Week 1: Fundamentals (Recap)

Day 1: Introduction to Oracle Database Basics

Welcome to Day 1 of our 30-day Oracle DBA Journey!
Let’s start with the basics:

- What exactly is Oracle Database Administration (DBA)?
An Oracle Database Administrator (DBA) is a crucial role in managing and maintaining Oracle databases—whether it’s for a small business or a large enterprise.

- Key Responsibilities of an Oracle DBA:

1) Database Installation and Configuration: Oracle DBAs install and configure Oracle Database software on various systems. They ensure the database is properly set up for development, testing, and production environments.

2) Backup and Recovery: The DBA is responsible for ensuring that data is backed up regularly and can be recovered quickly in case of failures. They use tools like RMAN (Recovery Manager) to automate this critical task.

3) Performance Tuning: It’s the DBA's job to make sure the database runs efficiently. This can involve optimizing SQL queries, configuring the system's memory, and managing disk I/O to reduce bottlenecks.

4) Security Management: DBAs are also responsible for safeguarding the database. They manage user access, roles, and permissions to ensure that only authorized users can access sensitive data.

5) Monitoring and Maintenance: Regularly monitoring the database for potential issues is key to maintaining optimal performance. DBAs use various Oracle tools and logs to track the health of the system.

6) Patching and Upgrading: Keeping the Oracle Database up-to-date with the latest patches and upgrades is critical for both performance and security.

- Why is the Role of a DBA Important?
  In today’s digital world, databases store everything from customer information to financial records. Oracle DBAs are the gatekeepers of this critical data. Without them, organizations could face system downtime, data loss, or even security breaches.

- Top Skills for an Oracle DBA:

  Technical Knowledge:

  ✓Understanding of Oracle architecture, SQL, PL/SQL, and database backup and recovery processes.
  ✓Problem-Solving: Quickly diagnosing and fixing database issues to minimize downtime.\
  ✓Attention to Detail: Precision in managing data structures, security settings, and recovery plans.
  ✓Continuous Learning: Staying up to date with new Oracle releases, features, and cloud technologies.

Day 2: Installation of Oracle Database on Linux

Step-by-Step Guide to Install Oracle 19c RAC

1. Hardware Requirements
    1. Minimum of two nodes (servers) for RAC.
    2. Shared storage (e.g., Oracle ASM, NFS).
    3. Sufficient memory and CPU resources for each node.
2. Software Requirements
    1. Oracle Linux 7 or 8 (or any supported Linux distribution).
    2. Oracle 19c software (downloaded from Oracle's website).
    3. Oracle Grid Infrastructure (GI) software.
3. Networking Requirements
    1. Ensure proper hostname resolution (use /etc/hosts or DNS).
    2. Configure private and public IP addresses for each node.
    3. Enable SSH between nodes without password prompts.

Step 1: Prepare the Environment

a) Create Users and Groups

```
groupadd oinstall
groupadd dba
useradd -g oinstall -G dba oracle
passwd oracle
```

b) Create Directories for Oracle Software

```
mkdir -p /u01/app/oracle
mkdir -p /u02/oracle/grid
chown -R oracle:oinstall /u01/app/oracle
chown -R oracle:oinstall /u02/oracle
```

c) Set Kernel Parameters Edit the /etc/sysctl.conf file and add or modify the following parameters:

```
kernel.shmmax = 4294967295
kernel.shmall = 2097152
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
net.ipv4.ip_local_port_range = 1024 65000
net.core.rmem_default = 262144
net.core.rmem_max = 262144
net.core.wmem_default = 262144
net.core.wmem_max = 262144
```

Then apply the changes:

```
sysctl -p
```

d) Set User Limits Edit the /etc/security/limits.conf file and add:

```
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

Step 2: Install Oracle Grid Infrastructure

1. **Log in as the Oracle User**

```
bash
Copy code
su - oracle
```

2. **Unzip the Oracle Software**

```
bash
Copy code
unzip linuxx64_19c_grid.zip -d /u01/app/oracle
```

3. **Run the Installer**

```
bash
Copy code
cd /u01/app/oracle/grid
./gridSetup.sh
```

4. **Follow the Installer Steps**
   - Choose **Configure Oracle Grid Infrastructure for a cluster.**
   - Select your nodes.
   - Configure the private and public interfaces.
   - Choose storage options (e.g., ASM).
   - Complete the installation process.

Step 3: Install Oracle Database

1. **Unzip the Oracle Database Software**

```
bash
Copy code
unzip linuxx64_19c_database.zip -d /u01/app/oracle
```

2. **Run the Installer**

```
bash
Copy code
cd /u01/app/oracle/database
./runInstaller
```

3. **Follow the Installer Steps**

- Choose **Create a Database**.
- Select **Advanced** configuration for RAC.
- Specify database name, global database name, and instance names.
- Choose the RAC database configuration.
- Configure database storage options and tablespaces.
- Complete the installation.

Step 4: Post-Installation Tasks

1. Run Root Scripts After installation, you will be prompted to run two scripts as the root user. Open a terminal and run:

   ```
   bash
   /u01/app/oraInventory/orainstRoot.sh
   /u01/app/oracle/product/19c/root.sh
   ```

2. **Configure Listener** Use the **Oracle Net Configuration Assistant** to configure the listener for your RAC instances.
3. **Create a Database Instance** You can create additional instances for each node using SQL*Plus:

   ```
   sql
   SQL> startup nomount;
   SQL> create database <db_name>
      template <template_name>
      instance <instance_name>;
   ```

4. **Verify the Installation** Use the following command to check the status of your RAC instances:

   ```
   sql
   SQL> SELECT instance_name, status FROM gv$instance;
   ```

5. **Check Oracle Services** Ensure all services are running as expected:

   ```
   bash
   srvctl status database -d <db_name>
   ```

Day 3: Oracle Database Architecture

Welcome to Day 3 of our Oracle DBA Journey! Today, we will explore the architecture of the Oracle Database. Understanding this architecture is important because it helps you grasp how the database works, making it easier to manage, troubleshoot, and optimize.

What is Oracle Database Architecture?

Oracle Database Architecture refers to how different parts of the Oracle Database are organized and how they interact with each other. It consists of two main components:

1. **Memory Structures**
2. **Processes**

Let’s break these down into simpler terms.

1. Memory Structures

Oracle Database uses memory to store data temporarily while it’s being processed. There are two main types of memory areas:

System Global Area (SGA)

- **What is it?** The SGA is a large, shared memory area that stores data and control information for the Oracle database.
- **Key Parts of SGA:**
  - **Database Buffer Cache:**
    - **What it does:** This part holds copies of data blocks that have been read from disk.
    - **Why it matters:** It speeds up the process of reading and writing data because the database can access this cache instead of going to the disk every time.
  - **Shared Pool:**
    - **What it does:** This area stores SQL execution plans (how the database runs queries) and parsed SQL statements (the SQL commands that have been prepared).
    - **Why it matters:** It saves time because the database doesn't have to parse (prepare) the same SQL statements over and over again.
  - **Redo Log Buffer:**
    - **What it does:** This buffer holds information about changes made to the database.
    - **Why it matters:** In case of a failure, this information is crucial for recovering lost data.
  - **Large Pool:**
    - **What it does:** This is an optional area for handling large memory allocations, like backups.
    - **Why it matters:** It helps improve performance when running certain operations.

Program Global Area (PGA)

- **What is it?** The PGA is a private memory area for each user session.
- **What it does:** It is used for operations like sorting data and performing calculations.
- **Why it matters:** The more complex the query (a request for data), the more memory it needs from the PGA, affecting performance.

2. Oracle Processes

Oracle runs various processes (programs) to manage the database. These processes are divided into two main categories:

Background Processes

1. These are automatic processes that run in the background, managing tasks in the database. Key background processes include:
    - **DBWn (Database Writer):**
        1. **What it does:** It writes modified data from the memory (buffer cache) back to the disk.
    - **LGWR (Log Writer):**
        1. **What it does:** It writes the changes recorded in the redo log buffer to the redo log files.
    - **CKPT (Checkpoint):**
        1. **What it does:** It updates the control file and ensures the data in memory and disk is consistent.
    - **SMON (System Monitor):**
        1. **What it does:** It performs recovery when the database starts up after a crash.
    - **PMON (Process Monitor):**
        1. **What it does:** It cleans up after failed processes and makes sure resources are freed up.

User Processes

- These are processes that represent the users (or applications) interacting with the database.
- **What they do:** They execute SQL commands, retrieve data, and handle transactions (changes made to the data).

3. Storage Structures

Oracle Database also has different types of storage structures to keep data organized on disk. Here are the main types:

- ✓ **Datafiles:**
    - **What are they?** These files store actual user data (like tables and indexes).
    - **Why they matter:** They are the primary files in the database where everything is saved.

- ✓ **Control Files:**
  - **What are they?** These files contain important metadata (data about data), like the names and locations of datafiles.
  - **Why they matter:** They are crucial for starting up and recovering the database.
- ✓ **Redo Log Files:**
  - **What are they?** These files keep a record of all changes made to the database.
  - **Why they matter:** They are essential for recovering the database in case of a failure.
- ✓ **Tablespaces:**
  - **What are they?** Tablespaces are logical storage areas that group related data together.

    Why they matter: They help organize datafiles and improve performance.

4. Instance vs. Database

To clarify:

- **Instance:** This refers to the combination of SGA (memory) and background processes that are running. It is responsible for all database operations.
- **Database:** This refers to the physical files (datafiles, control files, redo log files) that store the actual data.

How It All Works Together

- The **Memory Structures (SGA and PGA)** keep frequently accessed data handy for quick retrieval.
- The **Processes** manage tasks like writing data to disk, recovering from failures, and handling user requests.
- The **Storage Structures** keep the data safe and organized on disk.

Day 4: Understanding Oracle's Database Files

Welcome to Day 4 of our Oracle DBA Journey! Today, we'll dive into the various types of files used by Oracle Database. These files are essential for storing, managing, and recovering data. Understanding these files will help you effectively manage the database and ensure data integrity.

Key Oracle Database Files

Oracle Database relies on several types of files to function correctly. Each file type serves a specific purpose in storing data, managing transactions, and maintaining the overall database structure. Here's an overview of the core file types:

1. Datafiles

- **What are they?** Datafiles store the actual user data, such as tables and indexes. Every piece of information in an Oracle database, including user data and metadata, is kept in datafiles.
- **Location:** Datafiles are located on disk and associated with specific tablespaces.
- **Function:** They store data, enabling user queries and data modifications.

2. Control Files

- **What are they?** Control files contain metadata about the database, such as the structure of the database, the status of datafiles and redo log files, and essential information required during startup.
- **Contents:** Control files record the database's name, the timestamp of its creation, and the locations of datafiles and redo log files.
- **Function:** Used during startup and recovery, control files ensure that the database operates correctly.

3. Redo Log Files

- **What are they?** Redo log files are crucial for maintaining data consistency and integrity. They store a log of all changes made to the database.
- **Contents:** They record all database transactions.
- **Function:** These files are essential for data recovery; they help roll forward changes made after the last backup.

4. Archive Log Files

- **What are they?** Archive logs are copies of redo logs that are preserved once a redo log is full and switched out.
- **Contents:** Archived redo logs contain historical redo entries.
- **Function:** They allow for point-in-time recovery and play a critical role in database backup strategies.

5. Parameter File (PFILE) and Server Parameter File (SPFILE)

- **What are they?** These files store configuration settings that define how the database behaves.
    - **PFILE:** A text file containing initialization parameters.
    - **SPFILE:** A binary version of the PFILE used by Oracle to automatically configure the instance.
- **Function:** They control how Oracle starts and operates. SPFILE is used by default, while PFILE can be manually edited.

6. Password File

- **What is it?** The password file contains the credentials for privileged users (such as SYSDBA) who are allowed to perform administrative tasks on the database.
- **Function:** It authenticates database administrators for operations that require elevated privileges.

Understanding the Relationships Between Files

- **Datafiles** hold the actual data.
- **Control files** monitor the database structure and manage metadata.
- **Redo log files** track changes made to the database to ensure recovery options are available.
- **Archive logs** keep a history of transactions for backup and recovery.
- **Parameter and password files** control how the database starts and secure administrative access.

Why These Files Matter

Understanding these files is crucial for an Oracle DBA because they represent the backbone of the database. Proper management of these files ensures:

- **Data integrity:** Ensures data remains accurate and consistent.
- **Performance:** Optimizes how data is stored and accessed.
- **Recoverability:** Provides options to recover data in case of failure.

Day 5: Tablespaces in Oracle Database

Welcome to Day 5 of our Oracle DBA Journey! Today, we will explore **Tablespaces**—a fundamental concept in Oracle Database management. Understanding tablespaces is crucial for efficiently organizing and managing data within the database.

What is a Tablespace?

A **tablespace** is a logical storage unit in an Oracle Database that groups related logical structures such as tables, indexes, and other database objects. It acts as a container for these objects and plays a vital role in data management.

Key Features of Tablespaces

1. **Logical Grouping:**
   - Tablespaces allow you to logically group database objects, making it easier to manage and organize data.
2. **Data File Association:**
   - Each tablespace consists of one or more **datafiles**, which are physical files on disk that store the actual data. When a tablespace is created, you specify the datafiles associated with it.
3. **Data Organization:**
   - Tablespaces help in organizing data based on application requirements. For example, you might create separate tablespaces for different applications or user groups.

Types of Tablespaces

Oracle Database supports several types of tablespaces:

Key Types of Tablespaces:

1) SYSTEM and SYSAUX Tablespaces:
Purpose: Store Oracle's internal data (metadata and system-related objects).
Why They Matter: These are critical for the functioning of the database and should be left for Oracle to manage.

2) User Tablespaces:
Purpose: Used to store user data, such as tables and indexes created by users.
Why They Matter: This is where you, as a DBA, will focus your efforts in managing space for user data.

3) Temporary Tablespaces:
Purpose: Store temporary data needed during operations like sorting or hashing.
Why They Matter: Helps manage intermediate results of queries and other temporary data processing

4)Undo Tablespaces:
Purpose: Used to store undo data, which helps in rolling back transactions and supporting

read consistency.
Why They Matter: Ensures that the database can rollback transactions safely and keeps data consistent.

Creating a Tablespace

Here’s a simple SQL command to create a new tablespace:

```
CREATE TABLESPACE example_tablespace
DATAFILE 'example_tablespace.dbf'
SIZE 100M
AUTOEXTEND ON
NEXT 10M MAXSIZE UNLIMITED;
```

- This command creates a tablespace named **example_tablespace** with a datafile of 100MB that can automatically extend.

Adding a Datafile:

```
SQL>ALTER TABLESPACE users ADD DATAFILE 'users02.dbf' SIZE 100M;
```

Resizing a Datafile:

```
SQL>ALTER DATABASE DATAFILE 'users01.dbf' RESIZE 200M;
```

How Tablespaces Work:
A tablespace is made up of datafiles, which physically store the data on disk.
When you create a table or index, it is assigned to a particular tablespace. Oracle manages the physical storage (datafiles), while you as a DBA manage the logical storage (tablespaces).

Temporary and Undo tablespaces are automatically managed by Oracle but can be customized for performance tuning.

✓ Tablespace Management:

Managing tablespaces involves monitoring the space usage, adding or resizing datafiles, and making sure that there is always enough space for database operations. You can use the following commands to manage tablespaces:

Tablespaces are essential for managing data efficiently in Oracle databases. Tomorrow, we’ll move on to User Management and Security—another key skill for DBAs!

Day 6: User and Privilege Management in Oracle

Welcome to Day 6 of our Oracle DBA Journey! Today, we will explore **User and Privilege Management** in Oracle Database. Proper management of users and their privileges is essential for maintaining security, ensuring data integrity, and controlling access to the database.

What is User Management?

User management involves creating, modifying, and deleting database users. Each user is assigned a unique username and has specific privileges that determine what actions they can perform within the database.

Key Concepts:
1) User Accounts:
Purpose: Users access the database through accounts that are granted specific permissions.
Why It Matters: Each user has defined privileges, limiting what they can do within the database.

2)Roles:
Purpose: A role is a set of privileges that can be assigned to users.
Why It Matters: Instead of assigning individual permissions, you can assign a role that groups related privileges, simplifying user management.

3)Privileges:
System Privileges: Allow users to perform actions at the database level (e.g., create users, alter database).
Object Privileges: Control access to specific database objects like tables, views, or procedures (e.g., select, insert, update).

4)Profiles:
Purpose: Profiles limit resources for users, such as connection time or memory usage.
Why It Matters: Helps prevent one user from monopolizing resources, ensuring fair use of the database.

Creating a User

To create a new user in Oracle, you can use the following SQL command:

```
SQL> CREATE USER username IDENTIFIED BY password;
```

Example:

```
SQL>  CREATE USER AMIT IDENTIFIED BY strongpassword123;
```

Granting Privileges

After creating a user, you need to assign privileges to control what that user can do. Privileges can be categorized into two types:

1. **System Privileges:**
   - Allow users to perform specific actions in the database, such as creating tables, altering sessions, or managing users.
   - **Example of Granting System Privileges:**

```
SQL>  GRANT CREATE SESSION TO AMIT;
SQL>  GRANT CREATE TABLE TO AMIT;
```

2. **Object Privileges:**
   - Allow users to perform actions on specific database objects, like tables or views.
   - **Example of Granting Object Privileges:**

```
SQL>  GRANT SELECT, INSERT, UPDATE ON employees TO AMIT;
```

- In this example, AMIT is granted the ability to select, insert, and update records in the employees table.

Revoking Privileges

If you need to remove privileges from a user, you can use the REVOKE command:

```
SQL>  REVOKE privilege_name ON object_name FROM username;
```

- **Example:**

```
SQL>  REVOKE SELECT ON employees FROM AMIT;
```

- This command removes the SELECT privilege from AMIT for the employees table.

Roles in Oracle Database

Roles are a convenient way to manage privileges for multiple users. Instead of assigning privileges individually, you can create a role, grant it the necessary privileges, and then assign that role to users.

1. **Creating a Role:**

```
SQL>  CREATE ROLE role_name;
```

Example:

```
SQL>  CREATE ROLE data_entry;
```

2. **Granting Privileges to the Role:**

```
SQL>  GRANT SELECT, INSERT, UPDATE ON employees TO data_entry;
```

3. **Assigning the Role to a User:**

```
SQL>  GRANT data_entry TO AMIT;
```

- Now, AMIT has the privileges associated with the data_entry role.

Managing User Profiles

User profiles help control resource limits and security parameters for users. They can define limits on CPU usage, sessions, and password policies.

1. **Creating a Profile:**

```
SQL>CREATE PROFILE profile_name LIMIT
    SESSIONS_PER_USER 5
    CPU_PER_SESSION 10000;
```

Example:

```
SQL>  CREATE PROFILE limited_user_profile LIMIT
    SESSIONS_PER_USER 5
    CPU_PER_SESSION 10000;
```

2. **Assigning a Profile to a User:**

```
SQL>  ALTER USER john_doe PROFILE limited_user_profile;
```

Best Practices for User and Privilege Management

- **Principle of Least Privilege:** Always assign the minimum privileges necessary for users to perform their tasks.
- **Regular Audits:** Periodically review user privileges and roles to ensure they are still appropriate.
- **Use Roles for Management:** Use roles to simplify privilege management across multiple users.
- **Strong Password Policies:** Enforce strong password policies to enhance database security.

Day 7: Managing Oracle Processes – Background and User Processes

Today, we'll dive into the key Oracle processes that run behind the scenes to keep the database operating smoothly. Understanding these processes is essential for managing, troubleshooting, and optimizing Oracle Database.

Oracle Processes Overview

Oracle Database runs several processes, categorized into two types:

1) User Processes: These are initiated by users or applications to interact with the database.

2) Background Processes: These are internal processes Oracle runs automatically to manage database operations.

1. User Processes
User processes interact with the Oracle database on behalf of the end-user or application.
Initiates SQL Queries: Users connect to the database and run SQL commands.
Communicates with Server Processes: User processes communicate with Oracle's server processes to execute SQL, retrieve data, and perform transactions.
Example: When you run a query in an application, it triggers a user process to interact with the database.

2. Background Processes
These are crucial for the proper functioning of the database. They perform tasks such as writing data to disk, recovering the system, and maintaining the database's internal consistency.
Here are the most important background processes:

- DBWn (Database Writer)
  Purpose: Writes modified (dirty) data blocks from memory (Buffer Cache) to disk.
  Role: Helps free up space in the memory cache for new operations.

- LGWR (Log Writer)
  Purpose: Writes redo log entries from the Redo Log Buffer to the online redo log files.
  Role: Ensures that all changes made to the database are recorded for recovery in case of failure.

- SMON (System Monitor)
  Purpose: Responsible for instance recovery after a crash.
  Role: Cleans up temporary segments and coalesces free space in the database.

- PMON (Process Monitor)
  Purpose: Cleans up after failed processes, such as rolling back uncommitted transactions.
  Role: Frees resources held by failed processes and ensures system stability.

- CKPT (Checkpoint Process)

Purpose: Updates control files and datafile headers with the latest checkpoint information.
Role: Reduces recovery time by writing consistent data to disk periodically.

- ARCn (Archiver)
  Purpose: Copies redo log files to archive storage after a log switch.
  Role: Helps with database recovery and backups by keeping historical logs.

✓ **How These Processes Work Together**
User Process: Initiates a query.
Server Process: Executes the query and interacts with background processes.
Background Processes: Ensure data is written to disk, logs are updated, and the database remains healthy.

✓ **Why Are These Processes Important for a DBA?**
Understanding and managing these processes allows you to:
Diagnose performance issues: Like slow writes (DBWn) or redo log waits (LGWR).
Optimize system performance: By configuring parameters related to these processes.
Ensure system reliability: Through monitoring SMON, PMON, and ARCn for any recovery or cleanup tasks.