Performing User-Managed Backup and Recovery

Objectives

After completing this lesson, you should be able to:

- Describe the difference between user-managed and server-managed backup and recovery
- Perform user-managed complete database recovery
- Perform user-managed incomplete database recovery

Types of Backup and Recovery Practices

Types of database backup and recovery are:

- User-managed: Does not use RMAN.
  - Uses OS commands to move files around
  - Requires DBA to manually maintain backup activity records
- Server-managed: Uses RMAN

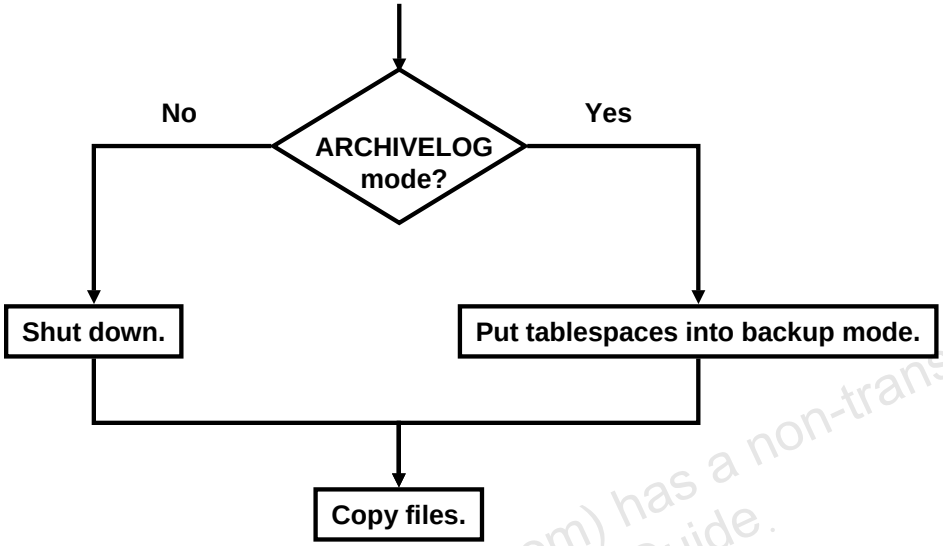
ORACLE

Types of Backup and Recovery Practices

There are two methods you can use to recover your database. You can use RMAN, and take advantage of its automatic recovery capabilities. It can restore the appropriate files and bring the database back to a current state by using very few commands. You can also recover manually. This is called *user-managed recovery*. User-managed recovery entails moving the files around using OS commands, and then issuing the recovery commands in SQL*Plus.

Both of these methods use restore and recovery processes.

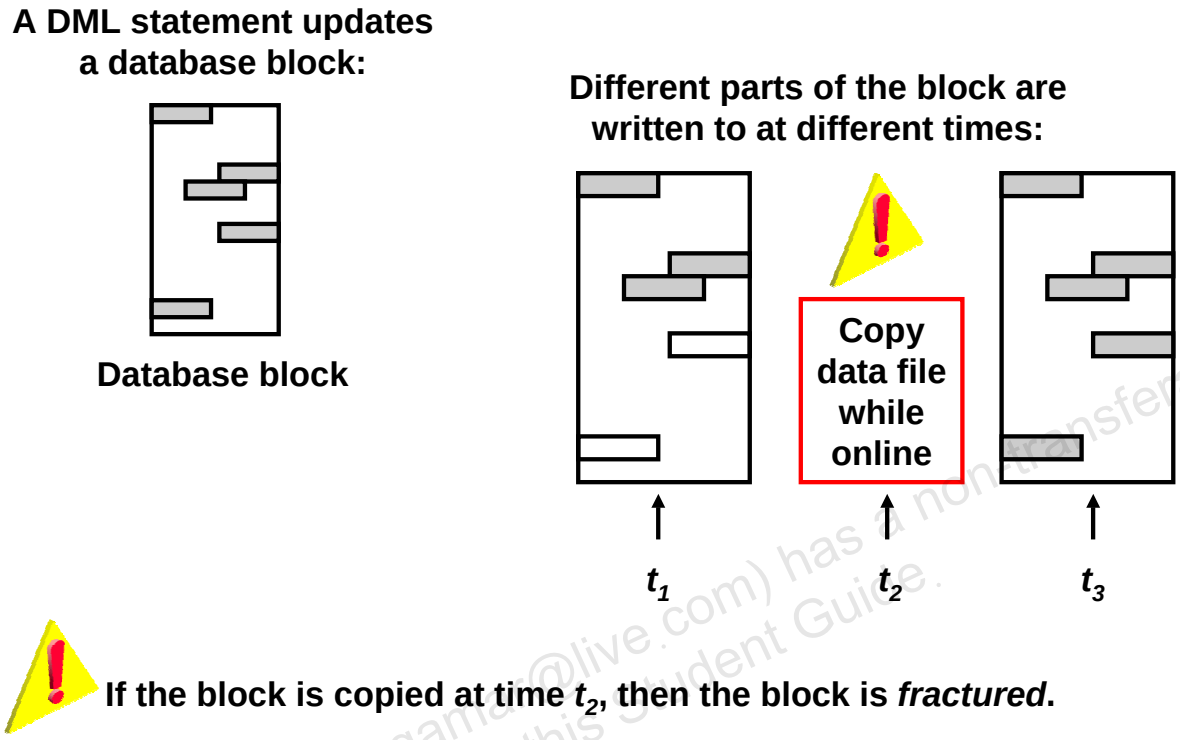
Performing a User-Managed Backup of the Database



Performing a User-Managed Backup of the Database

You can back up the database by using OS commands to make copies of the data files. The course of action depends on whether the database is in `ARCHIVELOG` mode or not. If it is, then you can keep the database open and available by putting each tablespace into backup mode before copying its data files. Otherwise, you have to shut down the database before copying the data files.

The Need for Backup Mode



The Need for Backup Mode

When a block is being written to as part of the execution of a data manipulation language (DML) statement, there could be several parts of the block affected. Not all of the modifications to the block happen at the same time, so there is the possibility of inconsistency in the block at certain times. Suppose time t_2 represents the time between when different parts of the block are written to. If, at time t_2, the block is copied as part of the execution of an OS copy command, then the block is considered fractured. Also, the OS copy command does not necessarily copy the file header first, so it must be frozen for the duration of the copy execution.

RMAN has the means to deal with this problem. If a fractured block is read, it keeps rereading it until it is consistent.

However, if an OS command such as the Linux `cp` command is copying the data file, the fractured block is not recognized as such, and the copy of the block is not consistent. In order to remedy this, put the tablespace, or even the entire database, into what is called *backup mode*. The effect of doing this is that additional redo is generated. An image of each block, before it is modified, is written to the redo log. Then, during recovery of blocks in that data file, the before image of a fractured block can be used for the basis of recovery and the additional redo data is applied to it. In order to reduce the overhead associated with maintaining extra redo data, Oracle recommends putting one tablespace at a time into backup mode, while its data files are being copied.

Identifying Files to Manually Backup

```
SQL> select name from v$datafile;

NAME
--------------------------------------------------------------------------------
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_system_36mky81f_.dbf
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_sysaux_36mky81p_.dbf
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_undotbs1_36mky857_.dbf
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_users_36mky876_.dbf
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_36ml2cmh_.dbf
/u01/app/oracle/oradata/ORCL/datafile/survey01.dbf
```

```
SQL> select name from v$controlfile;

NAME
--------------------------------------------------------------------------------
/u01/app/oracle/oradata/ORCL/controlfile/o1_mf_36ml1f8x_.ctl
/u01/app/oracle/flash_recovery_area/ORCL/controlfile/o1_mf_36ml1fkk_.ctl
```

ORACLE

Identifying Files to Manually Backup

User-managed backups require you to know the data file names and locations on disk, so you know what files need to be copied. Identify the data files to be backed up by querying the `V$DATAFILE` view. Identify the control file location by querying the `V$CONTROLFILE` view. Only one of the multiplexed control files needs to be backed up, because they are identical.

Manually Backing Up a NOARCHIVELOG Database

- Shut down the database instance:

```
SQL> SHUTDOWN IMMEDIATE
```

- Copy the data files to the backup location:

```
$ cp $ORACLE_BASE/ORCL/datafile/*.dbf \
> /u02/backup/datafile
```

- Copy the control files to the backup location:

```
$ cp $ORACLE_BASE/ORCL/controlfile/*.ctl \
> /u02/backup/controlfile
```

- Start the instance and open the database:

```
SQL> STARTUP
```

ORACLE

Manually Backing Up a NOARCHIVELOG Database

You can make a consistent, whole database backup of a NOARCHIVELOG database by shutting down the database and copying all the data files and the control file to a backup directory. Because the action of copying the files is done using OS commands (in this case, the Linux cp command), the database must be shut down first. This puts it in a consistent state. If your database is running in NOARCHIVELOG mode, this is your only option. Otherwise, in ARCHIVELOG mode, you can make inconsistent backups, which allows you to leave the database running while you take the backup.

Manually Backing Up an ARCHIVELOG Database

- Identify tablespaces and their data files:

```
SQL> select file_name, tablespace_name from dba_data_files;
FILE_NAME                                          TABLESPACE_NAME
-------------------------------------------------- --------------
/u01/app/oracle/oradata/orcl/users01.dbf           USERS
/u01/app/oracle/oradata/orcl/users02.dbf           USERS
/u01/app/oracle/oradata/orcl/undotbs1.dbf          UNDOTBS1
/u01/app/oracle/oradata/orcl/sysaux01.dbf          SYSAUX
/u01/app/oracle/oradata/orcl/system01.dbf          SYSTEM
/u01/app/oracle/oradata/orcl/example01.dbf         EXAMPLE
```

For each tablespace:

- Put the tablespace into backup mode:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
```

- Copy the data files for that tablespace to the backup location:

```
$ cp $ORACLE_HOME/oradata/orcl/users*.dbf /u02/backup/datafile
```

- Bring the tablespace out of backup mode:

```
SQL> ALTER TABLESPACE users END BACKUP;
```

ORACLE

Manually Backing Up an ARCHIVELOG Database

If the database is in ARCHIVELOG mode, then you do not necessarily have to shut it down before copying the files. You end up with an inconsistent backup, but the application of redo data recovers it to a consistent state.

Starting Backup Mode: You do have to put each of the data files into backup mode before copying them, though. Do this using the BEGIN BACKUP clause of the ALTER TABLESPACE and ALTER DATABASE commands. Here is the syntax for each:

```
ALTER TABLESPACE <tablespace> BEGIN BACKUP;
ALTER DATABASE BEGIN BACKUP;
```

The ALTER TABLESPACE command affects only those data files that belong to that tablespace. ALTER DATABASE affects all data files in the database.

Ending Backup Mode: It is important to bring the data files out of backup mode. You cannot have any data files in backup mode at the time the database is shut down. If you attempt to shut down the database in that state, you will receive an error. Also, because backup mode causes additional redo to be generated, there is extra load on the system. There is no reason to have any data files in backup mode if you are not actively backing them up.

Note: In addition, you need to archive out the current redo log files, and back them up safely as well.

Backing Up the Control File

Back up the control file:

- As an image copy, to a specifically named file:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO
  2> '/u01/backup/controlfile.bak';

Database altered.
```

- By generating a script that re-creates it, in a trace file:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

Database altered.
```


Backing Up the Control File

You should back up the control file every time you make a structural change to the database. Use one of the commands shown in the slide to do this. The first command creates a binary copy of the file. You can optionally supply the REUSE keyword if the backup file already exists and you want to overwrite it.

The second command makes a plain text version of the control file, which is actually a script that creates the control file when run. The resulting script is written to the diagnostics trace directory, such as:

```
$ORACLE_BASE/diag/rdbms/orcl/orcl/trace
```

You can also specify a name for the trace file by using the `AS 'filename'` clause.

Performing User-Managed Complete Database Recovery: Overview

User-managed complete database recovery:

- Recovers the database to the most recent SCN
- Can be done with the entire database at once, or a data file or tablespace at a time
- Requires a current or backup control file
- Requires backups of all files to be recovered
- Requires all archive logs up to the present

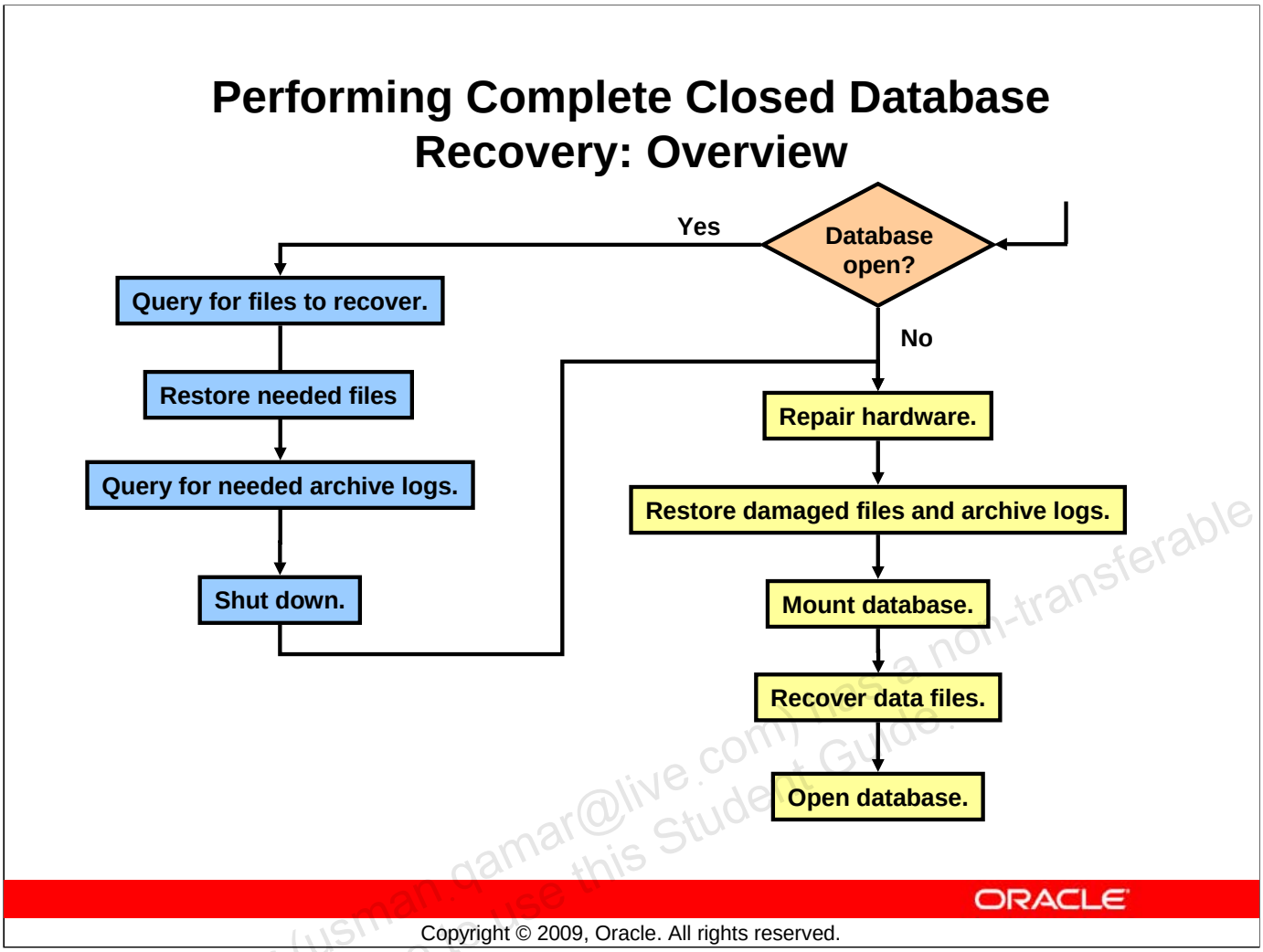
Performing Complete Database Recovery: Overview

Complete database recovery brings the database back to its most current state. You can recover the entire database, or a single tablespace or data file at a time. You must have a current or backup control file in order to perform complete database recovery. You must also have backups available for all files in need of media recovery or you must have all archived redo log files that were generated since the data file was added to the database. Refer to the *Oracle Database Backup and Recovery User's Guide* for additional information about re-creating data files when backups are not available.

You must have all the archive logs available, from the point in time the backups were taken, to the present. If you do not have all of them, you can recover only to the last point in time when redo is available. If no archive logs are required, then only online redo logs are applied.

Query the following views:

- **V$RECOVER_FILE:** To see which files need media recovery
- **V$RECOVERY_LOG:** To see which archive logs are required to perform recovery



Performing Complete Closed Database Recovery: Overview

Under certain circumstances, such as damage to a file belonging to the SYSTEM tablespace, the instance shuts down automatically. Even if the instance keeps running, and there are problems with other data files, you may decide there is no value is keeping the database running; too many database objects are affected. In that case, shut down the database to perform the recovery.

If the database is still open, you can query the V$RECOVER_FILE view to see which data files are in need of recovery, and after you restored them, query V$RECOVERY_LOG to see which archive logs are required. That will tell you which files need to be restored from backup, if any.

Then shut down the database. Look into the media failure to determine the cause of the problem. Repair the problem so that you can restore the files from backup. For example, you may need to replace a disk drive.

Now you can perform the recovery using the RECOVER command. Limit the scope of the recovery to only what is needed, such as data file or tablespace. If needed, recover the entire database. Then open the database.

Identifying Recovery-Related Files

- Identify data files that need to be recovered:

```
SQL> SELECT file#, error FROM v$recover_file;
```

- Identify archive log files that are required to complete recovery:

```
SQL> SELECT archive_name FROM v$recovery_log;
```


Identifying Recovery-Related Files

If the database is still open, query the files as described below. Otherwise, attempt to start the instance and mount the database to issue the queries.

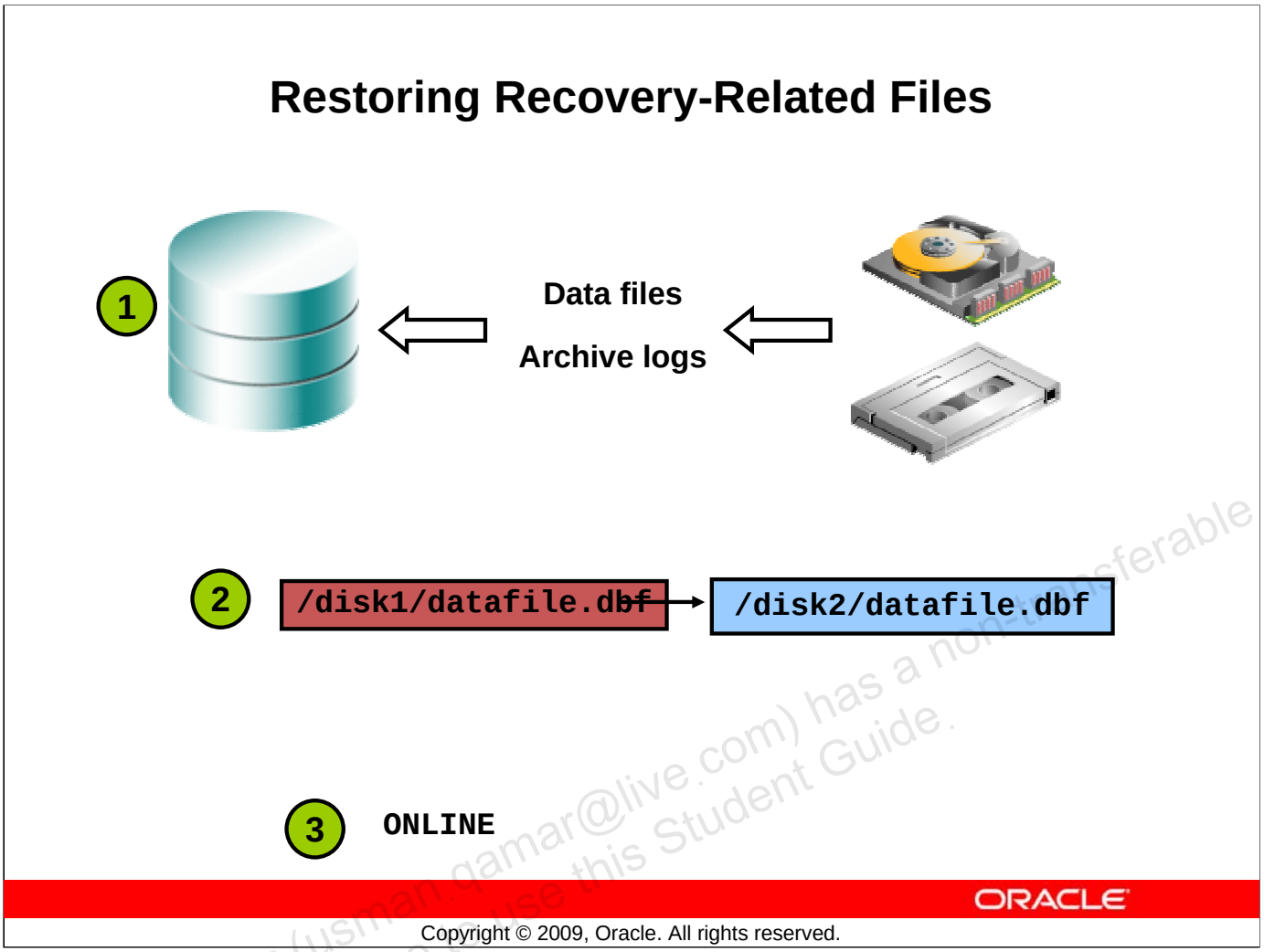
In order to determine which data files require recovery, query the V$RECOVER_FILE view. The ERROR column indicates why the file requires recovery. If this column has any value other than OFFLINE NORMAL, then it needs recovery. To see the whole picture of which data files and tablespaces are affected, join V$DATAFILE and V$TABLESPACE in this query. Here is an example:

```
SELECT r.FILE#, d.NAME df_name, t.NAME tbsp_name,
       d.STATUS, r.ERROR, r.CHANGE#, r.TIME
FROM   V$RECOVER_FILE r, V$DATAFILE d, V$TABLESPACE t
WHERE  t.TS# = d.TS#
AND    d.FILE# = r.FILE#;
```

This tells you the extent of the damage, helping you decide what the objects of the RECOVER command should be.

The V$RECOVERY_LOG view shows which archive log files are needed to perform the recovery. If the list shows files that have since been moved off the default archive log location, then you have to restore them to some location before performing recovery.

After recording the results of these queries, shut down the database.



Restoring Recovery-Related Files

After determining what data files and archive log files are required, restore them to appropriate disk locations. Restore a data file by copying it from the backup location, as in this example:

```
$ cp /disk2/backup/datafile/survey01.dbf \
> $ORACLE_BASE/oradata/ORCL/datafile/survey01.dbf
```

If any archive logs are needed for recovery, check whether they are still in the default disk location for archive logs. They may not be, if they have been moved to tape or another disk drive, for example. If they have been moved, they need to be restored, either to the default archive log location or to a temporary location. If there is enough space available in the default location (which is specified by the `LOG_ARCHIVE_DEST_1` initialization parameter), then restore them there. Otherwise, you can put them on some other disk location. When it is time to restore, you can specify that alternate location to find archive log files.

If you had to move a data file, that fact has to be recorded in the control file. That is done by executing the `ALTER DATABASE RENAME FILE` command, as shown in the following example:

```
SQL> ALTER DATABASE RENAME FILE
  2> '/u01/app/oracle/oradata/ORCL/datafile/survey01.dbf' TO
  3> '/newdisk/ORCL/datafile/survey01.dbf';
```

Note: You must start the instance and mount the database before executing the `ALTER DATABASE RENAME FILE` command.

Restoring Recovery-Related Files (continued)

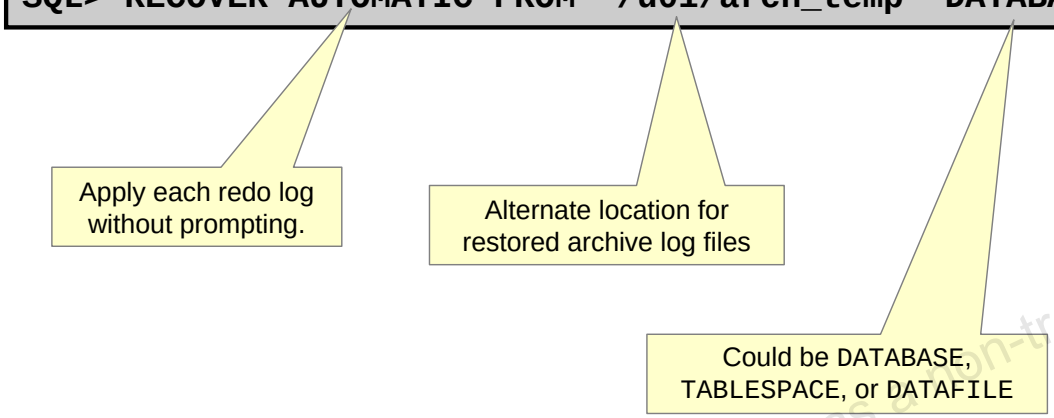
If you have not yet done so, mount the database and bring all the data files online. You can check the status of each data file by querying the V$DATAFILE view. Bring the data files online by using a command like the following:

```
SQL> ALTER DATABASE DATAFILE \
2  > '/newdisk/ORCL/datafile/survey01.dbf' ONLINE;
```

Applying Redo Data

1. Apply redo data using the RECOVER command:

```
SQL> RECOVER AUTOMATIC FROM '/u01/arch_temp' DATABASE;
```



2. Open the database:

```
SQL> ALTER DATABASE OPEN;
```

ORACLE

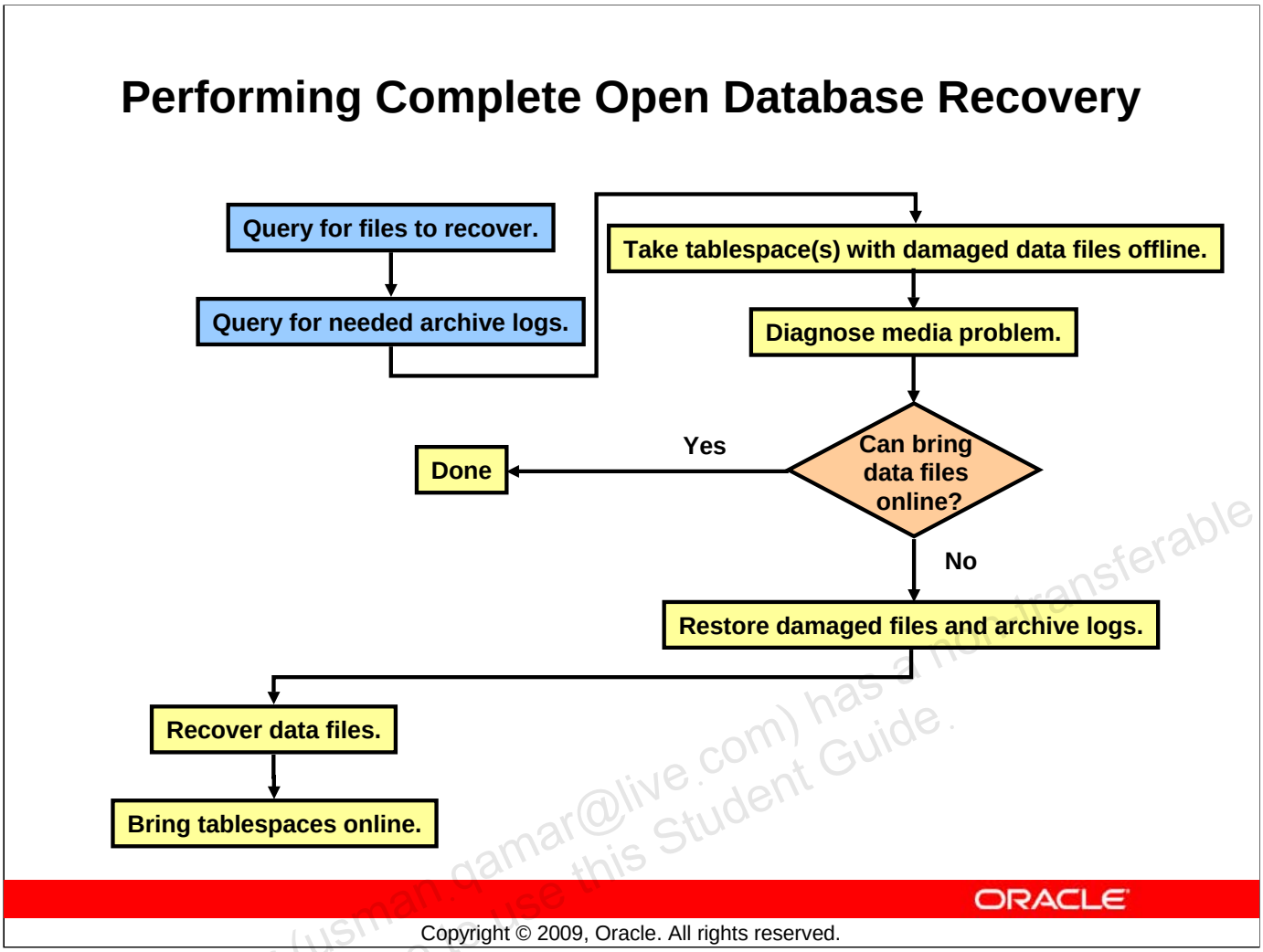
Applying Redo Data

Now the data files have been restored to some point in the past. The archive log files have also been restored: either to their default location or to some other location, for the purpose of this recovery only. You are ready to perform the actual recovery step, which means the redo is applied and the data files are brought up to the latest SCN. Do that using the SQL*Plus RECOVER command.

If you do not specify the AUTOMATIC option, then you are prompted for each redo log file that is about to be applied. That gives you more control over the recovery process. Typically, AUTOMATIC is used for full recovery.

If the archive log files have been restored to some disk location other than the default for the database, then you must specify the FROM clause. Supply the directory where the files are stored, and the recovery process will look there for the files.

Finally, open the database. It is now fully recovered.



Performing Complete Open Database Recovery

If media failure occurs while the database is open, the database continues to function. When an attempt is made to write to one of these data files, the data file is taken offline automatically. A query against one of these data files does not cause it to go offline, but it does result in an error being returned to the user that issued the query.

As with the closed database recovery, you first need to query for the files and archive logs that need to be recovered. Then, take all tablespaces that contain damaged data files offline. Use a command such as the following to do this:

```
SQL> ALTER TABLESPACE survey OFFLINE TEMPORARY;
```

Using the `TEMPORARY` option causes Oracle to perform a checkpoint on any online data files belonging to the tablespace. Checkpointed data files do not require recovery when they are brought back online, because they are up-to-date for the latest SCN of any transactions that would have affected them. This is the more desirable option, although the data files must be available at the time this command is run. It is possible that the problem was temporary, and you are able to bring the tablespaces online with no errors.

Performing Complete Open Database Recovery (continued)

Inspect the media to determine the cause of the problem. You can use the `DBVERIFY` utility for this. If the files are permanently damaged, then proceed to restore and recover as described for the closed database recovery earlier in this lesson. After the restore and recovery steps are complete, bring all the tablespaces online again.

Note: For more information about the `DBVERIFY` utility, see the *Backup and Recovery User's Guide*.

Performing User-Managed Incomplete Recovery: Overview

Recover the database to a past point in time in the following situations:

- You want the database to be in the state that existed before a user error or an administrative error occurred.
- The database contains corrupt blocks after you tried block media recovery.
- You are unable to perform complete database recovery because some of the redo log files are missing.
- You want to create a test database that is in the state at some time in the past.
- One or more unarchived redo log files and a data file are lost.

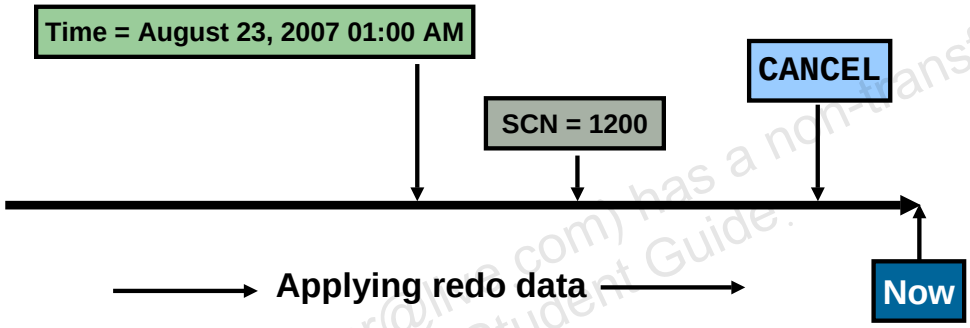
Performing User-Managed Incomplete Recovery: Overview

An incomplete recovery is one that does not bring the database back to the most recent SCN that was transacted. For some reason, as listed in the slide, you need to recover that database only up to a point in the past, not to the present. The processing that occurs when performing incomplete recovery differs from the processing for complete recovery, basically, in the amount of redo that is applied.

Choosing an Incomplete Recovery Method

Indicate when to stop applying redo data by:

- Specifying a time at which to stop
- Specifying an SCN at which to stop
- Issuing a CANCEL command while the recovery is executing



Choosing an Incomplete Recovery Method

As you plan your incomplete recovery, decide which method you want to use for specifying when to stop applying redo data. You stop the recovery process by specifying one of the following:

- **A time:** The time in the logs at which recovery should stop. This can be automated so that the process does not prompt you for each file name.
- **An SCN:** The system change number at which recovery should stop. This can be automated so that the process does not prompt you for each file name.
- **CANCEL:** Specify the `CANCEL` keyword when the recovery process prompts for the next redo log file name. You cannot automate this process because you must specify `CANCEL` to terminate the recovery operation.

Performing User-Managed Incomplete Recovery

- Recover a database until time:

```
SQL> RECOVER DATABASE UNTIL
  2  TIME '2005-12-14:12:10:03';
```

- Recover a database until cancel:

```
SQL> RECOVER DATABASE UNTIL CANCEL;
```


ORACLE

Performing User-Managed Incomplete Recovery

The following command is used to perform incomplete recovery:

```
RECOVER [AUTOMATIC] DATABASE option
```

Following are the meanings of the options:

- **AUTOMATIC:** Automatically applies archived and redo log files
- ***option*:** `UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS'`
  `UNTIL CANCEL`
  `UNTIL CHANGE <integer>`
  `USING BACKUP CONTROLFILE`

Cancel-Based Incomplete Recovery

Cancel-based incomplete recovery is very much like closed database complete recovery. The difference is how you execute the `RECOVER` command; specify the `UNTIL CANCEL` clause. This clause causes the recovery process to prompt you with the suggested name for each redo log file to be applied. So, as the recovery proceeds, you are prompted with an archived or online redo log file name, and, for each one, you can either accept it or change it. When you reach the point where you want the recovery to stop, enter `CANCEL` instead of accepting the file name. This stops the recovery.

After this is done, you have to open the database with the `RESETLOGS` option. The database is in another instantiation now, so the redo log sequence numbers need to be reset.

Performing User-Managed Incomplete Recovery (continued)

After opening the database, check the alert log for messages. This is how you find out if the recovery was successful.

Time- and Change-Based Incomplete Recovery

Both time- and change-based incomplete recovery are like the cancel-based recovery, except that different criteria are used to specify when to stop the recovery. Time-based recovery uses a time specified on the command line of the `RECOVER` command, to know when to stop. Change-based recovery uses an SCN, specified on the command line.

As with all incomplete recoveries, the database must then be opened using the `RESETLOGS` option.

Note: To apply redo log files automatically during recovery, you can use the SQL*Plus `SET AUTORECOVERY ON` command, enter `AUTO` at the recovery prompt, or use the `RECOVER AUTOMATIC` command.

Performing User-Managed Incomplete Recovery: Steps

To perform user-managed incomplete recovery, follow these steps:

1. Shut down the database.
2. Restore data files.
3. Mount the database.
4. Recover the database.
5. Open the database with the RESETLOGS option.

ORACLE

Performing User-Managed Incomplete Recovery: Steps

1. If the database is open, shut it down by using the NORMAL, IMMEDIATE, or TRANSACTIONAL option.
2. Restore all data files from backup. You must use a backup taken before the time you plan to recover to. You may also need to restore archived logs. If there is enough space available, restore to the LOG_ARCHIVE_DEST location or use the ALTER SYSTEM ARCHIVE LOG START TO <*LOCATION*> command or the SET LOGSOURCE <*LOCATION*> command to change the location. If you perform incomplete recovery to a point when the database structure is different than the current, you also need to restore the control file.
3. Mount the database.
4. Recover the database by using the RECOVER DATABASE command.
5. To synchronize data files with control files and redo logs, open the database by using the RESETLOGS option.

User-Managed Time-Based Recovery: Example

This is the scenario:

- A job ran in error, and its effects must be undone.
- This happened 15 minutes ago, and there has been little database activity since then.
- You decide to perform incomplete recovery to restore the database back to its state as of 15 minutes ago.

```
SQL> SHUTDOWN IMMEDIATE
$ cp /BACKUP/*.dbf /u01/db01/ORADATA
SQL> STARTUP MOUNT
SQL> RECOVER DATABASE UNTIL TIME '2005-11-28:11:44:00';
SQL> ALTER DATABASE OPEN RESETLOGS;
```


ORACLE

User-Managed Time-Based Recovery: Example

The following is a typical scenario employing `UNTIL TIME` recovery. Assume the following facts:

- The current time is 12:00 PM on November 28, 2005.
- A job was run incorrectly, and many tables in several schemas were affected.
- This happened at approximately 11:45 AM.
- Database activity is minimal because most staff are currently in a meeting. The state of the database before the job ran must be restored.

Because the approximate time of the error is known and the database structure has not changed since 11:44 AM, you can use the `UNTIL TIME` method:

1. If the database is open, shut it down by using the `NORMAL`, `IMMEDIATE`, or `TRANSACTIONAL` option.
2. Restore all data files from backup (the most recent if possible). You may also need to restore archived logs. If there is enough space available, restore to the `LOG_ARCHIVE_DEST` location or use the `ALTER SYSTEM ARCHIVE LOG START TO <LOCATION>` command or the `SET LOGSOURCE <LOCATION>` command to change the location.
3. Mount the database.

User-Managed Time-Based Recovery: Example (continued)

4. Recover the database:

```
SQL> recover database until time '2005-11-28:11:44:00'
ORA-00279: change 148448 … 11/27/05 17:04:20 needed for thread
...
Media recovery complete.
```

5. To synchronize data files with control files and redo logs, open the database by using the `RESETLOGS` option:

```
SQL> alter database open resetlogs;
SQL> archive log list
...
Oldest online log sequence 0
Next log sequence to archive 1
Current log sequence 1
```

When recovery is successful, notify users that the database is available for use, and any data entered after the recovery time (11:44 AM) will need to be reentered.

User-Managed Cancel-Based Recovery: Example

The scenario is the same as the one for the time-based example, except for these findings:

- Redo logs are not multiplexed.
- One of the online redo logs is missing.
- The missing redo log is not archived.
- The redo log contained information from 11:34 AM.
- Twenty-six minutes of data are lost.
- Users can reenter their data manually.

ORACLE

User-Managed Cancel-Based Recovery: Example

After searching through the directory for the redo log files, you notice that redo log `log2a.rdo` cannot be located and has not been archived. Therefore, you cannot recover past this point.

Querying `V$ARCHIVED_LOG` confirms the absence of archived log sequence 48 (`log2a.rdo`):

```
SQL> SELECT * FROM v$archived_log;
RECID   STAMP      ...  FIRST_CHANGE#    FIRST_TIME
-----   ---------  ...  -------------    ----------
   1    318531466  ...  88330            05-11-15:12:43
  47    319512880  ...  309067           05-11-28:11:26
```

User-Managed Cancel-Based Recovery: Example

Recover the database as follows:

- Shut down the database.
- Restore all data files from the most recent backup.
- Mount the database.
- Execute `RECOVER DATABASE UNTIL CANCEL`.
- Execute `ALTER DATABASE OPEN RESETLOGS` to open the database.

User-Managed Cancel-Based Recovery: Example (continued)

The steps for cancel-based recovery are the same as for time-based recovery, except for the `RECOVER DATABASE` step. When the `RECOVER DATABASE UNTIL CANCEL` command is executed, it recovers the database until it cannot find a log file. When you are prompted for the name of the missing archived redo log file, enter `CANCEL`; the recovery stops at that point in time.

Summary

In this lesson, you should have learned how to:

- Describe the difference between user-managed and server-managed backup and recovery
- Perform user-managed complete database recovery
- Perform user-managed incomplete database recovery