6

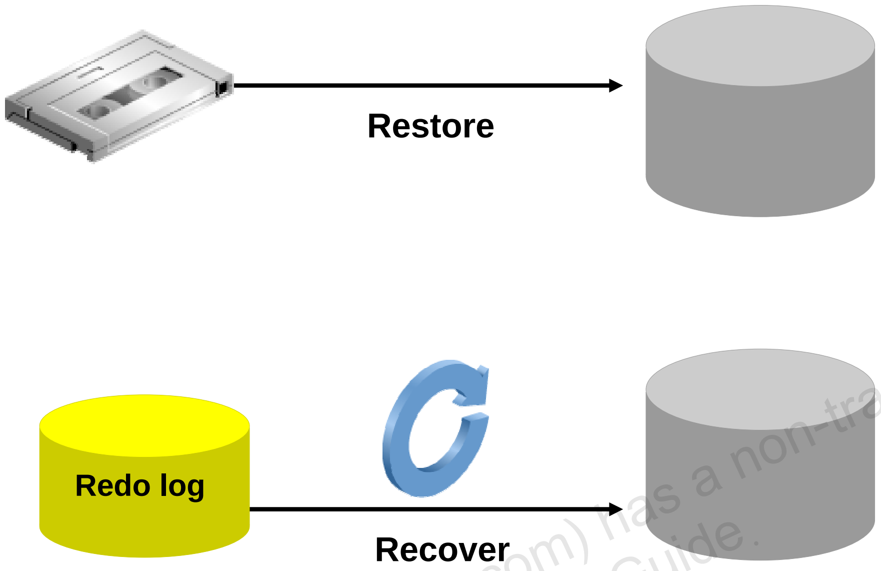
Restore and Recovery Tasks

Objectives

After completing this lesson, you should be able to:

- Describe the causes of file loss and determine the appropriate action
- Describe major recovery operations
- Back up and recover a control file
- Recover from a lost redo log group

Restoring and Recovering



Restoring and Recovering

The “recovery” portion of backup and recovery tasks includes two major types of activities: restoring and recovering. *Restoring* a file is the process of copying a backup into place to be used by the database. This is necessary if, for example, a file is damaged because the physical disk it is on fails. This is usually due to hardware problems, such as disk write errors, or controller failure. In that case, a backup of the file needs to be copied onto a new (or repaired) disk.

Recovering the file entails applying redo such that the state of the file is brought forward in time, to whatever point you want. That point is usually as close to the time of failure as possible.

In the database industry, these two operations are often referred to, collectively, with the single term “recovery.”

Causes of File Loss

File loss can be caused by:

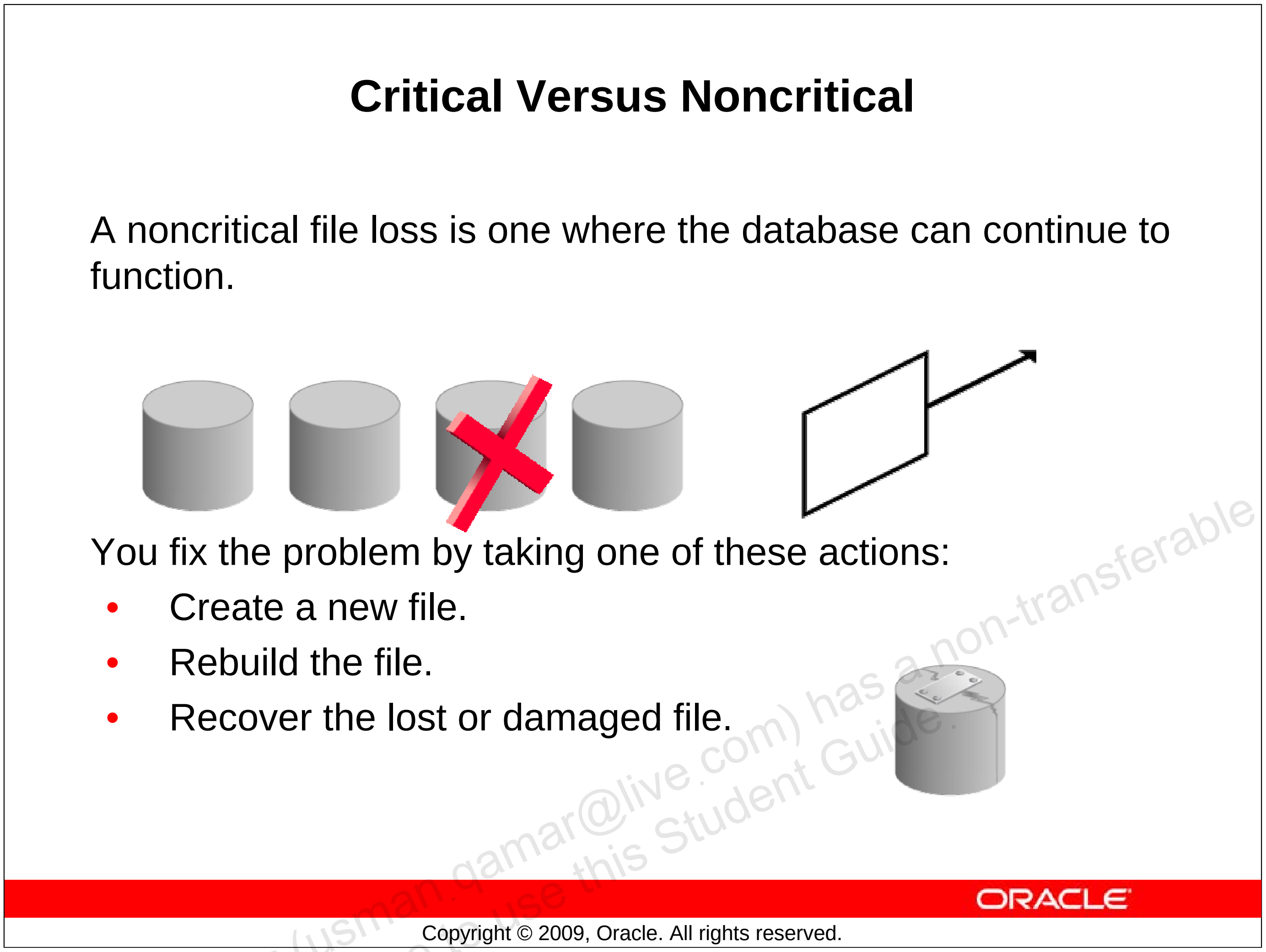
- User error
- Application error
- Media failure

ORACLE

Causes of File Loss

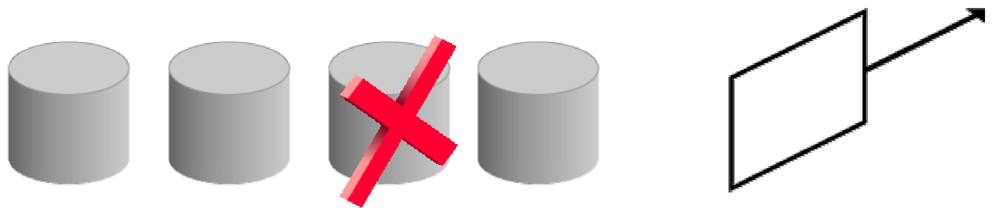
Files can be lost or damaged due to:

- **User error:** An administrator may inadvertently delete or copy over a necessary operating system file.
- **Application error:** An application or script can also have a logic error in it, as it processes database files, resulting in a lost or damaged file.
- **Media failure:** A disk drive or controller may fail fully or partially, and introduce corruption into files, or even cause a total loss of files.



Critical Versus Noncritical

A noncritical file loss is one where the database can continue to function.



You fix the problem by taking one of these actions:

- Create a new file.
- Rebuild the file.
- Recover the lost or damaged file.

Critical Versus Noncritical

A noncritical file is one that the database and most applications can operate without. For example, if the database loses one multiplexed redo log file, there are still other redo log file copies that can be used to keep the database operating.

Although the loss of a noncritical file does not cause the database to crash, it can impair the functioning of the database. For example:

- The loss of an index tablespace can cause applications and queries to run much slower, or even make the application unusable, if the indexes were used to enforce constraints.
- The loss of an online redo log group, as long as it is not the current online log group, can cause database operations to be suspended (when LGWR next tries to write to the group) until new log files are generated.
- The loss of a temporary tablespace can prevent users from running queries or creating indexes until they have been assigned to a new temporary tablespace.

Automatic Tempfile Recovery

SQL statements that require temporary space to execute may fail if one of the tempfiles is missing.

```
SQL> select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13;
select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13
              *
ERROR at line 1:
ORA-01565: error in identifying file
'/u01/app/oracle/oradata/orcl/temp01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

Good news:

- Automatic re-creation of temporary files at startup
- (Manual re-creation also possible)

Automatic Tempfile Recovery

If a temporary file (tempfile) belonging to the temporary tablespace is lost or damaged, the extents in that file will not be available. This problem may manifest itself as an error during the execution of SQL statements that require temporary space for sorting.

The SQL statement shown in the slide has a long list of columns to order by, which results in the need for temporary space. The missing file error is encountered when this statement requiring a sort is executed.

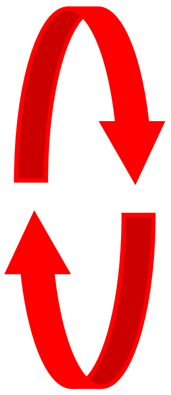
The Oracle database instance can start up with a missing temporary file. If any of the temporary files do not exist when the database instance is started, they are created automatically and the database opens normally. When this happens, a message like the following appears in the alert log during startup:

```
Re-creating tempfile /u01/app/oracle/oradata/orcl/temp01.dbf
```

In the unlikely case that you decide a manual recreation serves you better, use the following commands:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
'/u01/app/oracle/oradata/orcl/temp02.dbf' SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
'/u01/app/oracle/oradata/orcl/temp01.dbf';
```

Log Group Status: Review



A redo log group has a status of one of the following values at any given time:

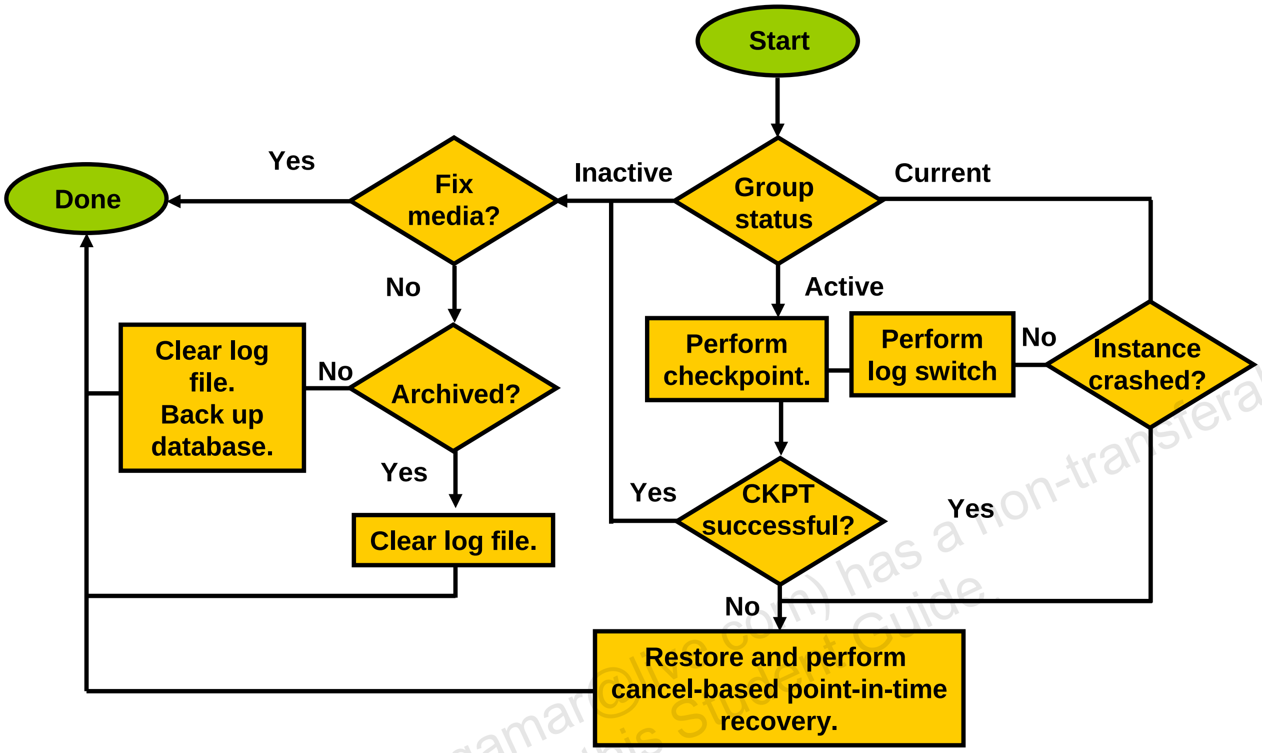
- **`CURRENT:`** The LGWR process is currently writing redo data to it.
- **`ACTIVE:`** It is no longer being written to, but it is still required for instance recovery.
- **`INACTIVE:`** It is no longer being written to, and it is no longer required for instance recovery.

Log Group Status: Review

To deal with the loss of redo log files, it is important to understand the possible states of redo log groups. Redo log groups cycle through three different states as part of the normal running of the Oracle database. They are, in order of the cycle:

- **`CURRENT:`** This state means that the redo log group is being written to by LGWR to record redo data for any transactions going on in the database. The log group remains in this state until there is a switch to another log group.
- **`ACTIVE:`** The redo log group still contains redo data that is required for instance recovery. This is the status during the time when a checkpoint has not yet executed that would write out to the data files all data changes that are represented in the redo log group.
- **`INACTIVE:`** The checkpoint discussed above has indeed executed, meaning that the redo log group is no longer needed for instance recovery, and is free to become the next `CURRENT` log group.

Recovering from the Loss of a Redo Log Group



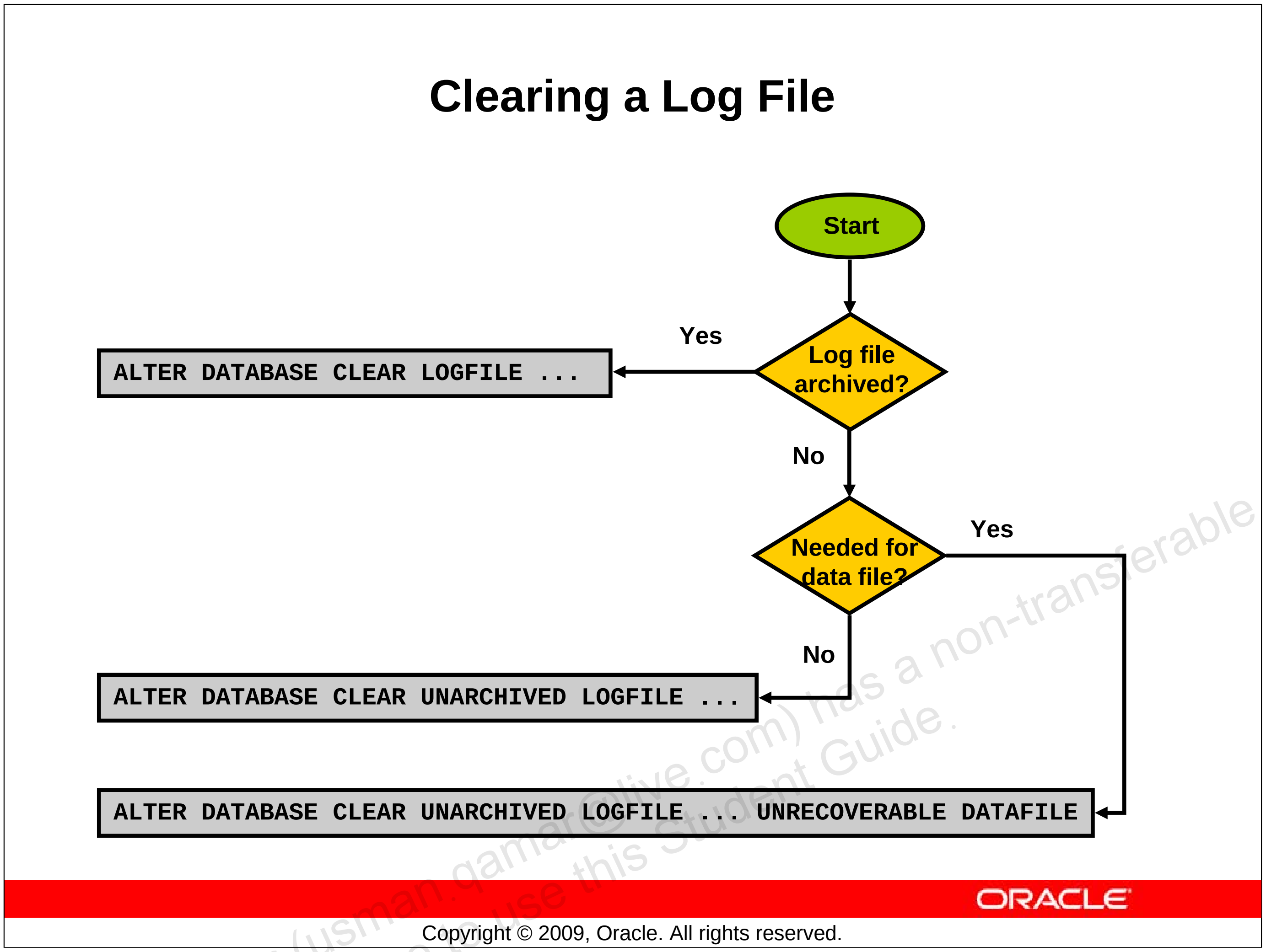
Recovering from the Loss of a Redo Log Group

If you have lost an entire redo log group, then all copies of the log files for that group are unusable or gone.

The simplest case is where the redo log group is in the `INACTIVE` state. That means it is not currently being written to, and it is no longer needed for instance recovery. If the problem is temporary, or you are able to fix the media, then the database continues to run normally, and the group is reused when enough log switch events occur. Otherwise, if the media cannot be fixed, you can clear the log file. When you clear a log file, you are indicating that it can be reused.

If the redo log group in question is `ACTIVE`, then, even though it is not currently being written to, it is still needed for instance recovery. If you are able to perform a checkpoint, then the log file group is no longer needed for instance recovery, and you can proceed as if the group were in the inactive state.

If the log group is in the `CURRENT` state, then it is, or was, being actively written to at the time of the loss. You may even see the LGWR process fail in this case. If this happens, the instance crashes. Your only option at this point is to restore from backup, perform cancel-based point-in-time recovery, and then open the database with the `RESETLOGS` option.



Clearing a Log File

Start
Log file archived?
Yes
ALTER DATABASE CLEAR LOGFILE ...
No
Needed for data file?
Yes
No
ALTER DATABASE CLEAR UNARCHIVED LOGFILE ...
ALTER DATABASE CLEAR UNARCHIVED LOGFILE ... UNRECOVERABLE DATAFILE
ORACLE

Clearing a Log File

Clear a log file using this command:

```
ALTER DATABASE CLEAR [UNARCHIVED] LOGFILE GROUP <n>
      [UNRECOVERABLE DATAFILE]
```

When you clear a log file, you are indicating that it can be reused. If the log file has already been archived, the simplest form of the command can be used. Use the following query to determine which log groups have been archived:

```
SQL> SELECT GROUP#, STATUS, ARCHIVED FROM V$LOG;
```

For example, the following command clears redo log group 3, which has already been archived:

```
SQL> ALTER DATABASE CLEAR LOFGILE GROUP 3;
```

If the redo log group has not been archived, then you must specify the `UNARCHIVED` keyword. This forces you to acknowledge that it is possible that there are backups that rely on that redo log for recovery, and you have decided to forgo that recovery opportunity. This may be satisfactory for you, especially if you take another backup right after you correct the redo log group problem; you then no longer need that redo log file.

It is possible that the redo log is required to recover a data file that is currently offline.

Recovering from a Lost Index Tablespace

- A tablespace that contains only indexes may be recovered without performing a RECOVER task.
- If a data file that belongs to an index-only tablespace is lost, it may be simpler to re-create the tablespace and re-create the indexes.

Recovering from a Lost Index Tablespace

Indexes are computed objects, in that they do not provide any original data, and they are only a different representation of data that already exists. So, in most cases, indexes can be re-created easily. If you have a tablespace that contains only indexes, recovering from a loss of a data file belonging to that tablespace can be simplified.

When a data file like this is lost, you can perform the following steps:

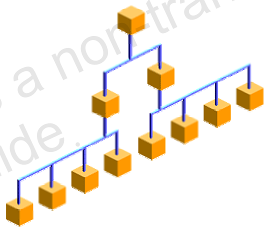
1. Drop the data file.
2. Drop the tablespace.
3. Re-create the index tablespace.
4. Re-create the indexes that were in the tablespace.

Re-Creating Indexes

Use options to reduce the time it takes to re-create the index:

- `PARALLEL`
- `NOLOGGING`

```
SQL> CREATE INDEX rname_idx
  2  ON hr.regions (region_name)
  3  PARALLEL 4;
```



Re-Creating Indexes

When creating or re-creating an index, you can use the following keywords to reduce the creation time:

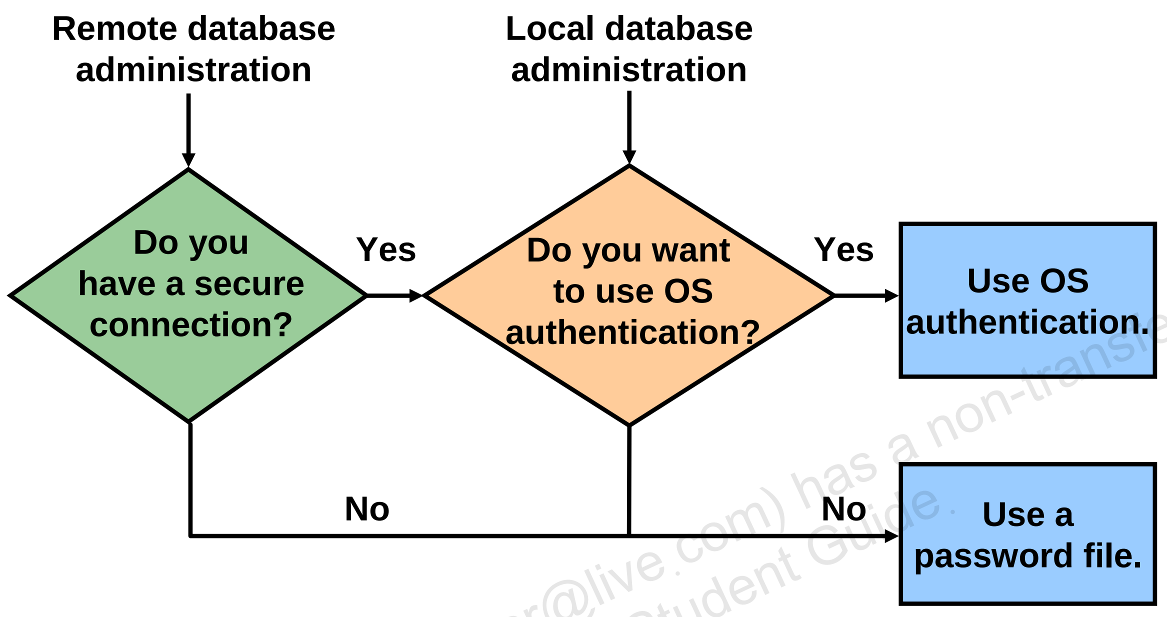
- **`PARALLEL` (`NOPARALLEL`** is the default**):** Multiple processes can work together simultaneously to create an index. By dividing the work necessary to create an index among multiple server processes, the Oracle server can create the index more quickly than if a single server process created the index sequentially. The table is randomly sampled and a set of index keys is found that equally divides the index into the same number of pieces as the specified degree of parallelism. A first set of query processes scans the table, extracts the key-and-row ID pairs, and sends each pair to a process in a second set of query processes based on the key. Each process in the second set sorts the keys and builds an index in the usual fashion. After all index pieces are built, the parallel coordinator concatenates the pieces (which are ordered) to form the final index.
- **`NOLOGGING`:** Using this keyword makes index creation faster because it creates a very minimal amount of redo log entries as a result of the creation process. This greatly minimized redo generation also applies to direct path inserts and Direct Loader (SQL*Loader) inserts. This is a permanent attribute and thus appears in the data dictionary. It can be updated with the `ALTER INDEX NOLOGGING/LOGGING` command at any time.

  Note: `NOLOGGING` can be overridden, if you are using Data Guard or `FORCE LOGGING` at the database or tablespace level.

Re-Creating Indexes (continued)

When an index is lost, it may be faster and simpler just to re-create it rather than attempt to recover it. You can use Data Pump Export with the `CONTENT=METADATA_ONLY` parameter to create a dump file containing the SQL commands to re-create the index. You can also use Data Pump Import with the `SQLFILE=<filename>` parameter on a previously created dump file. The Data Pump Export and Import utilities are covered in detail in the *Oracle Database 11g: Administration Workshop I* course. Additional information can be found in *Oracle Database Utilities*.

Authentication Methods for Database Administrators


Remote database administration
Local database administration
Do you have a secure connection?
Yes
Do you want to use OS authentication?
Yes
Use OS authentication.
No
No
Use a password file.

Authentication Methods for Database Administrators

Depending on whether you want to administer your database locally on the same machine on which the database resides or to administer many different database servers from a single remote client, you can choose either operating system or password file authentication to authenticate database administrators:

- If the database has a password file and you have been granted the SYSDBA or SYSOPER system privilege, then you can be authenticated by a password file.
- If the server is not using a password file, or if you have not been granted SYSDBA or SYSOPER privileges and are, therefore, not in the password file, you can use operating system authentication. On most operating systems, authentication for database administrators involves placing the operating system username of the database administrator in a special group, generically referred to as OSDBA. Users in that group are granted SYSDBA privileges. A similar group, OSOPER, is used to grant SYSOPER privileges to users.

Operating system authentication takes precedence over password file authentication. Specifically, if you are a member of the OSDBA or OSOPER group for the operating system, and you connect as SYSDBA or SYSOPER, you will be connected with associated administrative privileges *regardless of the username/password that you specify*.

Re-creating a Password Authentication File

```
SQL> grant sysdba to admin2;
grant sysdba to admin2
*
ERROR at line 1:
ORA-01994: GRANT failed: password file missing or disabled
```

To recover from the loss of a password file:

1. Re-create the password file by using `orapwd`.

```
$ orapwd file=$ORACLE_HOME/dbs/orapworcl password=ora entries=5
```

2. Add users to the password file and assign appropriate privileges to each user.

Re-creating a Password Authentication File

The Oracle database provides a password utility, `orapwd`, to create a password file. When you connect using the `SYSDBA` privilege, you are connecting as the `SYS` schema and not the schema associated with your username. For `SYSOPER`, you are connected to the `PUBLIC` schema. Access to the database using the password file is provided by `GRANT` commands issued by privileged users.

Typically, the password file is not included in backups because, in almost all situations, it can be easily re-created.

It is critically important to the security of your system that you protect your password file and the environment variables that identify the location of the password file. Any user with access to these could potentially compromise the security of the connection.

You should not remove or modify the password file if you have a database or instance mounted using `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE` or `SHARED`. If you do, you will be unable to reconnect remotely using the password file.

Note: Passwords are case-sensitive, so you must take that into consideration when re-creating the password file. Also if the original password file was created with the IGNORECASE=Y option, then it must be recreated with the same option.

Re-creating a Password Authentication File (continued)

Using a Password File

The following are the steps for re-creating the password file:

1. Create the password file by using the password utility `orapwd`.

```
       orapwd file=filename password=password entries=max_users
```

where:

- **`filename`** is the name of the password file (mandatory).
- **`password`** is the password for `SYS` (optional). You are prompted for the password if you do not include the `password` argument.
- **`Entries`** is the maximum number of distinct users allowed to connect as `SYSDBA` or `SYSOPER`. If you exceed this number, you must create a new password file. It is safer to have a larger number. There are no spaces around the "equal to" (=) character.

Example:
```
orapwd file=$ORACLE_HOME/dbs/orapwU15
  password=admin entries=5
```

2. Connect to the database by using the password file created in step 1, and grant privileges as needed.

```
       SQL> CONNECT sys/admin AS SYSDBA
       SQL> grant sysdba to admin2;
```

Password File Locations

UNIX: `$ORACLE_HOME/dbs`

Windows: `%ORACLE_HOME%\database`

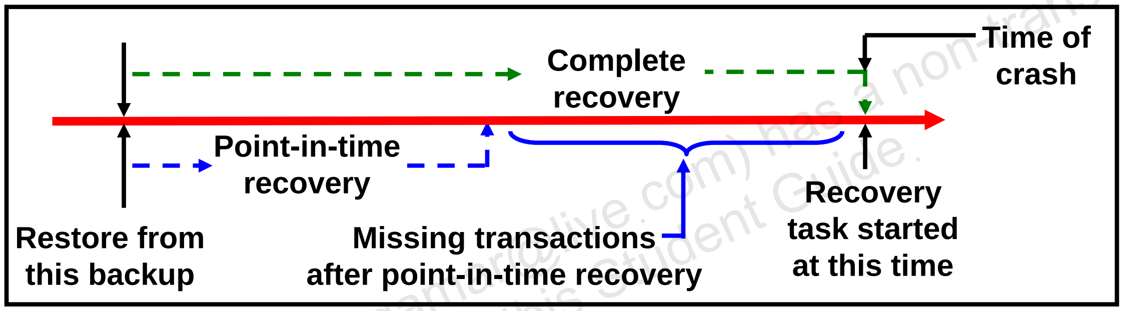
Maintaining the Password File

Delete the existing password file by using operating system commands, and create a new password file by using the password utility.

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- Complete recovery: Brings the database up to the present, including all committed data changes made to the point in time when the recovery was requested
- Incomplete or point-in-time recovery: Brings the database up to a specified point in time in the past, before the recovery operation was requested


Complete recovery
Time of crash
Point-in-time recovery
Recovery task started at this time
Restore from this backup
Missing transactions after point-in-time recovery

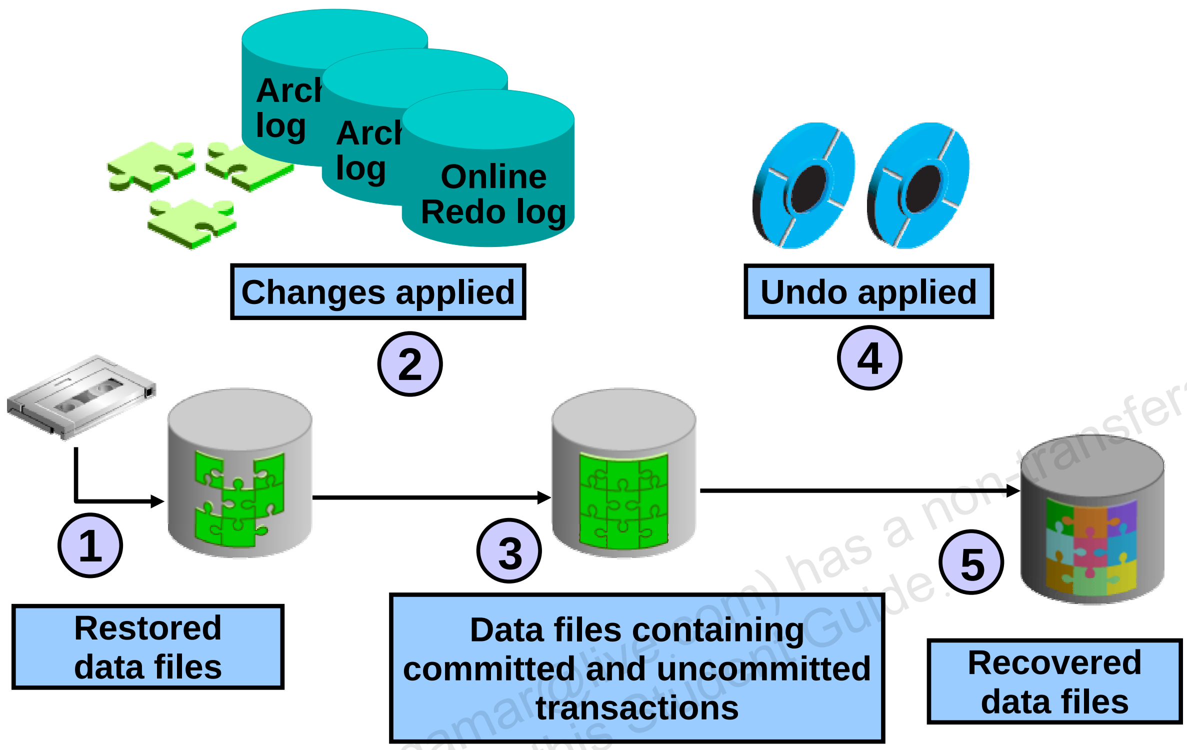
ORACLE

Comparing Complete and Incomplete Recovery

When you perform complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the present time.

Incomplete recovery, however, brings the database to some point in the past point-in-time. It is also known as “Database Point in Time Recovery ”. This means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

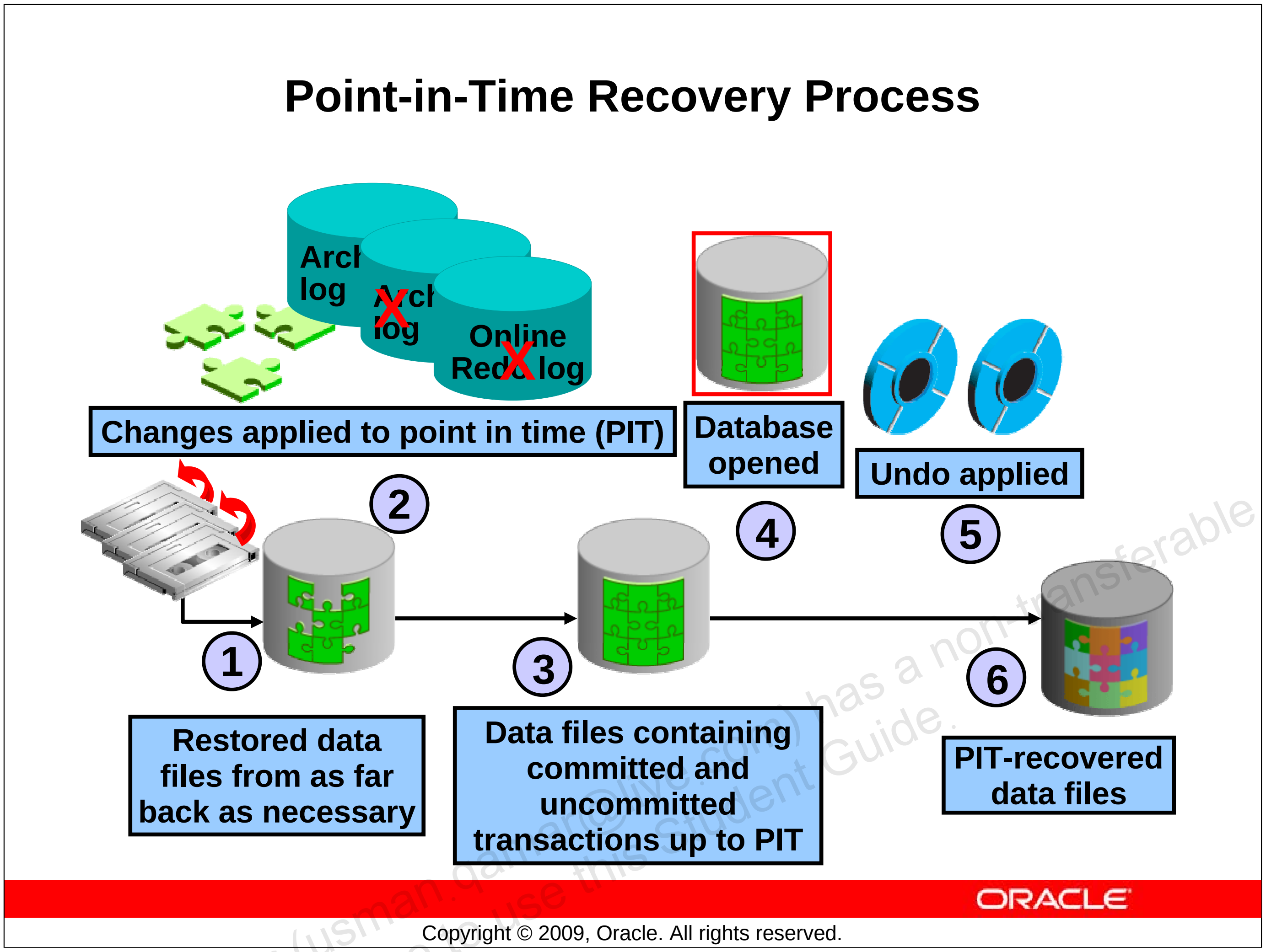
Complete Recovery Process



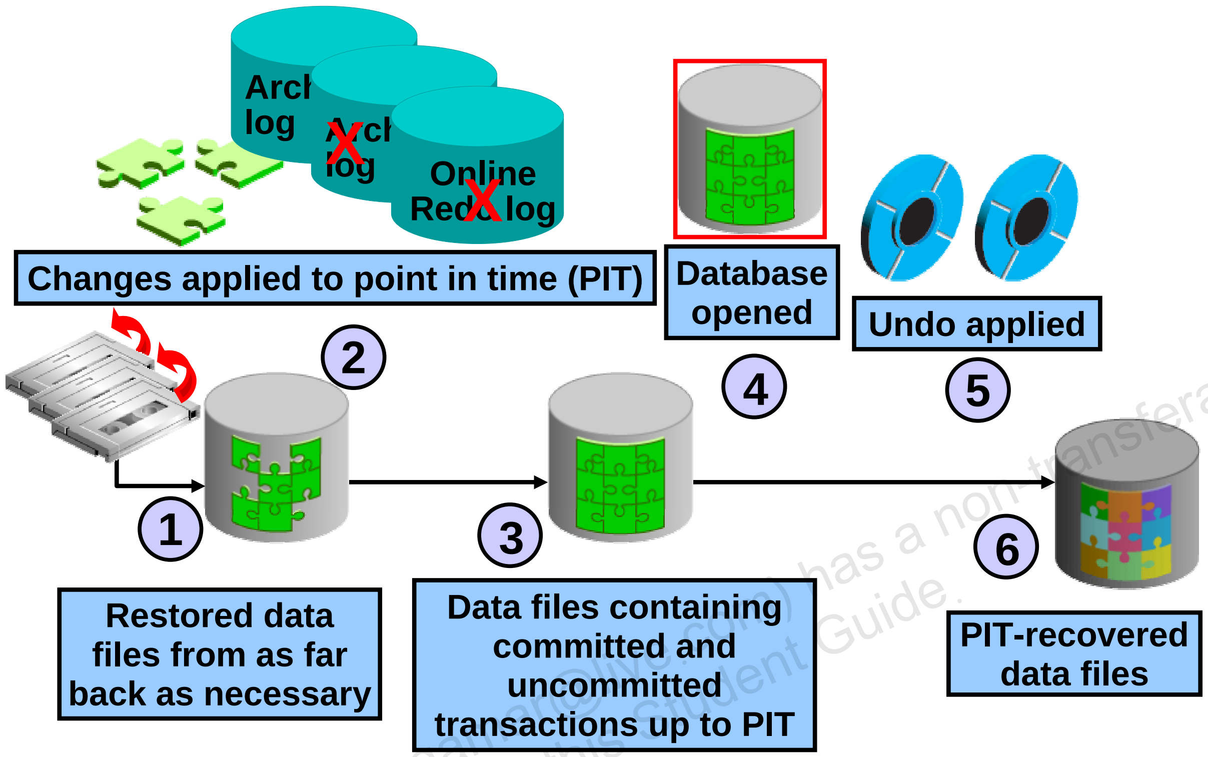
Complete Recovery Process

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent transactions have been reentered. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.



Point-in-Time Recovery Process


Arch log
Arch log
Online Redo log
Changes applied to point in time (PIT)
Database opened
Undo applied
2
4
5
1
3
6
Restored data files from as far back as necessary
Data files containing committed and uncommitted transactions up to PIT
PIT-recovered data files

ORACLE

Point-in-Time Recovery Process

Incomplete recovery, or database point-in-time recovery, uses a backup to produce a noncurrent version of the database. That is, you do not apply all of the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform Point-in-Time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The progression taken to perform an point-in-time recovery is listed below:

1. **Restore the data files from backup:** The backup that is used may not be the most recent one, if your restore point destination is to be not very recent. This entails either copying files using OS commands or using the RMAN `RESTORE` command.
2. **Use the `RECOVER` command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
3. **State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
4. **Use the `ALTER DATABASE OPEN` command:** The database is opened before undo is applied. This is to provide higher availability.

Point-in-Time Recovery Process (continued)

5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Point-in-time recovery is the only option if you must perform a recovery and discover that you are missing an archived log containing transactions that occurred sometime between the time of the backup you are restoring from and the target recovery SCN. Without the missing log, you have no record of the updates to your data files during that period. Your only choice is to recover the database from the point in time of the restored backup, as far as the unbroken series of archived logs permits, and then open the database with the `RESETLOGS` option. All changes in or after the missing redo log file are lost.

Recovering a Read-Only Tablespace

Special user-managed backup and recovery considerations for a read-only tablespace:

- You do not have to put it in backup mode in order to make a copy of its data files.
- You do not have to take the tablespace or data file offline before making a copy of it.

Recovering a Read-Only Tablespace

Because read-only tablespaces are not being written to, there are special considerations to take into account, which can make the recovery process faster and more efficient. You do not have to put a read-only tablespace into backup mode or take it offline before copying it to the backup location. Simply copy it.

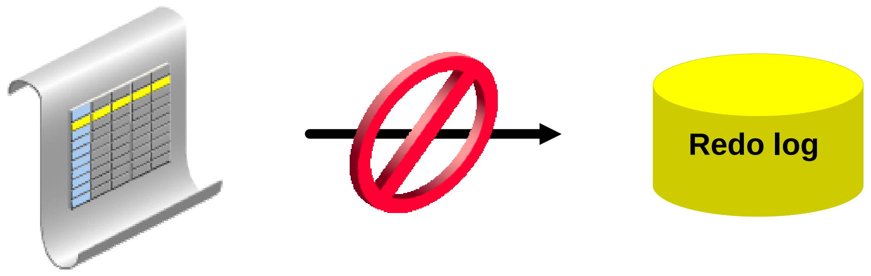
When restoring a read-only tablespace, take the tablespace offline, restore the data files belonging to the tablespace, and then bring the tablespace back online.

Consider the following scenario, where a read-only tablespace is changed to be read/write:

1. Make a backup of a read-only tablespace.
2. Make the tablespace read-write.
3. Recover the tablespace.

The backup you made in step 1 can still be used to recover this tablespace, even though, since the backup was made, the tablespace was made read-write, and has even possibly been written to. In this case, the tablespace requires recovery, after the files are stored from such a backup.

Recovering NOLOGGING Database Objects



```
SQL> CREATE TABLE sales_copy NOLOGGING;
SQL> INSERT /*+ APPEND */ INTO sales_copy
  2  SELECT * FROM sales_history;
```

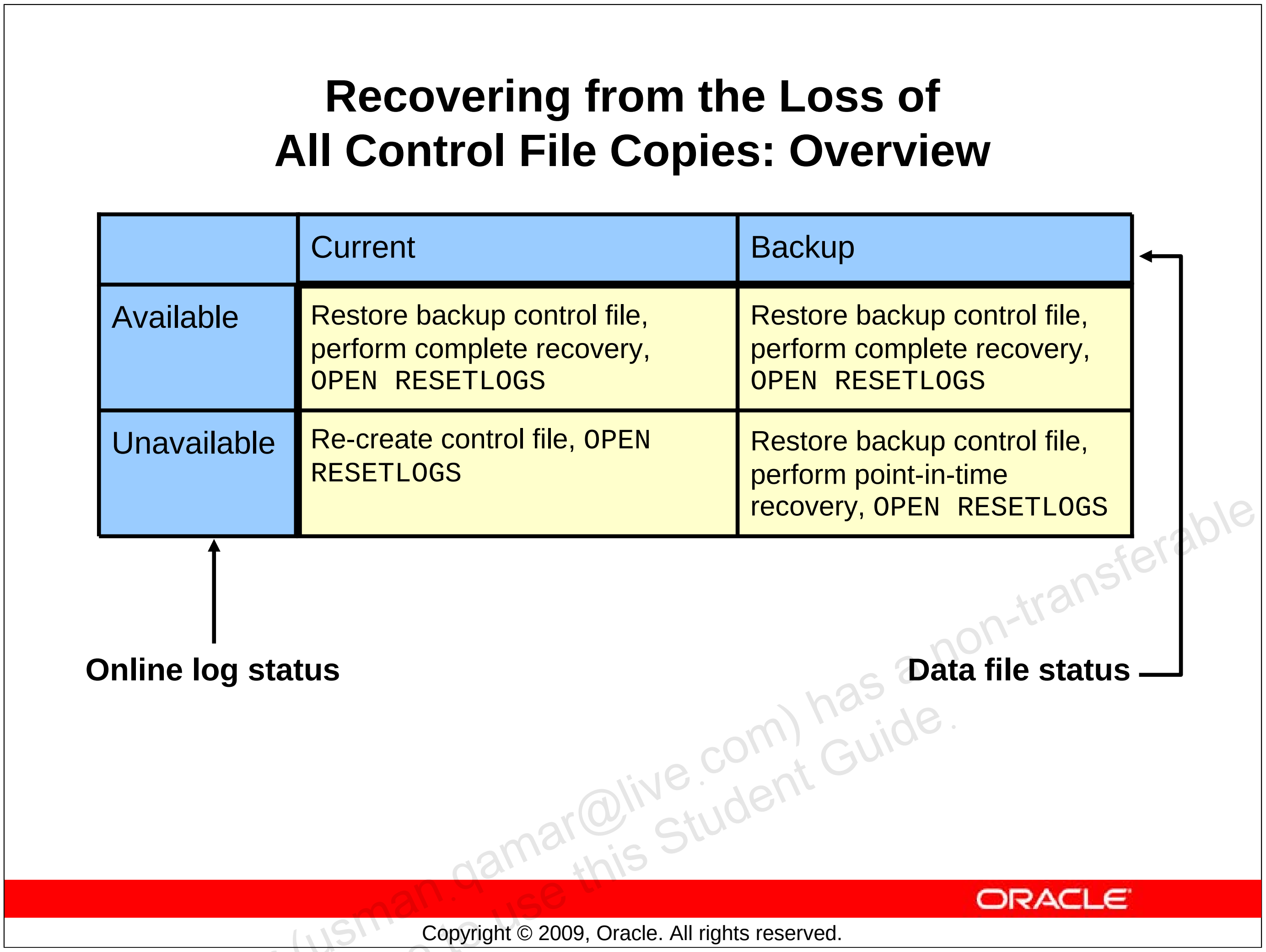
Recovering NOLOGGING Database Objects

Take advantage of the efficiencies of the NOLOGGING attribute of tables and indexes if you can. When you create a table as NOLOGGING, minimal redo data is written to the redo stream to support the creation of the object. This is useful for making large inserts go faster.

In the example in the slide, the SALES_COPY table is created as a NOLOGGING table. As a result, when an insert is done with the APPEND hint, no redo is generated for that particular insert statement. As a result, you cannot recover this transaction on the SALES_HISTORY table. If that is a problem, it is important that you make a backup of whatever tables you populate in this way, right afterward. Then you are able to go to the more recent backup of the table.

If you perform media recovery, and there are NOLOGGING objects involved, they will be marked logically corrupt during the recovery process. In this case, drop the NOLOGGING objects and re-create them.

Use the REPORT UNRECOVERABLE RMAN command to list the names of any tablespaces that contain one or more objects for which a NOLOGGING operation has been performed since the most recent backup of that tablespace.



Recovering from the Loss of All Control File Copies: Overview

	Current	Backup
Available	Restore backup control file, perform complete recovery, `OPEN RESETLOGS`	Restore backup control file, perform complete recovery, `OPEN RESETLOGS`
Unavailable	Re-create control file, `OPEN RESETLOGS`	Restore backup control file, perform point-in-time recovery, `OPEN RESETLOGS`

Online log status
Data file status

Recovering from the Loss of All Control File Copies: Overview

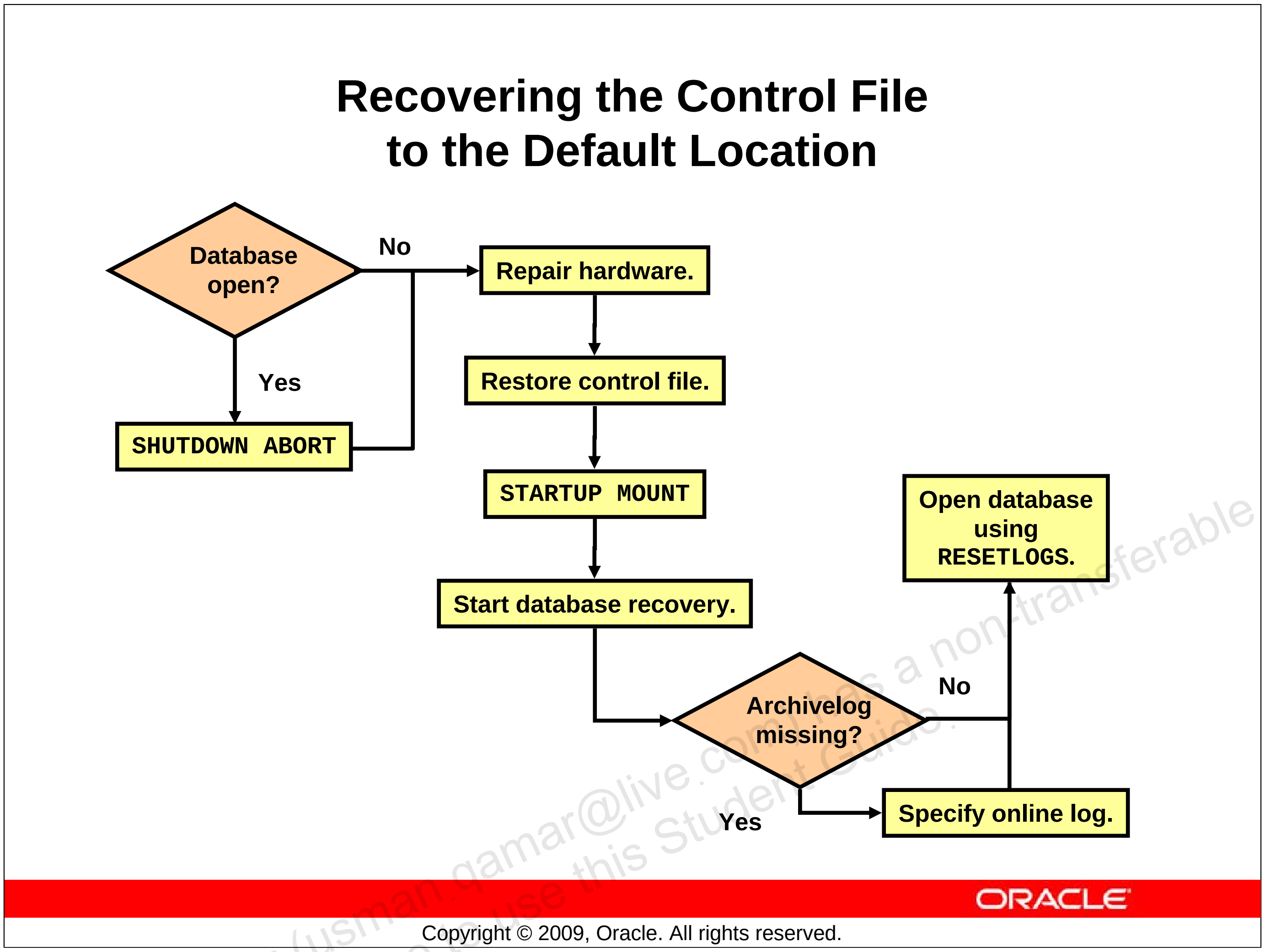
Loss of all control files should never happen. **Prevention is better than recovery**. Even though you have copies of the control file stored in different locations, there is still the possibility that you will, at some point, have to recover from losing all those copies. If you have lost all copies of the current control file, and have a backup control file, your course of action depends on the status of the online log files and the data files. The chart in the slide shows what to do in each of the situations shown.

Online Logs Available

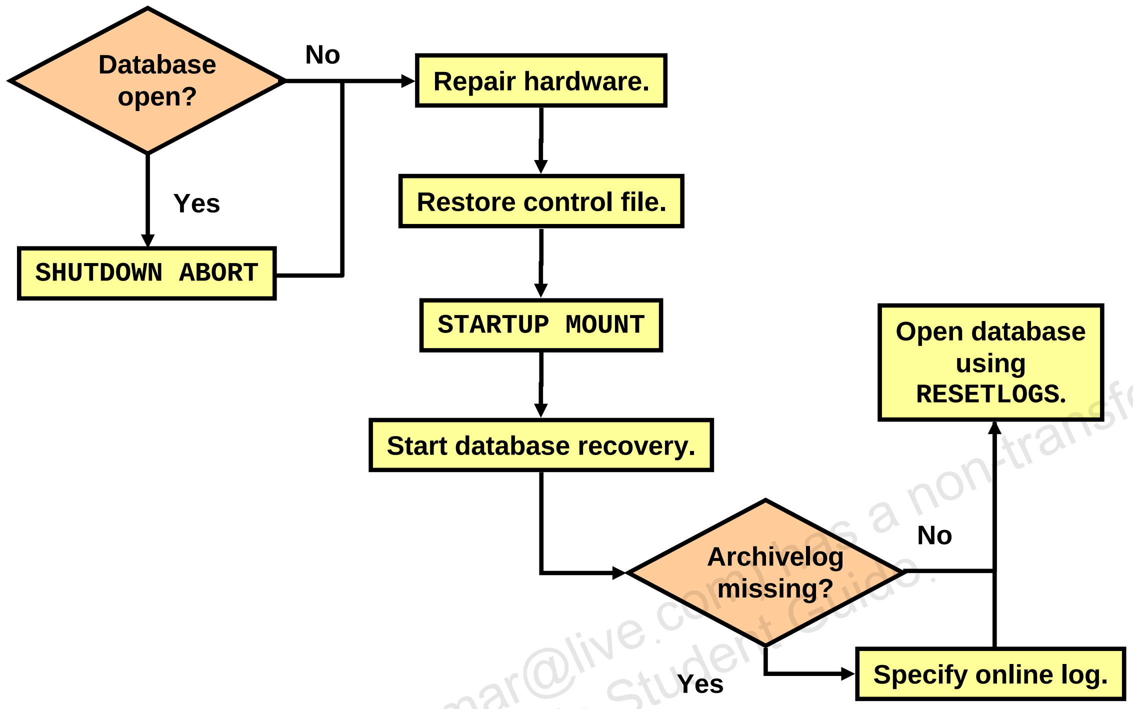
If the online logs are available and contain redo necessary for recovery, and the data files are current, then you can restore a backup control file, perform complete recovery, and open the database with the `RESETLOGS` option. You must specify the file names of the online redo logs during recovery. If the data files are not current, perform the same procedure.

Online Logs Not Available

If the online logs are not available, and the data files are current, then re-create the control file and open `RESETLOGS`. However, if the data files are not current, restore a backup control file, perform point-in-time recovery, and open `RESETLOGS`.



Recovering the Control File to the Default Location



Recovering the Control File to the Default Location

If you need to recover the control file, and the default location is still a valid one, follow the steps shown in the slide. The database must be shut down first. Then repair any hardware, so that the default location is able to remain as a valid one. Restore the control file to the default location. Do this using a command such as this, which copies the backup control file to the default location:

```
% cp /backup/control01.dbf /disk1/oradata/trgt/control01.dbf
% cp /backup/control02.dbf /disk2/oradata/trgt/control02.dbf
```

Mount the database, and start the recovery process. You must specify that a backup control file is being used.

```
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL;
```

If, during the recovery process, you are prompted for a missing redo log, it probably means that the missing redo log is an online redo log file. When prompted, supply the name of the online redo log file. After the recovery completes, open the database, specifying the `RESETLOGS` option.

(*More on this topic in the next lesson.*)

Quiz

In which of the following cases may the RMAN `RECOVER` command be issued?

1. The database is in `NOARCHIVELOG` mode using full backups.
2. The database is in `ARCHIVELOG` mode using full backups.
3. The database is in `NOARCHIVELOG` mode using incremental backups.
4. The database is in `ARCHIVELOG` mode using incremental backups.

Answer: 2, 3, 4

Quiz

Your password file is lost, From where can you, as DBA, recover the entries, so that you can re-create your lost password file?

1. Only from the RMAN catalog
2. From the control file
3. From the Enterprise Manager repository
4. From the data dictionary
5. You must manually regrant the `SYSOPER`, `SYSDBA`, and `SYSASM` entries.

Answer: 5

Summary

In this lesson, you should have learned how to:

- Describe the causes of file loss and determine the appropriate action
- Describe major recovery operations
- Back up and recover a control file
- Recover from a lost redo log group