUERY` view maps to the VERSIONS_XID pseudocolumn shown in the output of a `FLASHBACK_VERSIONS_QUERY` select statement.

B. The XID column of the FLASHBACK_VERSIONS_QUERY view maps to the VERSIONS_XID pseudocolumn shown in the output of a FLASHBACK_VERSIONS_QUERY select statement.

C. The XID column of the FLASHBACK_TRANSACTION_QUERY view maps to the VERSIONS_XID pseudocolumn shown in the output of a FLASHBACK_VERSIONS_QUERY select statement.

D. The VERSIONS_XID column of the FLASHBACK_TRANSACTION_QUERY view maps to the XID pseudocolumn shown in the output of a FLASHBACK_VERSIONS_QUERY select statement.

LAB QUESTION

Create a table with an index, constraint, primary key, materialized view log, and a trigger. Drop the table. Recover the table and its objects using the appropriate flashback techniques. Verify the results to make sure you recovered all the objects of the table.

SELF TEST ANSWERS

General Flashback Technology Considerations

1. ☑ **C.** The flashback technology is purely meant to address logical corruptions in data due to user and application errors.
   ☒ **A** and **D** are wrong because physical corruption of datafiles means that you must use the traditional recovery techniques, which involve using backup data files. **B** is wrong since flashback technology doesn't address physical corruption of data.

2. ☑ **A.** The `UNDO_RETENTION` parameter specifies the time period that undo data is saved in the undo segments, which are located in the undo tablespace.
   ☒ **B** is wrong because Oracle doesn't retain undo data in the flash recovery area. **C** is incorrect since undo data is not saved in datafiles, but in the undo tablespace. **D** is wrong since flashback database logs don't contain undo data; they contain the changed images of Oracle data blocks.

3. ☑ **B** and **D.** B is correct because Oracle doesn't guarantee undo retention by default. You have to enable undo retention explicitly. D is correct because once you enable undo retention, Oracle will save undo data for a default period of 900 seconds.
   ☒ **A** is wrong because Oracle doesn't guarantee undo retention by default. **C** is wrong because the default undo interval is 900 seconds.

4. ☑ **A** and **C.** A is correct because Oracle uses clock times internally, but it maps them to SCNs. C is correct because the time mapping granularity between Oracle's clock times and SCNs is 3 seconds. Therefore, Oracle will pick an SCN within three seconds of clock time that you specify.
   ☒ **B** is wrong because Oracle uses clock times internally and maps them to SCNs. **D** is incorrect because the time mapping granularity is three seconds, not one second.

Flashback Database

5. ☑ **D.** When flashing back a database, the Flashback Dtabase logs you must apply will take you to a point that's a little bit before your target time. You must apply archive logs to roll forward for that point on, to your target time.
   ☒ **A** is wrong since you most likely will have to use some archived log(s). **B** is wrong since there is no guarantee that a single archived log will be sufficient. The number of archived logs you need will depend on the size of the archived logs and the amount of DML changes in your database. **C** is wrong because you don't have to apply any of the logs produced during the flashback operation.

6. ☑ **B** and **C.** You must start the database in the `MOUNT EXCLUSIVE` mode before you can configure the flashback database feature. If you have a single instance, the `STARTUP MOUNT` command is sufficient. If you have a Real Application Cluster, you'll have to use the analogous

MOUNT EXCLUSIVE command. Since I didn't specify whether you are dealing with a single instance or a Real Application Cluster, both B and C would be correct alternatives.
☒ **A** is wrong since you can't open the database before turning the flashback database feature on. **D** is wrong since you have to be in the MOUNT, not in the UNMOUNT state, before turning the flashback feature on.

7. ☑ **A.** Oracle will immediately delete all Flashback Database logs once you disable the flashback database feature.
☒ **B** is incorrect since Oracle never archives Flashback Database logs. **C** is wrong because Oracle doesn't clear the flashback buffer area when you disable the Dlashback Database feature. **D** is wrong because Oracle deletes the Flashback Database logs immediately and doesn't wait until it is under space pressure.

8. ☑ **C.** Both the V$FLASHBACK_DATABASE_LOG and the V$FLASHBACK_DATABASE_STATS views contain an identical column, the ESTIMATED_FLASHBACK_SIZE.
☒ **A** and **B** are only partially correct, since they provide the name of only one of the views that contains the column ESTIMATED_FLASHBACK_SIZE. **D** refers to a nonexistent view.

Flashback Drop

9. ☑ **C.** The DROP TABLE ...PURGE command is the equivalent command to the old DROP TABLE command, since it immediately drops the table permanently.
☒ **A** is wrong since the command doesn't drop a table—it *recovers* it after you drop it. **B** is wrong because the Oracle Database 10*g* DROP TABLE command doesn't drop a table right away as in previous versions. **D** is wrong because it refers to a nonexistent command.

10. ☑ **A** and **C.** A is correct because the SHOW RECYCLEBIN command only shows the objects you can undrop. C is correct because the DBA_RECYCLEBIN view shows all the objects you can undrop and purge, under the CAN_UNDROP and CAN_PURGE columns, respectively.
☒ **B** is wrong because the command doesn't show objects you can purge. **D** is wrong because the DBA_RECYCLEBIN also shows items you can purge.

11. ☑ **B.** Once you drop a table and it's in the recycle bin, you can continue to make SELECT statements against the table (but not the UPDATE, DELETE, and INSERT commands). However, you'll have to refer to the table by its system-generated name, since there may be several tables in the recycle bin with the same original name.
☒ **A** and **C** are wrong because you can't address a table in the recycle bin by its original name. **D** is wrong since you can query after you drop it, provided it's in the recycle bin and you address it by its new, system-generated name.

12. ☑ **B.** When you use the PURGE RECYCLEBIN command (or the PURGE USER_RECYCLEBIN command), you get rid of *all objects* owned by the current user.
☒ **A** is wrong because Oracle removes only the objects owned by the user issuing the PURGE

`RECYCLEBIN` command. **C** is wrong since Oracle removes all objects belonging to the current user, not just tables. **D** is wrong because you must have the SYSDBA privilege only to use the `PURGE DBA_RECYCLEBIN` command.

Flashback Table

13. ☑ **D.** You can use the Flashback Table feature to take the table back to a previous point in time. Therefore, it is ideal in situations where you've updated a table with the wrong WHERE clause.
☒ **A** is wrong because when you drop a user (and the user's objects), you need to flashback the entire database. **B** is wrong because when you truncate a table, you are better off performing a flashback database operation. **C** is incorrect since you can use the flashback drop feature to bring back a table that you've accidentally dropped.

14. ☑ **A, B,** and **D.** A is correct because you must have the `INSERT, DELETE, UPDATE,` and `ALTER` privileges on the table to which you want to apply the Flashback Table feature. B is correct since the FLASHBACK ANY TABLE system privilege enables you to perform the operation on any table in any schema. D is also correct since the FLASHBACK TABLE privilege on the table you are flashing back is adequate to perform the operation.
☒ **C** is wrong because you do need special privileges as explained by alternatives A, B, and D.

15. ☑ **A.** You must enable row movement before you perform a flashback database operation because Oracle doesn't preserve ROWIDs during the operation.
☒ **B** and **C** are wrong since you must *enable*, not *disable*, row movement. **D** is wrong since Oracle doesn't preserve `ROWIDs`.

16. ☑ **C.** If Oracle encounters any constraint violations, it abandons the flashback process and leaves the tables as they were before the start of the operation.
☒ **A** is wrong because Oracle stops the flashback operation. **B** is incorrect since Oracle leaves the table in its original state, not the current state. **D** is wrong since you don't have to disable all constraints before you start the flashback table operation. You may leave them in an enabled state.

Row Level Flashback Features

17. ☑ **A.** The flashback versions query generates all *committed* versions of a row.
☒ **B** is wrong because the query doesn't produce any uncommitted versions of a row. **C** and **D** are wrong since Oracle will output *all* the versions of a row.

18. ☑ **A** and **B.** The pseudocolumns of a flashback versions query provide information on when a row version was first created (VERSION_STARTSCN and VERSION_STARTTIME columns) and what kind of DML operation took place (the VERSIONS_OPERATION column) among other things.

☒ **C** and **D** are wrong since the flashback versions query doesn't provide any information about undoing changes; it is the flashback transaction query that provides you with the undo information.

19. ☑ **A** and **D.** A is correct since the flashback transaction query feature relies on the `FLASHBACK_TRANSACTION_QUERY` view. D is correct since you can use the two features together to audit transactions and to correct logical transaction errors.
☒ **B** is wrong since `FLASHBACK_TRANSACTION_QUERY` is a view, not a procedure. **C** is wrong since you *can* use the two features together.

20. ☑ **C.** The XID column belongs to the `FLASHBACK_TRANSACTION_QUERY`, and it maps to the VERSIONS_XID pseudocolumn output by the flashback versions query select statement.
☒ **A, B,** and **D** refer to the wrong source for the XID column and VERSION_XID pseudocolumn.

LAB ANSWER

A user drops a table and you should get it back by using the flashback command.

1. First, connect as sysdba, enable the database flashback, and set a retention time of one day.

```
SQL> connect sys/password as  sysdba
SQL> alter system set db_flashback_retention_target = 1440; -- Ex: for two days.
SQL> alter database flashback on;
```

2. Next create your table called test_table.

```
SQL> create table test_table (
     c1 number, c2 number, c3 varchar2(50));
```

Create a simple trigger on test_table.

```
SQL> create or replace trigger test_trigger
     before delete or update or insert on
     test_table
     begin
       null
     end;
       Add a primary key constraint to your table.
SQL> alter table flashback_table
     add constraint pk1flashback_table
     primary key (c1);
```

3. Create an index on the *name* column.

```
SQL> create index name_idx
     on  test_table(name);
```

4. Create a materialized view log on test_table and perform an insert into your table.

```
SQL> create a materialized view log on test_table;
  SQL> begin
       for  this in 1..1000 loop
       insert into  test_table
       (this, this + 100, 'Test flashback…');
       end loop;
       commit;
      end;
```

5. Display all the objects and constraints that belong to the user.

```
SQL> select object_name, object_type
     from user_objects
       /
SQL> select  constraint_name, constraint_type, table_name
     from user_constraints
       /
```

6. Now, drop the test table.

```
SQL> drop table test_table flashback cascade;
```

7. Again, display all the objects and constraints that belong to you. You can see that all the deleted objects were renamed and still belong to you. Also, notice that Oracle has dropped and renamed all of the table's constraints as well..

```
SQL> select object_name, object_type
     from user_objects ;
SQL> select constraint_name, constraint_type, table_name
     from user_constraints;
```

8. Connect as SYSDBA and use the `FLASHBACK TABLE` command to recover a table and all its possible dependent objects from the recycle bin.

```
SQL> flashback table
     test_table to before drop;
```

9. Check the contents of your recycle bin.

```
SQL> select original_name, object_name, type, ts_name,
     droptime, related, space
     from dba_recyclebin
     where original_name = 'TEST_TABLE';
SQL> flashback table test_table to before drop;
```

10. Verify that your test_table has been retrieved by using the following statement:

```
SQL> select count(*) from test_table;
```

Notice that the materialized view log was not flashed back. All the other objects, including the constraints, were recovered successfully. The flashback procedures can't recover materialized view logs.