Chapitre 04

Using Backup and Recovery

▼ Understanding and Configuring Recovery Components

As a database administrator, your main objective is to ensure the database is continuously available for users. This includes:

- **Proactively addressing common failure causes**
- **Increasing the Mean Time Between Failures (MTBF)**
- **Ensuring hardware redundancy**
- **Utilizing Oracle options like Real Application Clusters (RAC), Oracle Streams, and Oracle Data Guard**

To reduce the Mean Time to Recover (MTTR), configure appropriate Oracle initialization parameters and ensure backups are readily available. Minimizing transaction loss involves using redo application, replication, and Oracle Data Guard.

Key Database Structures for Backup and Recovery

- **Control Files**: Maintain lists of database files and records of recent backups.
- **Checkpoint (CKPT) and Database Writer (DBWn) Processes**: Manage instance recovery times.
- **Redo Log Files**: Synchronize data files during instance recovery.
- **Archived Redo Log Files**: Used in media failure recovery to ensure no committed transactions are lost.
- **Fast Recovery Area**: A central area for recovery-related files introduced in Oracle 10g.

Backup and Recovery

Regularly scheduled backups are essential. Media failures typically require restoring a data file from backup before recovery can start. Features like multiplexing control and redo log files, using ARCHIVELOG mode, and configuring a Fast Recovery Area can enhance database availability and reduce recovery time.

▼ Understanding Control Files.

Control files are critical binary files that contain metadata about the database structure. They are essential for the database's operation and recovery processes.

▼ Control File Architecture

- **Size and Nature**: Control files are relatively small (in the megabyte range) and continuously updated. They should not be manually edited.
- **Contents**: Control files include:
    - Database name
    - Database creation timestamp
    - Data files' names, locations, and statuses
    - Redo log files' names and locations
    - Redo log archive information
    - Tablespace names
    - Current log sequence number
    - Most recent checkpoint information
    - Undo segments' start and end
    - RMAN (Recovery Manager) backup information
- **Updates**: Oracle processes like LGWR (Log Writer), CKPT (Checkpoint), and ARCn (Archiver) continuously update the control files with essential data.

Key Features

- **MAX Clauses**: The size is determined by MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES clauses during database creation. Oracle pre-allocates space in the control file based on these maximums.
- **Backup**: Control files should be backed up after any structural changes in the database.
- **Record Sections**: Control files have reusable and non-reusable sections. RMAN information is stored in the reusable section, which is managed in a circular fashion based on the CONTROL_FILE_RECORD_KEEP_TIME parameter.

Multiplexing Control Files

Control files are essential for database operations. Oracle recommends having at least three copies of control files on different disks, ideally managed by different disk controllers. This ensures redundancy and reduces the risk of a single point of failure.

Using init.ora

1. **Shut down the database**:

```
SQL> SHUTDOWN NORMAL;
```

2. **Copy the control file**:

```
$ cp /u02/oradata/ord/control01.ctl /u05/oradat
a/ord/control04.ctl;
```

3. **Update the init.ora file**:

```
CONTROL_FILES = ('/u02/oradata/ord/control01.ct
l',
                 '/u03/oradata/ord/control02.ct
l',
                 '/u04/oradata/ord/control03.ct
l',
                 '/u05/oradata/ord/control04.ct
l');
```

4. **Start the instance**:

```
SQL> STARTUP;
```

▼ Using spfile

1. **Alter the spfile**:

```
SQL> ALTER SYSTEM SET CONTROL_FILES = '/ora01/or
adata/MYDB/ctrlMYDB01.ctl',
                                 '/ora02/oradat
a/MYDB/ctrlMYDB02.ctl',
                                 '/ora03/oradat
a/MYDB/ctrlMYDB03.ctl',
                                 '/ora04/oradat
a/MYDB/ctrlMYDB04.ctl' SCOPE=SPFILE;
```

2. **Shut down the database**:

```
SQL> SHUTDOWN NORMAL;
```

3. **Copy the control file**:

```
$ cp /ora01/oradata/MYDB/ctrlMYDB01.ctl /ora04/o
radata/MYDB/ctrlMYDB04.ctl;
```

4. **Start the instance**:

```
SQL> STARTUP;
```

Note: You can create an spfile from an init file using `CREATE SPFILE FROM PFILE`.

▼ Understanding Checkpoints

Role of the CKPT Process

- **CKPT Process**: Manages the duration of instance recovery by updating the control file and data file headers with the last successful transaction's System Change Number (SCN).
- **SCN**: A sequential number assigned to each transaction, recorded in the control file against data files that are taken offline or made read-only.

Checkpoint Triggers

- **Automatic Triggers**:
  - Occur during redo log file switches, either when the file fills up or manually.
  - Routine writes by DBWn (Database Writer) processes in conjunction with CKPT.
  - During consistent database shutdowns.

Types of Checkpoints

- **Full Checkpoint (Thread Checkpoint)**:
  - Initiated manually using `ALTER SYSTEM CHECKPOINT`.
  - Occurs during normal or immediate database shutdowns.
  - Happens during online redo log switches.
- **Incremental Checkpoint**:
  - Advances the checkpoint in data files as DBWn writes dirty buffers.
  - CKPT only writes the checkpoint position in the control file, not the data file headers, during redo log switches.
  - Occurs automatically every 3 seconds when DBWn writes dirty blocks to data files.

Key Points

- **Dirty Buffers**: Data changes stored in memory that need to be written to disk.
- **MTTR (Mean Time To Recovery)**: Reduced by frequent checkpoints ensuring committed data is regularly written to disk.
- **DBWn's Role**: Wakes up every 3 seconds to write any dirty blocks to data files, advancing the checkpoint.

Summary

Checkpoints ensure database consistency and manage recovery times by periodically writing data changes from memory to disk. The CKPT process updates control files and data file headers with SCNs, while the DBWn process writes dirty buffers to disk. Checkpoints occur automatically during redo log switches and at regular intervals, ensuring minimal recovery time after system failures.

▼ Understanding Redo Log Files

Overview

- **Purpose**: Redo log files record all changes to the database before they are written to data files, protecting against data loss.
- **Importance**: Essential for recovering from instance or media failures by rolling data files forward to the last committed transaction.

▼ Redo Log File Architecture

- **Redo Records**: Contain change vectors, each describing changes made to a single block in the database.
- **LGWR Process**: Writes redo information to the online redo log files under various conditions:
    - When a transaction commits.
    - When the redo log buffer is one-third full.
    - When the buffer has about 1MB of changes.
    - During database checkpoints.

    - Every 3 seconds if no other writes have occurred.
- **Redo Log Groups**: Each database has its own set of online redo log groups, with each group potentially having multiple members. In RAC configurations, each instance has its own redo thread.
- **Committed and Uncommitted Transactions**: Both are recorded in the redo log file, with committed transactions assigned a system change number (SCN).

▼ Log Switch Operations

- **Current Log File**: The redo log file actively being written to.
- **Active Log Files**: Required for instance recovery.
- **Inactive Log Files**: Not currently required for recovery.
- **Log Switches**: Occur when Oracle finishes writing to one log group and starts on the next. They can be triggered automatically or manually using `ALTER SYSTEM SWITCH LOGFILE`.

▼ Best Practices

- **Multiplexing**: Having at least two members for each redo log group reduces the likelihood of data loss.
- **Sizing**: Properly size redo log files to avoid frequent log switches and checkpoints.
- **Performance**: Store redo log files on a separate disk to maximize I/O performance and avoid conflicts with other critical files like SYSTEM or UNDOTBS.

▼ Checkpoints and Redo Logs

- **Checkpoint Relationship**: Checkpoints are tied to redo log switches, ensuring that changes are flushed from the buffer cache to disk and control files/data files are updated.
- **Forcing Checkpoints**: Can be done using `ALTER SYSTEM CHECKPOINT` or by forcing a log-file switch.
- **Tuning Checkpoints**: Use the `FAST_START_MTTR_TARGET` parameter to control the maximum time allowed for instance recovery.

In summary, redo log files are critical for database recovery, capturing all changes made to the database. Proper management and configuration of redo logs and checkpoints ensure data integrity, efficient recovery processes, and optimal database performance.

▼ Multiplexing Redo Log Files

Purpose of Multiplexing:
Multiplexing online redo log files involves maintaining multiple identical copies of redo logs to protect against data loss due to file damage. The process ensures database continuity even if one log file becomes unavailable, as long as another copy exists. Each set of identical redo log files forms a redo group, identified by a unique integer, and each file in the group is called a redo member.

Recommendations:

1. **Disk Allocation:** Distribute redo log members across different disks to avoid a single point of failure.
2. **Number of Groups:** Maintain at least two redo log groups for normal operation and consider having an extra group as a precaution.
3. **Space Efficiency:** The additional space for extra log groups is minimal compared to the benefits of avoiding log-related issues.

▼ Creating Redo Log Groups

To create redo log groups during database creation, use the `CREATE DATABASE` statement with the `LOGFILE` clause:

```
CREATE DATABASE "MYDB01"
...
LOGFILE
GROUP 1 ('/path/redo0101.log', '/path/redo0102.lo
g') SIZE 500M,
GROUP 2 ('/path/redo0201.log', '/path/redo0202.lo
g') SIZE 500M;
```

To add groups to an existing database, use the `ALTER DATABASE` command:

```
ALTER DATABASE ADD LOGFILE
GROUP 3 ('/path/redo0301.log', '/path/redo0302.lo
g') SIZE 100M;
```

- **Adding New Members**

  If additional members need to be added to an existing group, use:

  ```
  ALTER DATABASE ADD LOGFILE MEMBER '/path/redo0203.l
  og' TO GROUP 2;
  ```

- **Renaming Log Members:**

  To rename a log member:

  1. Shutdown the database.
  2. Rename the file at the OS level.
  3. Restart and mount the database.
  4. Update the control file:

  ```
  ALTER DATABASE RENAME FILE 'old_file_name' TO 'new_
  file_name';
  ```

- **Dropping Redo Log Groups and Members:**

  To drop a redo log group, ensure it is inactive and use:

  ```
  ALTER DATABASE DROP LOGFILE GROUP 3;
  ```

  To drop a member from a group:

  ```
  ALTER DATABASE DROP LOGFILE MEMBER '/path/redo0203.
  log';
  ```

  Ensure there are at least two groups, and each group maintains the same number of members.

▼ **Clearing Online Redo Log Files:**

To clear corrupted redo log files without dropping them:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

Use `UNARCHIVED` to clear unarchived files if necessary, but perform a full backup afterwards:

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;
```

These practices ensure the robustness and reliability of the database's redo log system, minimizing data loss and downtime.

You can clear the redo logs by choosing Clear Logfile from the Actions drop-down box
in EM Cloud Control. The other options available in the drop-down box are as follows:
■■ Create Like
■■ Force Checkpoint
■■ Generate DDL
■■ Sizing Advice
■■ Switch Logfile

▼ Understanding and Configuring Archived Redo Log (ARCHIVELOG) Files

Purpose of ARCHIVELOG Mode:
ARCHIVELOG mode enhances database recoverability by saving copies of online redo log files before they are overwritten. This process enables recovery from media failures, not just instance failures, allowing databases to be restored to the point of failure or any specific point in time.

▼ **Archived Redo Log File Architecture:**

- **Archiving Process:** The ARCn background processes copy full redo log files to specified archive destinations before they are overwritten. This ensures that committed transactions can be recovered.

- **ARCHIVELOG vs. NOARCHIVELOG Mode:** In ARCHIVELOG mode, redo log files are archived, allowing for media recovery. In NOARCHIVELOG mode, redo logs are overwritten without being archived, limiting recovery options.

▼ **Setting the Archive Destination:**

- **LOG_ARCHIVE_DEST_n Parameter:** Up to 31 archiving destinations can be specified, with local and remote locations. Syntax:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/archive/MYDB01 MAN
DATORY'
LOG_ARCHIVE_DEST_2 = 'SERVICE=STDBY01 OPTIONAL REO
PEN 60';
```

- **Minimum Successful Destinations:** The `LOG_ARCHIVE_MIN_SUCCEED_DEST` parameter specifies the minimum number of successful writes required to continue overwriting redo logs.

▼ Setting ARCHIVELOG

Enabling ARCHIVELOG Mode:

1. Shut down the database.
2. Set appropriate initialization parameters.
3. Start and mount the database.
4. Enable ARCHIVELOG mode:

```
ALTER DATABASE ARCHIVELOG;
```

5. Open the database:

```
ALTER DATABASE OPEN;
```

6. Back up the database.

Disable ARCHVLOG

To disable ARCHIVELOG mode, follow similar steps, using `ALTER DATABASE NOARCHIVELOG` in step 4 without 2and 6.

Configuration and Management:

- **Archive Filename Format:** Use the `LOG_ARCHIVE_FORMAT` parameter to specify the naming convention for archived redo logs, ensuring uniqueness and traceability.
- **Monitoring:** Use dynamic performance views like `V$ARCHIVE_PROCESSES` and `V$DATABASE` to monitor the status and configuration of archiving processes.

- The dynamic performance view `V$DATABASE` tells you whether or not you are in ARCHIVELOG mode

```
SELECT dbid, name, created, log_mode FROM v$database
```

Managing Space for Archived Logs:

- Ensure adequate disk space for archived logs to prevent database hangs.
- Use RMAN and the Fast Recovery Area to automate backup and retention policies, managing disk space effectively and ensuring recovery capabilities.

Key Practices:

- Always run production databases in ARCHIVELOG mode to protect against data loss.
- Regularly monitor archive log space and set up automated backups to avoid running out of space.
- Configure multiple archive destinations for redundancy and ensure that archiving processes complete successfully before overwriting redo logs.

▼ Understanding the Fast Recovery Area

Purpose and Advantages:

- The Fast Recovery Area (FRA) is a unified storage space for all recovery-related files and activities in an Oracle database.
- Using disk space as the primary medium for recovery operations allows for faster database recovery compared to tape.

Components Stored in FRA:

1. **Control Files:** A copy of the control file, which can be used in case of media failure.
2. **Archived Log Files:** Automatically set to the FRA location.
3. **Flashback Logs:** Stored if Flashback Database is enabled.
4. **Control File and SPFILE Autobackups:** Stored if RMAN is configured for autobackups.
5. **Data File Copies:** Default destination for RMAN BACKUP AS COPY files.
6. **RMAN Backup Sets:** Default location for both backup sets and image copies.

Configuration of FRA:

- Two key initialization parameters:
  - `DB_RECOVERY_FILE_DEST_SIZE`: Defines the size of the FRA.
  - `DB_RECOVERY_FILE_DEST`: Specifies the physical location of the FRA.

SQL Commands to Configure FRA:

```
ALTER SYSTEM SET db_recovery_file_dest_size = 80G SCOPE=BOTH;
ALTER SYSTEM SET db_recovery_file_dest = '/OraFlash' SCOPE=BOTH;
```

- These commands dynamically set the size and location of the FRA without needing to restart the database.

Monitoring and Managing FRA:

- Use the `V$RECOVERY_AREA_USAGE` view to monitor space usage.
- The FRA should be large enough to store all necessary recovery files, and space can be adjusted dynamically if needed.
- EM Cloud Control can be used for configuration and management, including enabling Flashback Database.

Dealing with Space Issues in FRA:

- Oracle automatically manages space in the FRA, deleting obsolete files when space is needed.
- Alerts are issued at 85% and 97% capacity usage.
- Options to manage space include:
    1. Adjusting the retention policy.
    2. Increasing disk space.
    3. Backing up FRA files to other destinations.
    4. Deleting unnecessary files using RMAN.

Automatic and Manual Space Management:

- Oracle Database issues alerts and manages space based on the retention policy.
- Manual space management can be done using RMAN commands like BACKUP RECOVERY AREA.

Summary

The Fast Recovery Area (FRA) in Oracle Database is a critical component for efficient and rapid recovery operations. It centralizes all recovery-related files and activities, allowing for streamlined management and quicker recovery times compared to tape-based solutions. Proper configuration and monitoring of the FRA ensure that recovery operations can be performed effectively, with sufficient space to store necessary files and logs. Managing space through alerts and retention policies helps maintain the FRA's functionality and prevents database hangs due to space issues.

▼ Performing Backups

Backup Strategy Considerations:

- The strategy depends on database activity, required availability (SLAs), and tolerable downtime during recovery.

▼ Backup Terminology

1. **Whole Database Backup:** Includes all data files, at least one control file, archived log files, and a server parameter file (SPFILE).
2. **Partial Database Backup:** Includes selected tablespaces or data files, and optionally a control file.
3. **Full Backup:** Backs up all blocks of every data file in a whole or partial database backup.
4. **Incremental Backup:** Copies data blocks changed since a previous backup. Levels range from 0 to 4, with 0 being a baseline (full) backup.
5. **Consistent Backup:** Performed while the database is closed; no recovery needed as SCNs in control and data files match.
6. **Inconsistent Backup:** Performed while the database is open; requires recovery to synchronize SCNs.

Backup Methods:

1. **User-Managed Backup:** Uses operating system utilities and SQL commands.
2. **Recovery Manager (RMAN):** Oracle's recommended tool for creating and managing backups.

▼ Backing Up the Control File

- **Text Backup:** Creates an editable text file of the control file.

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

- **Binary Backup:** Creates a binary copy of the control file.

```
ALTER DATABASE BACKUP CONTROLFILE TO '/path/to/backu
```

```
p.bkp';
```

- **RMAN Backup:** Backs up the control file using RMAN.

```
RMAN> BACKUP CURRENT CONTROLFILE;
```

▼ Backing Up the Database

Oracle Database 12c offers several backup methods depending on the database mode (ARCHIVELOG or NOARCHIVELOG) and the tools used (OS utilities or RMAN).

Backup Modes:

1. **ARCHIVELOG Mode:**
    - **Online Backup (Inconsistent/Hot Backup):** The database remains open and operational during the backup.
    - **Offline Backup (Consistent/Cold Backup):** The database is shut down before the backup.
2. **NOARCHIVELOG Mode:**
    - Only offline backups are possible.

▼ User-Managed Backups:

Cold Backups:

- **Steps:**
    1. Shut down the database cleanly using `SHUTDOWN IMMEDIATE` or `SHUTDOWN TRANSACTIONAL`.
    2. Copy all control files, data files, parameter files (init file or spfile), and password files using OS commands.
    3. Optionally, copy redo logs if the shutdown was not clean.
- **Identify Files:**

```
SELECT name FROM V$CONTROLFILE;
SELECT name FROM V$DATAFILE;
```

Hot Backups:

- **Steps:**
  1. Ensure the database is in ARCHIVELOG mode.
  2. Place the tablespace in backup mode:
     ```
     ALTER TABLESPACE users BEGIN BACKUP;
     ```
  3. Use OS utilities to copy the data files of the tablespace.
  4. Take the tablespace out of backup mode:
     ```
     ALTER TABLESPACE users END BACKUP;
     ```
  - To back up the entire database, use:
    ```
    ALTER DATABASE BEGIN BACKUP;
    ALTER DATABASE END BACKUP;
    ```

▼ Using RMAN for Backups:

- RMAN simplifies backup and recovery operations and supports all types of backups: whole or partial, full or incremental, consistent or inconsistent.

▼ RMAN Configuration:

- **Backup Settings:** Configure using EM Cloud Control or RMAN command-line interface.
- **Key Parameters:**
  - **Parallelism:** Utilizes multiple CPUs or disk controllers to reduce backup time.

    - **Disk Backup Location:** Specify where backups should be stored.
    - **Disk Backup Type:** Options include Image Copy, Backup Set, or Compressed Backup Set.
    - **Automatic Control File and SPFILE Backup:** Recommended best practice.

Example RMAN Configuration:

```
rman target / nocatalog
RMAN> SHOW ALL;
```

Example RMAN Backup Commands:

```
RMAN> BACKUP DATABASE;
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

Additional Notes:

- **Block Change Tracking (BCT):** Helps identify changed blocks for incremental backups, reducing backup time.

  ```
  ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;
  ```

▼ Understanding Image Copies and Backup Sets

Image Copies:

- Duplicates of data files or archived redo log files.
- Every block of every file is backed up.
- Can be created using RMAN or OS commands.
- Can only be copied to a disk location.

Backup Sets:

- Copies of one or more data files or archived redo log files in a proprietary format.

- Do not include unused blocks in the data files, saving space.
- Can be compressed for further space savings.
- Can be written to disk or tape/secondary storage.

▼ Creating Full and Incremental Backups

Oracle-recommended Backup Strategy:

1. **Level 0 Backup:**
   - A one-time, whole-database baseline incremental level 0 online backup.
   - Command: `RMAN> backup incremental level 0 as compressed backupset database;`
2. **Level 1 Incremental Backups:**
   - Daily level 1 incremental backups, backing up only the changed blocks.
   - Command: `RMAN> backup incremental level 1 as compressed backupset database;`
3. **Incrementally Updated Backup:**
   - Uses incremental backup to update changed blocks in an existing image copy.
   - Saves time during recovery as incremental backup is already applied.
   - Script:

```
run {
  recover copy of database with tag 'inc_upd_img';
  backup incremental level 1 for recover of copy with tag 'inc_upd_img' database;
}
```

EM Cloud Control Backup Scheduling:

- Provides options for Oracle-suggested or customized backup policies.
- Oracle-suggested policy:
  - Day 1: Level 0 image backup.
  - Day 2+: Incremental level 1 backups applied to the previous backup.
- Custom backup schedules for full database, tablespace, data file, archive log, and recovery area files.
- Allows scheduling one-time or repeating backups.

▼ Managing Backups

▼ Maintaining Current Backups:

- Functions available in EM Cloud Control for backup management:
  - **Catalog Additional Files:** Adds RMAN catalog for external backups.
  - **Crosscheck All:** Verifies cataloged backups against actual files.
  - **Delete All Obsolete:** Removes backups not needed by retention policy.
  - **Delete All Expired:** Deletes catalog entries for missing backups.

▼ Viewing Backup Reports:

- EM Cloud Control's Backup Reports menu shows detailed status reports.
- RMAN command for listing backups:

  ```
  RMAN> list backup;
  ```

Summary

- **Cold Backups:** Suitable for databases in any mode but require downtime.
- **Hot Backups:** Require ARCHIVELOG mode and allow database operations to continue during the backup.
- **RMAN:** Preferred for its automation, versatility, and comprehensive backup capabilities. Configure using EM Cloud Control or command line for optimal results.

▼ Understanding Types of Database Failures

Database-related failures can be categorized into six general types. Knowing which category a failure belongs to helps quickly determine the nature of the recovery effort needed. Here are the six types:

▼ 1. Statement Failures

Statement failures occur when a single database operation, such as an INSERT or table creation, fails. Common problems and solutions include:

- **Lack of Privileges:** Provide appropriate privileges or create views with the necessary privileges.
- **Running Out of Space:** Add space to the tablespace, increase the user's quota, or enable resumable space allocation.
- **Invalid Data Entry:** Use constraints and triggers to enforce data integrity.

- **Logic Errors in Applications:** Work with developers to correct errors or add logic to handle errors gracefully.

▼ 2. User-Process Failures

These failures involve the abnormal termination of a user session. The PMON background process handles cleanup, including rolling back uncommitted transactions and releasing locks. Common causes include:

- Users closing SQL*Plus without logging out.
- Sudden workstation reboots.
- Application exceptions causing crashes.

▼ 3. Network Failures

Network failures occur due to issues in the network path between the client and the database server, or failures in the server's network components, like the listener process or network card. Solutions include:

- Providing redundant network paths and listener connections.
- Using redundant network cards on the server.

▼ 4. User-Error Failures

These occur when users inadvertently delete or modify data or drop database objects. Solutions include:

- Using ROLLBACK for uncommitted changes.
- Utilizing Flashback Query or LogMiner for committed changes.
- Recovering dropped tables using recycle-bin functionality or TSPITR and Flashback Database recovery.

▼ 5. Instance Failures

Instance failures happen when the database instance shuts down unexpectedly without synchronizing database files. Causes include:

- Power outages.
- Hardware failures.
- Oracle background process failures.

- Intentional shutdowns (e.g., SHUTDOWN ABORT).

Recovery involves running the STARTUP command, allowing Oracle to perform automatic instance recovery using online redo logs and undo data.

▼ 6. Media Failures

Media failures result from the loss of one or more database files (data files, control files, or redo log files). Causes include:

- Disk drive failures.
- Disk controller failures.
- Inadvertent deletion or corruption of database files.

Preparation involves:

- Mirroring control files and redo log files.
- Ensuring the availability of full backups and archived redo log files.

In the following sections, detailed solutions for media failures will be provided, including recovery from the loss of control files, data files, and redo log files.

▼ Performing Recovery Operations

Overview

When a database failure occurs, quick recovery is achievable if backup guidelines were followed and failure types are understood. Knowing Oracle instance startup and failure phases is crucial as certain recoveries must occur in specific phases. The steps include understanding startup phases, instance recovery mechanics, tuning, and recovering from user errors and media failures.

▼ Understanding Instance Startup

1. **SHUTDOWN**: No active background processes. Use `STARTUP` to initiate.
2. **NOMOUNT**: Accesses initialization-parameter file (init.ora or spfile).
3. **MOUNT**: Checks control files; remains in NOMOUNT if files are missing/corrupted.

4. **OPEN**: Requires all redo log groups and online data files to be available.

Errors during startup notify missing/corrupted files, which can be listed using:

```
SELECT file#, error FROM v$recover_file;
SELECT file#, name FROM v$datafile JOIN v$recover_file U
SING (file#);
```

▼ Handling Missing/Corrupted Files

- Missing data files: Use `V$RECOVER_FILE` to list files.
- Offline files: Instance can start if SYSTEM or UNDO tablespace files are unaffected.
- Out-of-sync files: Automatic instance recovery using redo logs; otherwise, media recovery with archived logs.

  ```
  SQL> select file#, error from v$recover_file;
  SQL> select file#, name from v$datafile join v$recove
  ```

▼ Keeping an Instance from Failing

- Critical failures: Loss of control file, redo log group, or SYSTEM/UNDO data file causes instance failure.
- Non-critical failures: Use `SHUTDOWN ABORT` to force shutdown without resynchronizing files. `STARTUP FORCE` can restart the instance.

▼ Recovering from Instance Failure

Instance failures prevent data file and control file synchronization. Recovery, or crash recovery, is done by starting the database with the `STARTUP` command. The process involves:

1. **Finding out-of-sync data files**.
2. **Using redo logs to roll-forward data files** to pre-failure state.
3. **Opening the database**.

4. **Rolling back uncommitted transactions** using the undo tablespace.

Instance recovery is required after an instance crash or `SHUTDOWN ABORT`.

▼ Tuning Instance Recovery

Process Overview

- Before receiving a "Commit complete" message, data must be written to a redo log file, and later, to data files.
- **Checkpoints** track transactions not yet written to data files, identified by SCN.
- Instance recovery time is the duration needed to update data files from the last checkpoint to the latest SCN.
- To avoid performance issues, the checkpoint to redo log group end distance must not exceed 90% of the log size.

Key Parameters

1. **FAST_START_MTTR_TARGET**: Sets recovery time target in seconds (0 to 3,600).
    - Default is 0, disabling the target.
    - Lower values increase recovery speed but may impact performance.
    - High values delay recovery, affecting system availability post-failure.
2. **LOG_CHECKPOINT_TIMEOUT**: Max time a modified block stays in buffer cache before being written to disk.
3. **LOG_CHECKPOINT_INTERVAL**: Number of redo log blocks between checkpoints.
4. **FAST_START_IO_TARGET**: Sets recovery target by I/Os instead of time.

Adjusting MTTR

- Set via EM Cloud Control or SQL*Plus:

```
ALTER SYSTEM SET fast_start_mttr_target=60 SCOPE=BOT
H;
```

- Ensure **LOG_CHECKPOINT_TIMEOUT**, **LOG_CHECKPOINT_INTERVAL**, and **FAST_START_IO_TARGET** are removed when using **FAST_START_MTTR_TARGET** to avoid conflicts.

Monitoring and Recommendations

- Use **V$MTTR_TARGET_ADVICE** and **V$INSTANCE_RECOVERY** views for insights and optimization.
- **OPTIMAL_LOGFILE_SIZE** helps determine the best size for redo logs.

▼ Recovering from User Errors

Scenarios and Tools

- User errors include accidental data changes, deletions, or dropped tables.
- **Flashback Technologies**:
  - **Flashback Query**: Retrieve table rows from a previous state using the `AS OF TIMESTAMP` clause.
  - **Flashback Drop**: Restore dropped tables from the recycle bin.
  - **Flashback Table**: Recover an entire table and its objects to a specific point in time.
  - **Flashback Transaction**: Rollback specific transactions and dependent transactions.
  - **LogMiner**: Query previous transactions in redo logs.

▼ Flashback Query

- Allows viewing table contents from a past state.
- **Setup**:
  - Ensure sufficient undo tablespace.

    - Set **UNDO_RETENTION** parameter (default is 900 seconds).

- Example:

```
SELECT employee_id, last_name, email
FROM hr.employees
AS OF TIMESTAMP (systimestamp - interval '15' minut
e)
WHERE employee_id = 101;
```

▼ Flashback Drop and Recycle Bin

- **Recycle Bin**: Logical tablespace structure holding dropped tables and related objects.
- **Retrieving Tables**:
    - Use `FLASHBACK TABLE ... TO BEFORE DROP`.
    - If table name conflicts, use `RENAME TO` clause.

Considerations and Limitations

- Recycle bin is only for non-SYSTEM locally managed tablespaces.
- Dependent objects in dictionary-managed tablespaces are protected.
- Some dependent objects like bitmap join indexes, foreign key constraints, and materialized view logs are not saved.
- Explicitly dropping an index does not place it into the recycle bin.

▼ Using Flashback Table

- **Purpose**: Recovers tables to a specific point in time without extensive operations like tablespace point-in-time recovery or Flashback Database.
- **Mechanism**: Rolls back only the changes to the table(s) and their dependent objects using data in the undo tablespace.
- **Difference from Flashback Drop**:

- **Flashback Table**: Undoes recent transactions on an existing table.
- **Flashback Drop**: Recovers a dropped table using the recycle bin.

Requirements and Steps for Flashback Table

1. **Enable Row Movement**: Necessary because DML operations used in flashback change row IDs.

   ```
   ALTER TABLE <table_name> ENABLE ROW MOVEMENT;
   ```

2. **Execute Flashback Table**: Command to revert tables to a past state.

   ```
   FLASHBACK TABLE <table1>, <table2> TO TIMESTAMP <timestamp>;
   ```

   - Example: Recovering accidentally deleted rows.

   ```
   FLASHBACK TABLE hr.employees, hr.departments TO TIMESTAMP systimestamp - interval '15' minute;
   ```

Considerations

- **Availability**: Only the involved tables are locked during the operation, which does not affect the rest of the database.
- **Integrity Constraints**: Maintained during flashback operations; typically group related tables.
- **Triggers**: Disabled by default during flashback; can be enabled using `ENABLE TRIGGERS` clause.

  ```
  FLASHBACK TABLE hr.employees TO TIMESTAMP TO_TIMESTAMP('02-NOV-12 22:00', 'DD-MON-YY HH24:MI') ENABLE TRIGGERS;
  ```

▼ Performing Flashback Table Using EM Cloud Control

1. **Navigate to Perform Recovery**: Under the Availability menu.
2. **Select Recovery Scope**: Choose "Tables" to access options for:
    - Flashback Existing Tables
    - Flashback Dropped Tables

▼ Using Flashback Transaction

- **Purpose**: Undo a specific transaction and its dependent transactions using `DBMS_FLASHBACK.TRANSACTION_BACKOUT`.
- **Prerequisites**:
    - Database in ARCHIVELOG mode.
    - Supplemental logging enabled.
    - User privileges: SELECT ANY TRANSACTION, EXECUTE on DBMS_FLASHBACK, appropriate DML privileges.
- **EM Cloud Control**: Perform flashback transaction via the Backup and Recovery menu.

▼ Using LogMiner

- **Purpose**: View past database activity by extracting DDL and DML statements from redo log files.
- **Functionality**:
    - Search a time period for all changes to a table.
    - Provides SQL statements to reverse changes.
    - Works with both online and archived redo logs.
- **Difference from Flashback Query**:
    - **Flashback Query**: Uses undo information to show table contents at a past time.

    - **LogMiner**: Uses redo logs to track changes and generate reversal SQL commands.

By understanding these tools and methods, database administrators can efficiently recover from logical corruptions, accidental deletions, and other user errors without extensive downtime or impact on overall database availability.

Example Recovery Scenario

1. Application admin mistakenly deletes rows and commits.
2. DBA uses Flashback Query to retrieve and reinstate rows.
3. Insert back missing rows without taking applications offline.
4. Example query to retrieve deleted rows:

   ```
   SELECT * FROM vms.dvbt606a
   AS OF TIMESTAMP to_timestamp ('12-Sep-12 12:20', 'DD-Mon-RR HH24:MI');
   ```

▼ Recovering from the Loss of a Control File

1. **Shutdown**: If the instance is not shut down, use `SHUTDOWN ABORT` to force a complete shutdown.
2. **Copy Control File**: Copy a good control file to the location of the missing/corrupted file or to a new location if the disk failed. Update the initialization parameter file if needed.
3. **Restart Instance**: Start the instance using `STARTUP`.

▼ Example Recovery Scenario

- **Start in NOMOUNT Mode**: Necessary because MOUNT mode checks for all control file copies.

  ```
  SQL> startup nomount
  ```

- **View Current Control Files**: Use `V$SPPARAMETER` to see the control file references.

```
SQL> select name, value from v$spparameter where na
me = 'control_files';
```

- **Update Control Files in SPFILE**: Alter the `CONTROL_FILES` parameter and restart the instance.

```
SQL> alter system set control_files = '/u02/oradat
a/ord/control01.ctl', '/u06/oradata/ord/control02.c
tl' scope = spfile;
SQL> shutdown immediate
SQL> startup
```

- **Confirm Update**: Verify that the control file references are updated.

```
SQL> select name, value from v$spparameter where na
me = 'control_files';
```

▼ Using the Data Recovery Advisor (DRA)

- **Purpose**: DRA diagnoses database failures and suggests recovery options.
- **User Interfaces**: Available via EM Cloud Control and RMAN.
    - **EM Cloud Control Methods**:
        - Perform Recovery screen (if failures are detected, the Advise and Recover button is enabled).
        - Support Workbench from the Diagnostics menu (shows failures and invokes DRA).
        - Advisor Central page (access DRA via a link under Advisors).

▼ Health Monitoring and Proactive Checks

- **Health Monitor (HM)**: Proactively monitors database health and reports to DRA.

- **Failure Checks**: Reactive (triggered by errors) and proactive checks by DRA.
- **Common Failures Addressed by DRA**:
  - Missing or corrupted data files.
  - Incorrect OS permissions.
  - Offline tablespaces.
  - I/O failures.
  - Logical and physical corruptions.

Conclusion

Understanding and utilizing these recovery methods ensures efficient resolution of control file loss and other database failures, maintaining database integrity and availability.

▼ Recovering from the Loss of a Redo Log File

▼ Importance of Redo Log Files

- **Database Availability**: An instance stays operational as long as at least one redo log group member is available.
- **Monitoring**: Alert log and V$LOGFILE dynamic performance view provide information on the status of redo log files.

▼ Recovery Steps for a Missing Redo Log Group Member

1. **Identification**: Verify which redo log group member is missing.
2. **Archiving**: Archive the redo log group's contents if needed.
3. **Recreation**: Clear the log group to recreate the missing redo log file members.

▼ Example Recovery Scenario

- **Identifying Missing Member**: Check the status of redo log file groups using `V$LOGFILE`.

  ```
  SQL> SELECT group#, member FROM v$logfile;
  ```

- **Removing the Missing Member**: Simulate the loss of a redo log file member.

```
SQL> ! rm /u01/app/oracle/oradata/ocadb1/redo02.log
```

- **Archiving and Recreation**: Archive and recreate the missing redo log-file group member.

```
SQL> alter system archive log group 2;
SQL> alter database clear logfile group 2;
```

▼ Monitoring and Further Recovery Options

- **Alert Log**: Reflects the issue and its resolution.
- **Database Recovery Advisor (DRA)**: Provides detailed failure information and recovery options.
  - Priority adjustments and incident closure available in DRA.
- **EM Cloud Control**: Actions like "Switch Log File" and "Clear Log File" facilitate redo log group member recovery.

By following these steps and utilizing monitoring tools like DRA and EM Cloud Control, administrators can efficiently recover from the loss of a redo log file member, ensuring database integrity and availability.

▼ Recovering from the Loss of a Non-System-Critical Data File

▼ Loss in NOARCHIVELOG Mode

- **Recovery Process**: Requires complete restoration of the database, including control files and all data files. All changes since the last backup must be reentered.

▼ Loss in ARCHIVELOG Mode

- **Scope of Affected Objects**: Limited to objects in the missing file, allowing recovery while the rest of the database remains online. Committed transactions in the lost data file do not need reentry.

- **Recovery Approach**: Less complex compared to system-critical data file recovery. Database remains available to users, except for the data files being recovered.

▼ Recovery Options

- **Using Data Recovery Advisor (DRA)**:
  - **Manual Actions**: DRA provides recommendations for recovery actions.
  - **RMAN Script Generation**: Generates RMAN scripts for manual execution.
  - **Job Submission**: Allows for job submission to start the restore and recovery process.

Data Recovery Advisor Views

- **V$IR_ Views**: Newly added views in Oracle Database 12c for DRA management, including failure listing `V$IR_FAILURE`, repair recommendations `V$IR_REPAIR` , and manual checklist `V$IR_MANUAL_CHECKLIST`.

▼ Recovering from the Loss of a System-Critical Data File

Recovery in Different Modes

- **ARCHIVELOG Mode**: Preferred mode for production databases; allows recovery without reentering committed transactions.
- **NOARCHIVELOG Mode**: Requires complete restoration of the database, including control files and all data files, with reentry of changes made since the last backup.

▼ Loss in NOARCHIVELOG Mode

- **Recovery Process**: Complete restoration of the database, necessitating reentry of changes since the last backup.
- **Cold Backup Requirement**: Restoration must have been from a cold backup.

▼ Loss in ARCHIVELOG Mode

- **Database State**: Recovery performed while the database is in MOUNT state, not OPEN.
- **Transaction Integrity**: No need to reenter committed transactions due to ARCHIVELOG mode.

▼ Recovery Procedure

1. **Shutdown and Mount Database**: If not already aborted, shut down the database and start it in MOUNT state.
2. **Restore and Recover**: Once in MOUNT state, restore and recover the missing data file.
3. **Open Database**: After recovery completion, open the database using `ALTER DATABASE OPEN`.

▼ Using Data Recovery Advisor (DRA)

- **Initiating Recovery**: Access DRA through EM Cloud Control's Availability screen.
- **Recovery Scope**: Select Datafiles, choose Restore to Current Time, and add files for recovery.
- **Submission**: Submit the recovery job and open the database after successful completion.

By following these procedures and leveraging tools like DRA, administrators can efficiently recover from the loss of system-critical data files, ensuring minimal downtime and data loss.

▼ Table Recovery Using RMAN

Oracle Database 12c Advancements

- **Streamlined Process**: RMAN in Oracle Database 12c simplifies table recovery from backups.
- **Previous Versions**: Recovery involved resource provisioning, database restoration, table export/import, and temporary database cleanup.

▼ RMAN Table Recovery

1. **RECOVER TABLE Command**: Initiates the recovery process in RMAN.
2. **Backup Piece Determination**: RMAN identifies required backup pieces based on the specified recovery time.
3. **Temporary Database Creation**: RMAN creates a temporary database containing necessary tablespaces and performs recovery.
4. **Data Pump Export**: Exports tables specified in the RECOVER command from the temporary database.
5. **Table Import**: Imports the table into the target database, allowing renaming or relocation to another schema.

▼ Benefits

- **Automation**: RMAN handles all recovery steps, reducing manual intervention and complexity.
- **Efficiency**: Eliminates the need for resource provisioning, separate database restoration, and manual export/import steps.
- **Ease of Use**: Simplifies the table recovery process for administrators.

▼ Additional Resources

- **Oracle Documentation**: Chapter 22 of "Oracle Database Backup and Recovery User's Guide 12c Release 1 (12.1)" provides detailed information on recovering tables and table partitions from RMAN backups. Available at Oracle Documentation.

▼ Summary

Summary: Database Backup and Recovery Essentials

Key Elements for Smooth Recovery

- **Control Files**: Store metadata about database structures.
- **Online Redo Log Files**: Ensure transaction durability and performance.
- **Archived Redo Log Files**: Copy online redo log files to prevent data loss.

Multiplexing and Redundancy

- **Multiplexing**: Creating redundant copies of database components to minimize media failures' impact.
- **Fast Recovery Area**: Central location for critical database backups.

Backup Strategies and Modes

- **ARCHIVELOG Mode**: Recommended for production environments, offers benefits in recovery flexibility.
- **NOARCHIVELOG Mode**: Limits backup types, not suitable for robust recovery.

Recovery Manager (RMAN)

- **Advantages**: Offers automation, advanced backup options, and compressed incremental backups.
- **Usage**: Accessible via EM Cloud Control or command-line interface for comprehensive backup and recovery tasks.

Instance Failure Recovery Considerations

- **Required Components**: Control files, at least one member of each redo log group, and SYSTEM and UNDO tablespaces' data files.
- **Recovery Time**: Use FAST_START_MTTR_TARGET parameter to specify target recovery time.

Media Failure Recovery with Data Recovery Advisor

- **ARCHIVELOG Mode**: Allows recovery without losing committed transactions.
- **NOARCHIVELOG Mode**: Limited to the last good, cold backup.

RMAN Support for Data Recovery Advisor

- Provides several commands to facilitate recovery and support the Data Recovery Advisor.

Understanding these essentials ensures efficient database management and robust recovery strategies, minimizing downtime and data loss in case of failures.

▼ Review Questions

1. Among the failure events, which is the most serious and may cause data loss?
   A. The loss of an entire redo log-file group but no loss in any other group
   B. The loss of one member of each redo log-file group
   C. The failure of the ARCn background process
   D. The failure of the LGWR background process
2. When the database is in ARCHIVELOG mode, database recovery is possible up to which event or time?

   A. The last redo log file switch
   B. The last checkpoint position
   C. The last commit
   D. The last incremental backup using RMAN
3. Which is a true statement regarding image copies and backup sets.

   A. An image copy stores one data file per image copy, and a backup set can store many data files in a single file.
   B. An image copy stores one data file per image copy, and a backup set consists of one file per data file backed up.
   C. Both image copies and backup sets use a single file to store all objects to be backed up.

D. A backup set stores each data file in its own backup file, but an image copy places all data files into a single output file.

4. Which of the following is not a step in configuring your database to archive redo log files?

   A. Place the database in ARCHIVELOG mode.
   B. Multiplex the online redo log files.
   C. Specify a destination for archived redo log files.
   D. Specify a naming convention for your archived redo log files.

5. Why are online backups known as inconsistent backups?
   A. Because not all control files are synchronized to the same SCN until the database is shut down
   B. Because both committed and uncommitted transactions are included in a backup when the database is online
   C. Because a database failure while an online backup is in progress can leave the database in an inconsistent state
   D. Because online backups make copies of data files while they are not consistent with the control file

6. Which of the following initialization parameters specifies the location where the

   control file trace backup is sent?
   A. DIAGNOSTIC_DEST
   B. BACKGROUND_DUMP_DEST
   C. LOG_ARCHIVE_DEST
   D. CORE_DUMP_DEST

7. Which of the following pieces of information is not available in the control file?

   A. Instance name
   B. Database name
   C. Tablespace names
   D. Log sequence number

8. Which statement adds a member /logs/redo22.log to redo log-file group 2?

   A. ALTER DATABASE ADD LOGFILE '/logs/redo22.log' TO GROUP 2;
   B. ALTER DATABASE ADD LOGFILE MEMBER '/logs/redo22.log' TO GROUP 2;
   C. ALTER DATABASE ADD MEMBER '/logs/redo22.log' TO GROUP 2;
   D. ALTER DATABASE ADD LOGFILE '/logs/redo22.log';

9. To place the database into ARCHIVELOG mode, in which state must you start the database?

   A. MOUNT
   B. NOMOUNT
   C. OPEN
   D. SHUTDOWN
   E. Any of the above

10. Which of the following substitution-variable formats are always required for specifying the names of the archived redo log files? Choose all that apply.

    A. %d
    B. %s
    C. %r
    D. %t

11. Which of the following initialization parameters controls the mean time to recover the database, in seconds, after an instance failure?

    A. FAST_START_IO_TARGET
    B. LOG_CHECKPOINT_TIMEOUT
    C. FAST_START_MTTR_TARGET
    D. MTTR_TARGET_ADVICE
    E. FAST_START_TARGET_MTTR

12. Identify the statement that is not true regarding the loss of a control file.

    A. A damaged control file can be repaired by using one of the remaining undamaged control files, assuming there are at least two copies of the

control file.
B. The missing or damaged control file can be replaced while the instance is still active.
C. You can temporarily run the instance with one fewer control file, as long as you remove one of the references to the missing control file in the spfile or init.ora file.
D. An instance typically fails when one of the multiplexed control files is lost or damaged.

13. Which failures can be detected by the Data Recovery Advisor, which then provides repair recommendations? Choose all that apply.

    A. Instance failure
    B. Accidental deletion of a data file
    C. Disk containing one redo log member is offline
    D. User accidentally dropped a table

14. The instance can still be started even if some data files are missing; this rule does not apply to which tablespaces? (Choose all that apply.)

    A. USERS
    B. SYSTEM
    C. TEMP
    D. SYSAUX
    E. UNDO

15. Select the statement that is not true regarding media failure. A media failure occurs when:

    A. The network card on the server fails.
    B. The DBA accidentally deletes one of the data files for the SYSTEM tablespace.
    C. There is a head crash on all physical drives in the RAID controller box.
    D. A corrupted track on a CD containing a read-only tablespace causes a query to fail.

16. Choose the correct statement about the Data Recovery Advisor.

A. The Data Recovery Advisor is a standalone tool.
B. The Data Recovery Advisor does not support RAC databases.
C. The CHANGE FAILURE command can be used in a SQL*Plus session.
D. The REPAIR FAILURE command works only after LIST FAILURE.

17. Place the following events or actions leading up to and during instance recovery in the correct order.

    a. The database is opened and available.

    b. Oracle uses undo segments in the undo tablespace to roll back uncommitted trans- actions.

    c. The DBA issues the STARTUP command at the SQL*Plus prompt.

    d. Oracle applies the information in the online redo log files to the data files.
    A. 4, 3, 2, 1
    B. 3, 4, 1, 2
    C. 2, 1, 3, 4
    D. 2, 1, 4, 3
    E. 3, 2, 4, 1
    F. 3, 4, 2, 1

18. You've noticed that when an instance crashes, it takes a long time to start up the data- base. Which advisor can be used to tune this situation?

A. The Undo Advisor
B. The SQL Tuning Advisor
C. The Database Tuning Advisor
D. The MTTR Advisor
E. The Instance Tuning Advisor

19. In ARCHIVELOG mode, the loss of a data file for any tablespace other than the SYSTEM or UNDO tablespace affects which objects in the database?

A. The loss affects only objects whose extents reside in the lost data file.
B. The loss affects only the objects in the affected tablespace, and work can continue in other tablespaces.
C. The loss will not abort the instance but will prevent other transactions in

any tablespace other than SYSTEM or UNDO until the affected tablespace is recovered.
D. The loss affects only those users whose default tablespace contains the lost or damaged data file.

20. Which of the following conditions prevents the instance from progressing through the NOMOUNT, MOUNT, and OPEN states?

    A. One of the redo log-file groups is missing a member.
    B. The instance was previously shut down uncleanly with SHUTDOWN ABORT.
    C. Either the spfile or init.ora file is missing.
    D. One of the five multiplexed control files is damaged.
    E. The USERS tablespace is offline, with one of its data files deleted.

▼ Answers

1. A. Losing an entire redo log-file group can result in losing committed transactions that

may not yet have been written to the database files. Losing all members of a redo log-
file group except for one does not affect database operation and does not result in data

loss. A message is placed in the alert log file. The failure of LGWR causes an instance
failure, but you do not lose any committed transaction data. When an ARCn process
fails or is manually terminated, Oracle spawns another one.

1. C. In ARCHIVELOG mode, recovering the database is possible up to the last COMMIT state-
   ment; in other words, no committed transactions are lost in ARCHIVELOG mode.

Download from Join eBook (www.joinebook.com)

Chapter 15: Using Backup and Recovery 1111

1. A. Image copies are duplicate data and log files in OS format. Backup sets are binary
   compressed files in Oracle proprietary format. In addition to storing multiple data files
   in a single output file, backup sets do not contain unused blocks.
2. B. Although it is recommended that you multiplex your online redo log files, it is not
   required to enable ARCHIVELOG mode of the database.
3. D. During an online backup, even if all data files are backed up at the same time, they
   are rarely, if ever, in sync with the control file.
4. A. The trace backup is created in a subdirectory under the location specified by the
   DIAGNOSTIC_DEST parameter—
   $DIAGNOSTIC_DEST/diag/<dbname>/<instancename>/
   trace directory.
5. A. The instance name is not in the control file. The control file has information about
   the physical database structure.
6. B. When adding log-file members, specify the group number or specify all the existing
   group members.
7. A. To put the database into ARCHIVELOG mode, the database must be in the MOUNT state;
   the control files and all data files that are not offline must be available to change the
   database to ARCHIVELOG mode.
8. B, C, D. The substitution variable %d, which represents the database ID, is required
   only if multiple databases share the same archive log destination.
9. C. The parameter FAST_START_MTTR_TARGET specifies the desired time, in seconds,

to recover a single instance from a crash or instance failure. The parameters LOG_
CHECKPOINT_TIMEOUT and FAST_START_IO_TARGET can still be used in Oracle 12c but
should be used only together with an advanced-tuning scenario or for compatibility
with older versions of Oracle. MTTR_TARGET_ADVICE and FAST_START_TARGET_MTTR
are not valid initialization parameters.

10. B. The instance must be shut down, if it is not already down, to repair or replace the
missing or damaged control file.
11. B, C. Media failure, physical corruption, logical corruption, and missing data files all
can be identified by the Data Recovery Advisor, which also provides recommendations
for repair.
12. B, E. If a tablespace is taken offline because a data file is missing, the instance can
still be started as long as the missing data file does not belong to the SYSTEM or UNDO
tablespace.
13. A. If a network card fails, the failure type is network; the actual media containing the
database files are not affected.

1. B. The Data Recovery Advisor in Oracle Database 12c Release 1 does not support
RAC databases. It is integrated with EM Cloud Control and with RMAN. CHANGE
FAILURE and other commands can be executed using RMAN. The ADVISE

FAILURE
command must be run before you can perform REPAIR FAILURE.

2. B. Instance recovery, also known as crash recovery, occurs when the DBA attempts to

open the database but the files were not synchronized to the same SCN when the data-
base was shut down. Once the DBA issues the STARTUP command, Oracle uses informa-
tion in the redo log files to restore the data files (including the undo tablespace’s data

files) to the state before the instance failure. Oracle then uses undo data in the undo
tablespace after the database has been opened and made available to users to roll back
uncommitted transactions.
18. D. The MTTR Advisor can tell the DBA the most effective value for the FAST_START_

MTTR_TARGET parameter. This parameter specifies the maximum time required in sec-
onds to perform instance recovery.

1. B. The loss of one or more of a tablespace’s data files does not prevent other users from
doing their work in other tablespaces. Recovering the affected data files can continue
while the database is still online and available.

2. D. All copies of the control files, as defined in the spfile or the init.ora file, must be

identical and available. If one of the redo log-file groups is missing a member, a warn-
ing is recorded in the alert log, but instance startup still proceeds. If the instance was

previously shut down with SHUTDOWN ABORT, instance recovery automatically occurs

during startup. Only an spfile or an init.ora file is needed to enter the
NOMOUNT state,
not both. If a tablespace is offline, the status of its data files is not checked
until an
attempt is made to bring it online; therefore, it will not prevent instance startup.