2024

Oracle 19c Database Administration

DAY 4

Day 4 Topics

Backup and Recovery Configuration

Configuring for Recoverability

1. Enabling Archivelog Mode
Enabling Archivelog mode is crucial for performing complete database recovery. In Archivelog mode, Oracle keeps a copy of each redo log file when it is filled, allowing you to recover the database to any point in time.

Steps to Enable Archivelog Mode:
Shut Down the Database:

```
SHUTDOWN IMMEDIATE;
```

Start the Database in Mount Mode:

```
STARTUP MOUNT;
```

Enable Archivelog Mode:

```
ALTER DATABASE ARCHIVELOG;
```

Open the Database:

```
ALTER DATABASE OPEN;
```

Verify Archivelog Mode:

```
ARCHIVE LOG LIST;
```

2. Configuring Archive Log Destinations
You need to specify the location where the archived redo log files will be stored.

```
ALTER SYSTEM SET log_archive_dest_1='LOCATION=/u01/app/oracle/archivelog' SCOPE=BOTH;
```

3. Setting Initialization Parameters
Several initialization parameters need to be configured to ensure proper recoverability.

Example:

```
ALTER SYSTEM SET db_recovery_file_dest='/u01/app/oracle/fast_recovery_area' SCOPE=BOTH;
ALTER SYSTEM SET db_recovery_file_dest_size=10G SCOPE=BOTH;
ALTER SYSTEM SET db_flashback_retention_target=1440 SCOPE=BOTH; -- Retention period in minutes
```

Redo Log Files

Redo log files are crucial for the recovery process, as they store all changes made to the database. Properly configuring and managing redo log files is essential for ensuring database recoverability.

1. Creating and Managing Redo Log Groups

Creating Redo Log Groups:
To create additional redo log groups, use the ALTER DATABASE statement.

```
ALTER DATABASE ADD LOGFILE GROUP 4 ('/u01/app/oracle/oradata/redo04.log') SIZE 50M;
ALTER DATABASE ADD LOGFILE GROUP 5 ('/u01/app/oracle/oradata/redo05.log') SIZE 50M;
```

Dropping Redo Log Groups:
To drop a redo log group, ensure it is not currently active.

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

2. Adding Redo Log Members
Adding multiple members to each redo log group provides redundancy. This ensures that if one redo log member is corrupted, the database can still be recovered using the other members.

```
ALTER DATABASE ADD LOGFILE MEMBER '/u01/app/oracle/oradata/redo04b.log' TO GROUP 4;
ALTER DATABASE ADD LOGFILE MEMBER '/u01/app/oracle/oradata/redo05b.log' TO GROUP 5;
```

3. Resizing Redo Log Files
If redo log files are too small, they may switch too frequently, leading to performance issues. Conversely, if they are too large, recovery time may increase.

Steps to Resize Redo Log Files:
Add New Redo Log Files with Desired Size:

```
ALTER DATABASE ADD LOGFILE GROUP 6 ('/u01/app/oracle/oradata/redo06.log') SIZE 100M;
ALTER DATABASE ADD LOGFILE GROUP 7 ('/u01/app/oracle/oradata/redo07.log') SIZE 100M;
```

Drop the Old Redo Log Files:

```
ALTER DATABASE DROP LOGFILE GROUP 1;
ALTER DATABASE DROP LOGFILE GROUP 2;
```

4. Monitoring Redo Log Activity
You can monitor redo log activity using dynamic performance views.

```
SELECT group#, members, bytes, status FROM v$log;
SELECT member, group#, type, is_recovery_dest_file FROM v$logfile;
```

Example Workflow for Configuring Recoverability and Managing Redo Log Files

```
--Enable Archivelog Mode:
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
ARCHIVE LOG LIST;

--Configure Archive Log Destination:
ALTER SYSTEM SET log_archive_dest_1='LOCATION=/u01/app/oracle/archivelog' SCOPE=BOTH;

--Set Initialization Parameters:
ALTER SYSTEM SET db_recovery_file_dest='/u01/app/oracle/fast_recovery_area' SCOPE=BOTH;
ALTER SYSTEM SET db_recovery_file_dest_size=10G SCOPE=BOTH;
ALTER SYSTEM SET db_flashback_retention_target=1440 SCOPE=BOTH;

--Create Redo Log Groups and Members:
ALTER DATABASE ADD LOGFILE GROUP 4 ('/u01/app/oracle/oradata/redo04.log') SIZE 50M;
ALTER DATABASE ADD LOGFILE MEMBER '/u01/app/oracle/oradata/redo04b.log' TO GROUP 4;
```

```
--Monitor Redo Log Activity:
SELECT group#, members, bytes, status FROM v$log;
SELECT member, group#, type, is_recovery_dest_file FROM v$logfile;

--Perform Regular Backups:
rman target /
RUN {
  BACKUP DATABASE PLUS ARCHIVELOG;
  BACKUP CURRENT CONTROLFILE;
  BACKUP SPFILE;
}
```

Creating Database Backups

RMAN Backup

Types of RMAN Backups

- Full Backups: Capture the entire contents of the database.
- Incremental Backups: Capture only the changes made since the last backup.
- Differential Incremental Backup: Backs up all blocks changed since the last level 1 or level 0 backup.
- Cumulative Incremental Backup: Backs up all blocks changed since the last level 0 backup.
- Archivelog Backups: Back up archived redo logs to ensure all transactions are recoverable.

Key RMAN Commands

- BACKUP: Creates backups of the database, tablespaces, datafiles, control files, SPFILE, and archived redo logs.
- CONFIGURE: Sets persistent configurations for backup and recovery settings.
- SHOW: Displays the current RMAN configuration settings.
- RUN: Executes a series of RMAN commands within a block.

RMAN Backup Examples

1. Configuring RMAN for Backups

Before performing backups, configure RMAN with necessary settings like the default backup device type and retention policy.

Example: Configuring RMAN Settings

```
rman target /

-- Set the default device type to disk
CONFIGURE DEFAULT DEVICE TYPE TO DISK;

-- Set the retention policy to keep backups for 7 days
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;

-- Enable backup optimization
CONFIGURE BACKUP OPTIMIZATION ON;
```

```
-- Configure the control file autobackup
CONFIGURE CONTROLFILE AUTOBACKUP ON;

-- Set the backup location
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/u01/app/oracle/backup/%U';
```

2. Performing a Full Database Backup
A full database backup captures the entire contents of the database.

Example: Full Database Backup

```
rman target /

BACKUP DATABASE PLUS ARCHIVELOG;
```

3. Performing Incremental Backups
Incremental backups can be used to reduce the amount of data backed up by only capturing changes.

Example: Level 0 Incremental Backup

```
rman target /

BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

Example: Level 1 Incremental Backup

```
rman target /

BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

4. Backing Up Archived Redo Logs
Backing up archived redo logs ensures that all transactions are recoverable.

Example: Archivelog Backup

```
rman target /

BACKUP ARCHIVELOG ALL;
```

Backing Up Databases on DBCS

On DBCS, Oracle provides automated tools and utilities for managing backups.

1. Automated Backups
DBCS offers automated daily backups that can be configured through the Oracle Cloud console.

Configuring Automated Backups:

1) Log in to the Oracle Cloud Console:
    - Navigate to the Oracle Cloud Infrastructure dashboard.

2) Select the Database Instance:
    - Go to the "Autonomous Database" or "DB Systems" section.

3) Configure Backup Settings:
    - Set the backup retention period and specify the backup window.
    - Enable automatic backups if not already enabled.

2. Manual Backups Using RMAN on DBCS

In addition to automated backups, you can perform manual backups using RMAN.

Example: Manual Full Database Backup on DBCS

1) SSH into the DBCS VM:
   Use SSH to connect to your DBCS VM.

2) Start RMAN and Connect to the Target Database:

```
ssh -i <private_key> opc@<DBCS_VM_IP>
sudo su - oracle
rman target /
```

3) Perform the Backup:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

Example Workflow for RMAN Backups on DBCS

```
--SSH into the DBCS VM:
ssh -i <private_key> opc@<DBCS_VM_IP>
sudo su - oracle

--Start RMAN and Configure Settings:
rman target /

CONFIGURE DEFAULT DEVICE TYPE TO DISK;
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/u01/app/oracle/backup/%U';

--Perform a Full Database Backup:
BACKUP DATABASE PLUS ARCHIVELOG;

--Perform an Incremental Backup:
BACKUP INCREMENTAL LEVEL 1 DATABASE;

--Back Up Archived Redo Logs:
BACKUP ARCHIVELOG ALL;
```

Performing Database Recovery

Data Recovery Advisor

The Data Recovery Advisor is an Oracle tool that helps diagnose and repair data failures. It provides recommendations and automated scripts to fix the issues.

Usage:
Start RMAN and Connect to the Target Database:

```
rman target /
```

List Failures:

```
LIST FAILURE;
```

Advise on Failures:

```
ADVISE FAILURE;
```

Execute Repair Script:

```
REPAIR FAILURE;
```

Loss of a Control File

Control files are crucial for database operation, containing metadata about the database structure. Loss of all control files can prevent the database from starting.

Recovery Steps:
If the control file is lost, Oracle will indicate it during the startup process.

Use RMAN to restore the control file from a backup.

```
--Shutdown the Database:
SHUTDOWN IMMEDIATE;

--Start RMAN and Connect to the Target Database:
rman target /

--Restore the Control File:
RESTORE CONTROLFILE FROM '/backup/location/controlfile_backup.ctl';

--Mount the Database:
ALTER DATABASE MOUNT;

--Recover the Database:
RECOVER DATABASE;

--Open the Database:
ALTER DATABASE OPEN RESETLOGS;
```

Loss of a Redo Log File

Redo log files are critical for maintaining transaction integrity. Loss of a redo log file can impact the database's ability to recover from crashes.

Recovery Steps:
Oracle will indicate the loss of a redo log file during operation or startup.

Use SQL commands to drop and recreate the redo log file if it's missing or corrupted.

```
--Identify the Corrupted Redo Log Group:
SELECT group#, status FROM v$log;

--Drop the Corrupted Redo Log Group:
ALTER DATABASE DROP LOGFILE GROUP 3;

--Recreate the Redo Log Group:
ALTER DATABASE ADD LOGFILE GROUP 3 ('/path/to/redo03.log') SIZE 50M;

--Switch Log Files:
ALTER SYSTEM SWITCH LOGFILE;
```

DBCS: Performing Recovery by Using the Console

In Oracle Database Cloud Service (DBCS), the Oracle Cloud Console provides tools to perform database recovery tasks.

Steps:

1) Log in to the Oracle Cloud Console:
    - Navigate to the Oracle Cloud Infrastructure dashboard.

2) Select the Database Instance:
    - Go to the "Autonomous Database" or "DB Systems" section.

3) Perform Recovery:
    - Use the available options to restore from backups or recover specific files.

dbaascli Utility

The dbaascli utility is a command-line tool provided by Oracle Cloud for managing Oracle Database Cloud Service instances, including performing backup and recovery tasks.

Key Commands:

Backing Up the Database:

```
dbaascli backup db --target=DBAAS --type=FULL
```

Restoring the Database:

```
dbaascli restore db --bckid=backup_id
```

Listing Backups:

```
dbaascli list backups
```

Example: Using dbaascli for Recovery

SSH into the DBCS VM:

```
ssh -i <private_key> opc@<DBCS_VM_IP>
sudo su - oracle
```

List Available Backups:

```
dbaascli list backups
```

Restore the Database from a Specific Backup:

```
dbaascli restore db --bckid=backup_id
```

Monitoring and Tuning Database Performance

Automatic Workload Repository (AWR)

AWR is a built-in repository in Oracle Database that collects, processes, and maintains performance statistics. It provides a historical view of database performance and is the foundation for many Oracle performance tuning tools.

Key Features:

- ✓ Snapshots: AWR collects performance data snapshots at regular intervals (default is every hour).
- ✓ Reports: Generate AWR reports to analyze performance over specific periods.
- ✓ Baselines: Create baselines to compare current performance against historical performance.

Usage:

Generating AWR Reports:

Using SQL*Plus:

```
-- Connect to the database as SYSDBA
sqlplus / as sysdba

-- Generate an AWR report for a specific time period
@?/rdbms/admin/awrrpt.sql
```

Using Oracle Enterprise Manager (OEM):

1) Navigate to Performance > AWR > AWR Reports.
2) Select the time period and generate the report.

Managing AWR Snapshots:

List Snapshots:

```
SELECT snap_id, begin_interval_time, end_interval_time
FROM dba_hist_snapshot
ORDER BY snap_id;
```

Create a Manual Snapshot:

```
EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT();
```

Creating Baselines:

Create a Baseline:

```
EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE(
  start_snap_id => 100,
  end_snap_id => 110,
  baseline_name => 'MY_BASELINE');
```

Automatic Database Diagnostic Monitor (ADDM)

ADDM analyzes AWR data to identify performance issues and provide recommendations for improvement. It runs automatically after each AWR snapshot is taken.

Key Features:

- ✓ Findings: Provides detailed findings on performance issues.
- ✓ Recommendations: Suggests actionable recommendations to address identified issues.
- ✓ Impact Analysis: Shows the potential impact of recommendations.

Usage:

Running ADDM Analysis:

Using SQL*Plus:

```
-- Connect to the database as SYSDBA
sqlplus / as sysdba

-- Generate an ADDM report for a specific time period
@?/rdbms/admin/addmrpt.sql
```

Using Oracle Enterprise Manager (OEM):

1) Navigate to Performance > Advisors Home > ADDM.
2) View the findings and recommendations for the selected time period.

Viewing ADDM Findings and Recommendations:

Query Findings:

```
SELECT dbms_addm.get_report(
  dbid => 123456789,
  begin_snap_id => 100,
  end_snap_id => 110) AS report
FROM dual;
```

Advisory Framework

The advisory framework in Oracle Database provides a set of advisors that help you optimize various aspects of database performance, including memory, storage, and SQL execution.

Key Advisors:

- ✓ SQL Tuning Advisor: Analyzes SQL statements and provides recommendations for improving their performance.
- ✓ SQL Access Advisor: Recommends the creation of indexes and materialized views to optimize SQL queries.
- ✓ Memory Advisors: Includes SGA and PGA advisors to help allocate memory efficiently.

Usage:
SQL Tuning Advisor:

Using SQL*Plus:

```
-- Connect to the database as SYSDBA
sqlplus / as sysdba

-- Execute SQL Tuning Advisor for a specific SQL statement
EXEC DBMS_SQLTUNE.CREATE_TUNING_TASK(
  sql_text => 'SELECT * FROM employees',
  user_name => 'HR',
  task_name => 'sql_tuning_task');

EXEC DBMS_SQLTUNE.EXECUTE_TUNING_TASK(
  task_name => 'sql_tuning_task');

SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('sql_tuning_task') AS report
FROM dual;
```

Using Oracle Enterprise Manager (OEM):

1) Navigate to Performance > Advisors Home > SQL Tuning Advisor.
2) Select the SQL statement and generate recommendations.

SQL Access Advisor:
Using SQL*Plus:

```
-- Create a workload for SQL Access Advisor
EXEC DBMS_ADVISOR.CREATE_TASK(
  advisor_name => 'SQL Access Advisor',
  task_name => 'access_advisor_task');

EXEC DBMS_ADVISOR.ADD_SQLWKLD_SQLSET(
  task_name => 'access_advisor_task',
  sqlset_name => 'my_sqlset');

EXEC DBMS_ADVISOR.EXECUTE_TASK(
  task_name => 'access_advisor_task');

SELECT DBMS_ADVISOR.GET_TASK_REPORT('access_advisor_task') AS report
```

```
FROM dual;
```

Memory Advisors:

Using SQL*Plus:

```
-- Query SGA Target Advice
SELECT size_mb, size_factor, estd_db_time, estd_physical_reads
FROM v$sga_target_advice;

-- Query PGA Target Advice
SELECT pga_target_for_estimate, pga_target_factor, estd_extra_bytes_rw
FROM v$pga_target_advice;
```

Tuning SQL

Tuning SQL involves optimizing SQL statements to improve their performance. This includes analyzing execution plans, creating indexes, rewriting queries, and gathering statistics.

Steps for SQL Tuning:

Analyze Execution Plans:

Using SQL*Plus:

```
-- Display the execution plan for a SQL statement
EXPLAIN PLAN FOR
SELECT * FROM employees WHERE department_id = 10;

SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

Gather Optimizer Statistics:

Using SQL*Plus:

```
-- Gather statistics for a table
EXEC DBMS_STATS.GATHER_TABLE_STATS('HR', 'EMPLOYEES');
```

Create Indexes:

Using SQL*Plus:

```
-- Create an index to optimize a query
CREATE INDEX emp_dept_idx ON employees(department_id);
```

Rewrite Queries:

Example: Using Subquery Factoring:

```
-- Use WITH clause to simplify and optimize a query
WITH dept_emps AS (
  SELECT department_id, COUNT(*) AS emp_count
  FROM employees
  GROUP BY department_id
)
SELECT * FROM dept_emps WHERE emp_count > 5;
```

Example Workflow for Performance Monitoring and Tuning

```
--Generate an AWR Report:
sqlplus / as sysdba
@?/rdbms/admin/awrrpt.sql

--Run ADDM Analysis:
sqlplus / as sysdba
@?/rdbms/admin/addmrpt.sql

--Use SQL Tuning Advisor:
EXEC DBMS_SQLTUNE.CREATE_TUNING_TASK(
  sql_text => 'SELECT * FROM employees',
  user_name => 'HR',
  task_name => 'sql_tuning_task');

EXEC DBMS_SQLTUNE.EXECUTE_TUNING_TASK(
  task_name => 'sql_tuning_task');

SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('sql_tuning_task') AS report
FROM dual;

--Gather Optimizer Statistics:
EXEC DBMS_STATS.GATHER_TABLE_STATS('HR', 'EMPLOYEES');

--Analyze Execution Plan:
EXPLAIN PLAN FOR
SELECT * FROM employees WHERE department_id = 10;

SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

SQL Tuning Process

Oracle Optimizer

The Oracle Optimizer is a key component of the database that determines the most efficient way to execute a SQL query. It considers various factors, such as available indexes, data distribution, and query conditions, to generate an optimal execution plan.

Key Components:

- ✓ Cost-Based Optimizer (CBO): Uses statistics to estimate the cost of different execution plans and chooses the one with the lowest cost.
- ✓ Rule-Based Optimizer (RBO): An older method that uses a fixed set of rules to determine the execution plan. (Deprecated and not recommended for use).

Optimization Phases:

a) Parsing: The SQL statement is parsed, and syntax and semantic checks are performed.
b) Optimization: The optimizer generates multiple execution plans and chooses the one with the lowest cost.

c) Row Source Generation: The chosen execution plan is converted into a set of row sources that produce the result set.
d) Execution: The execution engine executes the plan and returns the results.

Optimizer Statistics

Optimizer statistics are metadata about database objects (tables, indexes, columns) that the optimizer uses to estimate the cost of various execution plans. Accurate statistics are crucial for the optimizer to make the best decisions.

Types of Statistics:

- Table Statistics: Information about the number of rows, number of blocks, average row length, etc.
- Column Statistics: Information about data distribution, such as histograms, distinct values, and nulls.
- Index Statistics: Information about index selectivity, clustering factor, and leaf blocks.

Gathering Statistics:
Statistics can be gathered manually or automatically using the DBMS_STATS package. Oracle automatically gathers statistics for database objects during maintenance windows.

Example:

```
-- Gather statistics for a specific table
EXEC DBMS_STATS.GATHER_TABLE_STATS('HR', 'EMPLOYEES');

-- Gather statistics for the entire schema
EXEC DBMS_STATS.GATHER_SCHEMA_STATS('HR');

-- Gather statistics for the entire database
EXEC DBMS_STATS.GATHER_DATABASE_STATS;
```

SQL Execution Plan

A SQL execution plan (also known as an execution plan or query plan) is a detailed roadmap of how the database will execute a SQL query. It shows the sequence of operations (such as table scans, index scans, joins) and the flow of data through these operations.

Viewing Execution Plans:
Execution plans can be viewed using the EXPLAIN PLAN statement, the DBMS_XPLAN package, or Oracle Enterprise Manager (OEM).

Example:

```
-- Explain the execution plan for a SQL statement
EXPLAIN PLAN FOR
SELECT * FROM employees WHERE department_id = 10;

-- Display the execution plan
SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

SQL Tuning Steps

1) Identify Problematic SQL Statements:
    - Use tools like AWR, ADDM, and SQL Trace to identify SQL statements that are consuming significant resources.

2) Generate and Analyze Execution Plans:
    - Use EXPLAIN PLAN and DBMS_XPLAN to generate and analyze the execution plans of problematic SQL statements.

3) Gather Optimizer Statistics:
    - Ensure that accurate statistics are gathered for all relevant database objects.

4) Use SQL Tuning Advisor:
    - Use the SQL Tuning Advisor to get recommendations on how to improve the performance of SQL statements.

5) Apply Recommended Changes:
    - Implement the recommendations provided by the SQL Tuning Advisor, such as creating indexes or rewriting SQL statements.

Example Workflow for SQL Tuning

Identify Problematic SQL Statements:

```
-- Use AWR to identify high-resource-consuming SQL statements
SELECT * FROM dba_hist_sqlstat WHERE executions > 1000 ORDER BY buffer_gets DESC;
```

Generate and Analyze Execution Plans:

```
-- Explain the execution plan for a SQL statement
EXPLAIN PLAN FOR
SELECT * FROM employees WHERE department_id = 10;

-- Display the execution plan
SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

Gather Optimizer Statistics:

```
-- Gather statistics for the EMPLOYEES table
EXEC DBMS_STATS.GATHER_TABLE_STATS('HR', 'EMPLOYEES');
```

Use SQL Tuning Advisor:

```
-- Create a tuning task for a specific SQL statement
EXEC DBMS_SQLTUNE.CREATE_TUNING_TASK(
  sql_text => 'SELECT * FROM employees WHERE department_id = 10',
  user_name => 'HR',
  task_name => 'sql_tuning_task');

-- Execute the tuning task
EXEC DBMS_SQLTUNE.EXECUTE_TUNING_TASK(task_name => 'sql_tuning_task');
```

```
-- Get the tuning recommendations
SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('sql_tuning_task') AS report
FROM dual;
```

Apply Recommended Changes:

```
-- Create an index based on the SQL Tuning Advisor recommendation
CREATE INDEX emp_dept_idx ON employees(department_id);
```