2024

Oracle 19c Database Administration

DAY 2

Day 2 Topics

Oracle Net Services

Oracle Net Services

Oracle Net Services is a suite of networking components that enable Oracle database clients and servers to communicate with each other. It supports the Transparent Network Substrate (TNS) protocol, which allows for secure and efficient data transport.

Key Components of Oracle Net Services

- ✓ Listener
- ✓ Oracle Net Configuration Files
- ✓ Oracle Net Manager
- ✓ Oracle Net Configuration Assistant

Oracle Net Listener

The Oracle Net Listener is a process that runs on the database server and handles incoming client connection requests. It listens for connection requests on a specified port (default is 1521) and forwards them to the appropriate database instance.

Configuring the Listener
The listener configuration file, listener.ora, is located in the network/admin directory under the Oracle Home directory.

Example listener.ora file:

```
LISTENER =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myserver)(PORT = 1521))
  )
 )

SID_LIST_LISTENER =
 (SID_LIST =
  (SID_DESC =
   (GLOBAL_DBNAME = orcl)
   (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
   (SID_NAME = orcl)
  )
 )
```

Starting and Stopping the Listener
You can start and stop the listener using the lsnrctl command:

```
lsnrctl start
lsnrctl stop
```

Establishing Oracle Network Connections

Oracle Net Configuration Files

- **tnsnames.ora**
  The tnsnames.ora file contains network service names mapped to connect descriptors. This file is used by clients to identify database servers.

Example tnsnames.ora file:

```
ORCL =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myserver)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = orcl)
  )
 )
```

- **sqlnet.ora**
  The sqlnet.ora file contains client-side network configuration parameters. It can specify the default domain, logging options, and other settings.

Example sqlnet.ora file:

```
SQLNET.AUTHENTICATION_SERVICES = (NONE)
NAMES.DIRECTORY_PATH = (TNSNAMES, EZCONNECT)
```

Oracle Net Manager
Oracle Net Manager is a GUI tool that helps you configure and manage Oracle Net Services. You can use it to create and edit listener.ora, tnsnames.ora, and sqlnet.ora files.

To start Oracle Net Manager:

```
netmgr
```

Oracle Net Configuration Assistant (NETCA)
NETCA is a wizard-based tool that simplifies the configuration of network components. It can create and configure listeners, naming methods, net service names, and directory usage.

To start NETCA:

```
netca
```

Connecting to an Oracle Database

Example Workflow

Configure the Listener (listener.ora):

```
LISTENER =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myserver)(PORT = 1521))
  )
 )

SID_LIST_LISTENER =
 (SID_LIST =
  (SID_DESC =
   (GLOBAL_DBNAME = orcl)
   (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
   (SID_NAME = orcl)
  )
 )
```

Start the Listener:

```
lsnrctl start
```

Configure the Client (tnsnames.ora):

```
ORCL =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myserver)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = orcl)
  )
 )
```

Configure the Client (sqlnet.ora):

```
SQLNET.AUTHENTICATION_SERVICES = (NONE)
NAMES.DIRECTORY_PATH = (TNSNAMES, EZCONNECT)
```

Connect to the Database:

```
sqlplus user/password@ORCL
```

Troubleshooting Tips

Check Listener Status:

```
lsnrctl status
```

Check Connectivity:

```
tnsping ORCL
```

Enable Logging:
Add the following to sqlnet.ora:

```
TRACE_LEVEL_CLIENT = 16
```

Review Log Files:
Listener logs can be found in the log directory under the Oracle Home directory:

```
$ORACLE_BASE/diag/tnslsnr/<hostname>/listener/trace/listener.log
```

User Security

Database User Accounts

1. Creating and Managing User Accounts

Creating a User
To create a new user account, you use the CREATE USER statement. You need to specify the username and password, and optionally other settings like default tablespace, temporary tablespace, and profile.

```
CREATE USER username IDENTIFIED BY password
  DEFAULT TABLESPACE tablespace_name
  TEMPORARY TABLESPACE temp_tablespace_name
  PROFILE profile_name;
```

Example:

```
CREATE USER john IDENTIFIED BY securepassword
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  PROFILE default;
```

Viewing All Users
The DBA_USERS view contains information about all users in the database. To query this view, you need DBA privileges.

```
SELECT username, account_status, default_tablespace, created
FROM dba_users;
```

Viewing Users Accessible to the Current User
The ALL_USERS view shows information about all users that the current user has access to.

```
SELECT username, created
FROM all_users;
```

Viewing the Current User's Information
The USER_USERS view provides information about the current user.

```
SELECT username, default_tablespace, created
FROM user_users;
```

Detailed Examples

Example 1: Viewing All Users (Using DBA_USERS)

```
-- Connect as a user with DBA privileges
sqlplus / as sysdba

-- Query the DBA_USERS view
SELECT username, account_status, default_tablespace, created
FROM dba_users
ORDER BY username;
```

Example 2: Viewing Users Accessible to the Current User (Using ALL_USERS)

```
-- Connect as any user
sqlplus username/password@database

-- Query the ALL_USERS view
SELECT username, created
FROM all_users
ORDER BY username;
```

Example 3: Viewing Current User's Information (Using USER_USERS)

```
-- Connect as the current user
sqlplus username/password@database

-- Query the USER_USERS view
SELECT username, default_tablespace, created
FROM user_users;
```

Example Outputs

Output for DBA_USERS Query:

USERNAME	ACCOUNT_STATUS	DEFAULT_TABLESPACE	CREATED
ANONYMOUS	EXPIRED & LOCKED	USERS	01-JAN-2020 12:00:00
APEX_040000	EXPIRED & LOCKED	SYSAUX	01-JAN-2020 12:00:00
HR	OPEN	USERS	01-JAN-2020 12:00:00
SCOTT	OPEN	USERS	01-JAN-2020 12:00:00
SYS	OPEN	SYSTEM	01-JAN-2020 12:00:00
SYSTEM	OPEN	SYSTEM	01-JAN-2020 12:00:00

Output for ALL_USERS Query:

USERNAME	CREATED
HR	01-JAN-2020 12:00:00
SCOTT	01-JAN-2020 12:00:00
SYS	01-JAN-2020 12:00:00
SYSTEM	01-JAN-2020 12:00:00

Output for USER_USERS Query:

USERNAME	DEFAULT_TABLESPACE	CREATED
HR	USERS	01-JAN-2020 12:00:00

Assigning Roles and Privileges

After creating a user, you need to grant appropriate roles and privileges to allow the user to perform required tasks.

Granting Roles

Roles are collections of privileges that can be granted to users.

```
GRANT role_name TO username;
```

Common Predefined Roles

CONNECT

Grants basic privileges to connect to the database.
Privileges Included:
CREATE SESSION

RESOURCE

Grants privileges to create and manage schema objects.
Privileges Included:
CREATE TABLE
CREATE VIEW
CREATE SEQUENCE
CREATE PROCEDURE
CREATE TRIGGER

DBA

Grants all system privileges, including the ability to administer the database.
Privileges Included:
All system privileges, such as CREATE USER, DROP USER, CREATE ANY TABLE, DROP ANY TABLE, etc.

IMP_FULL_DATABASE

Grants privileges to perform full database import operations.
Privileges Included:
EXECUTE_CATALOG_ROLE
SELECT ANY TABLE
INSERT ANY TABLE
UPDATE ANY TABLE
DELETE ANY TABLE

EXP_FULL_DATABASE

Grants privileges to perform full database export operations.
Privileges Included:
EXECUTE_CATALOG_ROLE
SELECT ANY TABLE

SELECT_CATALOG_ROLE
Grants privileges to select data from any data dictionary view.
Privileges Included:
SELECT ANY DICTIONARY

EXECUTE_CATALOG_ROLE
Grants execute privileges on various PL/SQL packages.
Privileges Included:
EXECUTE privileges on packages such as DBMS_RLS, DBMS_CRYPTO, DBMS_AQ, DBMS_FGA, etc.

Example:

```
GRANT CONNECT, RESOURCE TO john;
```

Granting System Privileges
System privileges allow users to perform specific actions on the database.

```
GRANT privilege TO username;
```

Example:

```
GRANT CREATE SESSION TO john;
GRANT CREATE TABLE TO john;
```

Granting Object Privileges
Object privileges allow users to perform actions on specific schema objects like tables, views, and procedures.

```
GRANT privilege ON object TO username;
```

Example:

```
GRANT SELECT, INSERT ON employees TO john;
```

User Profiles
Profiles help manage and control database resource usage for users. You can set limits on various resources like CPU time, logical reads, and the number of sessions.

Creating a Profile

```
CREATE PROFILE profile_name LIMIT
 SESSIONS_PER_USER 3
 CPU_PER_SESSION 10000
 CONNECT_TIME 60;
```

Example:

```
CREATE PROFILE limited_user LIMIT
 SESSIONS_PER_USER 2
 CONNECT_TIME 30;
```

Assigning a Profile to a User

```
ALTER USER username PROFILE profile_name;
```

Example:

```
ALTER USER john PROFILE limited_user;
```

Password Management

Oracle provides features for enforcing password complexity and expiration policies.

Enforcing Password Policies

Password policies are defined within profiles. You can enforce password complexity, expiration, and reuse.

```
ALTER PROFILE DEFAULT LIMIT
  PASSWORD_LIFE_TIME 30
  PASSWORD_REUSE_TIME 180
  PASSWORD_REUSE_MAX 10
  PASSWORD_LOCK_TIME 1
  PASSWORD_GRACE_TIME 7
  FAILED_LOGIN_ATTEMPTS 3;
```

Auditing

Auditing helps track and log user activities for security and compliance purposes.

Enabling Auditing

```
AUDIT CREATE SESSION;
AUDIT SELECT TABLE, INSERT TABLE, UPDATE TABLE, DELETE TABLE BY john BY ACCESS;
```

Example Workflow

```
--Create a User:
CREATE USER alice IDENTIFIED BY strongpassword;

--Grant Roles and Privileges:
GRANT CONNECT, RESOURCE TO alice;
GRANT CREATE SESSION TO alice;
GRANT SELECT, INSERT ON employees TO alice;

--Create a Profile and Assign It:
CREATE PROFILE developer_profile LIMIT
  SESSIONS_PER_USER 5
  CPU_PER_SESSION 10000
  CONNECT_TIME 120;

ALTER USER alice PROFILE developer_profile;

--Set Password Policies:
ALTER PROFILE developer_profile LIMIT
  PASSWORD_LIFE_TIME 45
  PASSWORD_REUSE_TIME 180
  PASSWORD_REUSE_MAX 5
  PASSWORD_LOCK_TIME 1
  PASSWORD_GRACE_TIME 10
  FAILED_LOGIN_ATTEMPTS 5;
```

```
--Enable Auditing:
AUDIT CREATE SESSION BY alice BY ACCESS;
AUDIT SELECT TABLE, INSERT TABLE, UPDATE TABLE, DELETE TABLE BY alice BY ACCESS;
```

Useful SQL Commands

Changing a User's Password:

```
ALTER USER alice IDENTIFIED BY newpassword;
```

Locking and Unlocking a User Account:

```
ALTER USER alice ACCOUNT LOCK;
ALTER USER alice ACCOUNT UNLOCK;
```

Dropping a User:

```
DROP USER alice CASCADE;
```

Oracle-Supplied Administrator Accounts

1. SYS

Description:

- The SYS user is the most powerful account in an Oracle database. It owns all the base tables and views for the database's data dictionary.
- The SYS account is created automatically when a database is created.

Privileges:

- The SYS user has the SYSDBA privilege, which grants all system privileges with administrative options.
- SYS owns the data dictionary tables, which store metadata about the database.

Usage:

- This account is used for all administrative tasks, including startup, shutdown, and recovery of the database.
- The SYS user should not be used for routine tasks; it's meant for high-level administrative functions.

Example:

```
sqlplus / as sysdba
```

2. SYSTEM

Description:

- The SYSTEM user is also created automatically when the database is created.
- It is used for creating and managing tables, views, and other schema objects.

Privileges:

- The SYSTEM user has a set of administrative privileges but does not have the SYSDBA privilege by default.
- It can create additional database users, roles, and profiles.

Usage:

- This account is used for routine administrative tasks and should be used instead of SYS for most administrative work.

Example:

```
sqlplus system/password
```

3. SYSBACKUP

Description:

- The SYSBACKUP user is designed specifically for performing backup and recovery operations.
- Introduced in Oracle Database 12c.

Privileges:

- The SYSBACKUP user has the SYSBACKUP privilege, which allows it to perform backup and recovery tasks without having full SYSDBA privileges.

Usage:

- This account is used for RMAN (Recovery Manager) operations and other backup tasks.

Example:

```
sqlplus / as sysbackup
```

4. SYSDG

Description:

- The SYSDG user is used for Data Guard operations.
- Introduced in Oracle Database 12c.

Privileges:

- The SYSDG user has the SYSDG privilege, which allows it to perform Data Guard-related tasks.

Usage:

- This account is used for managing Data Guard configurations.

Example:

```
sqlplus / as sysdg
```

5. SYSKM

Description:

- The SYSKM user is used for encryption key management.
- Introduced in Oracle Database 12c.

Privileges:

- The SYSKM user has the SYSKM privilege, which allows it to perform encryption key management tasks.

Usage:

- This account is used for managing Transparent Data Encryption (TDE) and other encryption-related tasks.

Example:

```
sqlplus / as syskm
```

6. SYSMAN

Description:

- The SYSMAN user is used by Oracle Enterprise Manager (OEM) for database management.

Privileges:

- The SYSMAN user has the necessary privileges to manage the database using Oracle Enterprise Manager.

Usage:

- This account is used internally by Oracle Enterprise Manager and should not be used for manual database administration tasks.

Example:

```
sqlplus sysman/password
```

Creating Oracle Database Users in a Multitenant Environment

- Common Users: Created in the root container and have access to all PDBs.
- Local Users: Created in specific PDBs and have access only to that PDB.
- Granting and Revoking Privileges: Use GRANT and REVOKE statements to assign and remove system and object privileges.
- Switching Containers: Use ALTER SESSION SET CONTAINER=pdb_name to switch between containers.

Creating Common Users

Common users are created in the root container (CDB$ROOT) and have access to all PDBs. They have the same identity across the entire multitenant environment.

Example:

```
-- Connect to the root container as SYSDBA
sqlplus / as sysdba

-- Create a common user
CREATE USER c##admin_user IDENTIFIED BY AdminPassword1 CONTAINER=ALL;

-- Grant system privileges to the common user
GRANT CONNECT, RESOURCE TO c##admin_user CONTAINER=ALL;
```

Creating Local Users

Local users are specific to individual PDBs and do not have access to other PDBs.

Example:

```
-- Connect to the root container as SYSDBA
sqlplus / as sysdba

-- Switch to the PDB
ALTER SESSION SET CONTAINER=pdb1;

-- Create a local user in the PDB
CREATE USER pdb_user IDENTIFIED BY PdbPassword1;
```

```
-- Grant system privileges to the local user
GRANT CONNECT, RESOURCE TO pdb_user;
```

Granting and Revoking Privileges

System Privileges

System privileges allow users to perform specific actions on the database.

Granting System Privileges

```
GRANT system_privilege TO username CONTAINER=ALL; -- For common users
GRANT system_privilege TO username; -- For local users
```

Example:

```
GRANT CREATE SESSION TO c##common_user CONTAINER=ALL;
GRANT CREATE TABLE TO pdb_user;
```

Revoking System Privileges

```
REVOKE system_privilege FROM username CONTAINER=ALL; -- For common users
REVOKE system_privilege FROM username; -- For local users
```

Example:

```
REVOKE CREATE SESSION FROM c##common_user CONTAINER=ALL;
REVOKE CREATE TABLE FROM pdb_user;
```

Object Privileges

Object privileges allow users to perform actions on specific schema objects like tables, views, and procedures.

Granting Object Privileges

```
GRANT object_privilege ON object TO username;
```

Example:

```
GRANT SELECT, INSERT ON employees TO pdb_user;
```

Revoking Object Privileges

```
REVOKE object_privilege ON object FROM username;
```

Example:

```
REVOKE SELECT, INSERT ON employees FROM pdb_user;
```

Managing Users in a Multitenant Environment

Listing Common Users:

```
SELECT username, common FROM cdb_users WHERE common = 'YES';
```

Listing Local Users in a PDB:

```
SELECT username FROM dba_users;
```

*Ensure you are connected to the specific PDB when running this query.

Switching Between Containers:

```
ALTER SESSION SET CONTAINER=cdb$root;
ALTER SESSION SET CONTAINER=pdb_name;
```

Example Workflow

```
--Connect to the CDB as SYSDBA:
sqlplus / as sysdba

--Create a Common User:
CREATE USER c##common_user IDENTIFIED BY commonPass CONTAINER=ALL;
GRANT CONNECT, RESOURCE TO c##common_user CONTAINER=ALL;

--Create a Local User in a PDB:
ALTER SESSION SET CONTAINER=pdb1;
CREATE USER pdb_user IDENTIFIED BY pdbPass;
GRANT CONNECT, RESOURCE TO pdb_user;

--Grant and Revoke Privileges:
-- Grant system privileges
GRANT CREATE SESSION TO c##common_user CONTAINER=ALL;
GRANT CREATE TABLE TO pdb_user;

-- Grant object privileges
GRANT SELECT, INSERT ON employees TO pdb_user;

-- Revoke system privileges
REVOKE CREATE SESSION FROM c##common_user CONTAINER=ALL;
REVOKE CREATE TABLE FROM pdb_user;

-- Revoke object privileges
REVOKE SELECT, INSERT ON employees FROM pdb_user;

--Verify Common and Local Users:
-- List common users
SELECT username, common FROM cdb_users WHERE common = 'YES';

-- List local users in the PDB
ALTER SESSION SET CONTAINER=pdb1;
SELECT username FROM dba_users;
```

Creating PDBs

Pluggable Databases (PDBs) are a key feature of Oracle's Multitenant architecture, introduced in Oracle Database 12c. They provide a way to consolidate multiple databases into a single Container Database (CDB), allowing for easier management and more efficient use of resources.

- Container Database (CDB): The root container that holds zero, one, or more PDBs. It includes the root container (CDB$ROOT), the seed PDB (PDB$SEED), and any user-created PDBs.
- Pluggable Database (PDB): A portable collection of schemas, schema objects, and non-schema objects that appears to an application as a separate database. Each PDB can have its own set of users, data, and configuration.
- Seed PDB (PDB$SEED): A template PDB that is used for creating new PDBs. It is read-only and provided by Oracle.

Benefits of PDBs

- Consolidation: Multiple PDBs can be consolidated into a single CDB, reducing the overhead of managing multiple database instances.
- Resource Management: Resources such as CPU and memory can be managed more efficiently across multiple PDBs within a single CDB.
- Security and Isolation: Each PDB is isolated from the others, ensuring that operations in one PDB do not affect the others.
- Portability: PDBs can be easily moved or cloned from one CDB to another, facilitating easier database deployment and management.
- Simplified Patching and Upgrades: Applying patches and upgrades to a single CDB can cover all PDBs, simplifying maintenance tasks.

Creating a New PDB from PDB$SEED

A new PDB can be created using the CREATE PLUGGABLE DATABASE statement. The PDB can be created from the seed PDB, from an existing PDB, or by cloning another PDB.

Showing PDB list

Using SHOW PDBS

```
-- Connect as a privileged user
sqlplus / as sysdba
```

```
-- Show all PDBs
SHOW PDBS;
```

Output:

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           MOUNTED    NO
```

Querying CDB_PDBS

```
-- Connect as a privileged user
sqlplus / as sysdba
```

```
-- Query to list all PDBs
SELECT pdb_id, pdb_name, status, open_mode FROM cdb_pdbs;
```

Output:

```
 PDB_ID  PDB_NAME                       STATUS      OPEN_MODE
-------- ------------------------------ ----------- ----------
    2    PDB$SEED                       NORMAL      READ ONLY
    3    PDB1                           NORMAL      READ WRITE
    4    PDB2                           NORMAL      MOUNTED
```

Additional Information from CDB_PDBS

The CDB_PDBS view provides more detailed information that can be customized in queries:

```
SELECT pdb_id, pdb_name, status, open_mode, creation_scn, con_id FROM cdb_pdbs;
```

Creating a PDB from the Seed PDB

```
CREATE PLUGGABLE DATABASE pdb1 ADMIN USER pdb_admin IDENTIFIED BY password
 FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/cdb1/pdbseed', '/u01/app/oracle/oradata/cdb1/pdb1');
```

Opening a PDB

```
ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

Closing a PDB

```
ALTER PLUGGABLE DATABASE pdb1 CLOSE IMMEDIATE;
```

Opening All PDBs

```
ALTER PLUGGABLE DATABASE ALL OPEN;
```

Closing All PDBs

```
ALTER PLUGGABLE DATABASE ALL CLOSE IMMEDIATE;
```

Switching Between Containers

To perform operations on a specific PDB, you need to switch your session to that PDB.

Switching to a PDB

```
ALTER SESSION SET CONTAINER = pdb1;
```

Switching Back to the Root Container

```
ALTER SESSION SET CONTAINER = CDB$ROOT;
```

Cloning PDBs

Prerequisites

- ✓ Ensure the source PDB is in read-only mode if cloning within the same CDB.
- ✓ Adequate storage must be available for the new PDB.
- ✓ Appropriate privileges are required (e.g., SYSDBA).

Steps to Clone a PDB

1. Cloning Within the Same CDB

Example for Cloning Within the Same CDB

```
-- Connect to the CDB as a privileged user
sqlplus / as sysdba

-- Set the source PDB to read-only mode
ALTER PLUGGABLE DATABASE source_pdb CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE source_pdb OPEN READ ONLY;

-- Create the new PDB by cloning the source PDB
CREATE PLUGGABLE DATABASE target_pdb FROM source_pdb
  FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/cdb1/source_pdb/',
'/u01/app/oracle/oradata/cdb1/target_pdb/');

-- Open the new PDB
ALTER PLUGGABLE DATABASE target_pdb OPEN;

-- Optionally, set the source PDB back to read-write mode
ALTER PLUGGABLE DATABASE source_pdb CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE source_pdb OPEN READ WRITE;
```

2. Cloning from a Different CDB

Example for Cloning from a Different CDB

```
-- Connect to the target CDB as a privileged user
sqlplus / as sysdba

-- Create a database link in the target CDB
CREATE DATABASE LINK source_cdb_link CONNECT TO source_user IDENTIFIED BY source_password USING
'source_cdb_tns';

-- Create the new PDB by cloning the source PDB from the source CDB
CREATE PLUGGABLE DATABASE target_pdb FROM source_pdb@source_cdb_link
  FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/source_cdb/source_pdb/',
'/u01/app/oracle/oradata/target_cdb/target_pdb/');

-- Open the new PDB
ALTER PLUGGABLE DATABASE target_pdb OPEN;
```

Additional Options

Using Snapshot Copy

If storage supports it, you can use a snapshot copy to clone the PDB, which is faster and more storage-efficient:

```
CREATE PLUGGABLE DATABASE target_pdb FROM source_pdb SNAPSHOT COPY
  FILE_NAME_CONVERT = ('/path/to/source_pdb/', '/path/to/target_pdb/');
```

Cloning with the NO DATA Option

To create a new PDB structure without data:

```
CREATE PLUGGABLE DATABASE target_pdb FROM source_pdb
  NO DATA
  FILE_NAME_CONVERT = ('/path/to/source_pdb/', '/path/to/target_pdb/');
```

Unplugging and Plugging in PDBs

Unplugging a PDB

Unplugging a PDB involves generating an XML file that contains metadata about the PDB. This file is used to plug the PDB into a new CDB.

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

-- Close the PDB
ALTER PLUGGABLE DATABASE pdb1 CLOSE IMMEDIATE;

-- Unplug the PDB
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO '/u01/app/oracle/oradata/cdb1/pdb1.xml';

-- Drop the PDB (optional)
DROP PLUGGABLE DATABASE pdb1 KEEP DATAFILES;
```

Plugging in a PDB

Plugging in a PDB involves using the XML file generated during the unplugging process to integrate the PDB into a new or existing CDB.

Example

```
-- Connect to the target CDB as SYSDBA
sqlplus / as sysdba

-- Create the new PDB from the XML file
CREATE PLUGGABLE DATABASE pdb1 USING '/u01/app/oracle/oradata/cdb1/pdb1.xml' COPY
FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/old_cdb/', '/u01/app/oracle/oradata/new_cdb/');

-- Open the new PDB
ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

Additional Options

1. Using NOCOPY Option:

If the file locations are the same and you do not want to copy the files:

```
CREATE PLUGGABLE DATABASE pdb1 USING '/u01/app/oracle/oradata/cdb1/pdb1.xml' NOCOPY
FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/old_cdb/', '/u01/app/oracle/oradata/new_cdb/');
```

2. Opening PDBs in Read-Only Mode:

If you need to open the PDB in read-only mode for verification:

```
ALTER PLUGGABLE DATABASE pdb1 OPEN READ ONLY;
```

3. Checking PDB Compatibility:

Before plugging in a PDB, you can check compatibility:

```
DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  pdb_descr_file => '/u01/app/oracle/oradata/cdb1/pdb1.xml',
  pdb_name => 'PDB1');
```

Creating and Managing Tablespaces

Tablespaces in Oracle Database are logical storage units that group related logical structures together. They play a crucial role in database storage management by providing a way to manage and allocate space for database objects such as tables and indexes.

Key Concepts

- ✓ Logical Storage Unit: Tablespaces group logical storage structures like segments, which consist of extents. Extents are made up of data blocks.
- ✓ Data Files: Each tablespace consists of one or more physical data files on disk that store the actual data.
- ✓ Default Tablespace: When creating users or objects, if a specific tablespace is not specified, they are created in the default tablespace.

Types of Tablespaces

Permanent Tablespaces: Store persistent data like tables and indexes.
Example: USERS, SYSTEM, SYSAUX

Temporary Tablespaces: Store temporary data that is generated during SQL operations such as sorts.
Example: TEMP

Undo Tablespaces: Store undo information for uncommitted transactions.
Example: UNDOTBS1

Common Tablespaces

- SYSTEM: Contains the data dictionary and essential system data.
- SYSAUX: Auxiliary tablespace to offload data from the SYSTEM tablespace.
- USERS: Default tablespace for user data.
- TEMP: Temporary tablespace for sorting operations.
- UNDO: Stores undo information for transactions.

Creating Tablespaces

Creating a Tablespace
You can create a tablespace using the CREATE TABLESPACE statement.

Example:

```
CREATE TABLESPACE example_tbs
DATAFILE '/u01/app/oracle/oradata/example_tbs01.dbf' SIZE 50M
AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

Creating a Temporary Tablespace

```
CREATE TEMPORARY TABLESPACE temp_tbs
TEMPFILE '/u01/app/oracle/oradata/temp_tbs01.dbf' SIZE 50M
AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

Creating an Undo Tablespace

```
CREATE UNDO TABLESPACE undo_tbs
DATAFILE '/u01/app/oracle/oradata/undo_tbs01.dbf' SIZE 100M
AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

Verify the Tablespace Creation:

```
SELECT tablespace_name, status, contents, extent_management, allocation_type
FROM dba_tablespaces
WHERE tablespace_name = 'EXAMPLE_TBS';
```

Output:

```
TABLESPACE_NAME    STATUS          CONTENTS      EXTENT_MANAGEMENT         ALLOCATION_TYPE
---------------- --------- -------- ----------------- ---------------
EXAMPLE_TBS           ONLINE          PERMANENT    LOCAL                          SYSTEM
```

Altering and Dropping Tablespaces

Add a Data File to a Tablespace:

```
ALTER TABLESPACE example_tbs
ADD DATAFILE '/u01/app/oracle/oradata/example_tbs02.dbf' SIZE 50M
AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
```

Enable Autoextend for a Data File:

```
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/example_tbs01.dbf'
AUTOEXTEND ON NEXT 10M MAXSIZE 1G;
```

Resize a Data File:

```
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/example_tbs01.dbf'
RESIZE 200M;
```

Take a Tablespace Offline:

```
ALTER TABLESPACE example_tbs OFFLINE;
```

Bring a Tablespace Online:

```
ALTER TABLESPACE example_tbs ONLINE;
```

Drop a Tablespace (Including Data Files):

```
DROP TABLESPACE example_tbs INCLUDING CONTENTS AND DATAFILES;
```

Drop a Tablespace (Keep Data Files):

```
DROP TABLESPACE example_tbs KEEP DATAFILES;
```

Viewing Tablespace Information

Querying Tablespace Information:

```
SELECT tablespace_name, status, contents, extent_management, allocation_type
FROM dba_tablespaces;
```

Tablespace Encryption by Default in DBCS

Oracle Database Cloud Service (DBCS) offers enhanced security features, including tablespace encryption by default. This feature ensures that all data stored in the tablespaces is encrypted, providing an additional layer of security to protect sensitive information.

Key Concepts of Tablespace Encryption

- ✓ Encryption Algorithms: Oracle supports various encryption algorithms such as AES128, AES192, and AES256. AES256 is the most secure and commonly used.
- ✓ Transparent Data Encryption (TDE): TDE encrypts data at rest, ensuring that data is automatically encrypted when written to the disk and decrypted when read from the disk.

When you create a new tablespace in DBCS, it is automatically encrypted using the default encryption algorithm (AES256) unless specified otherwise.

Example: Creating a Default Encrypted Tablespace

```
CREATE TABLESPACE secure_tbs
DATAFILE '/u01/app/oracle/oradata/secure_tbs01.dbf' SIZE 100M;
```

*In the above example, secure_tbs is automatically encrypted because tablespace encryption is enabled by default.

Example: Specifying Encryption Algorithm

If you want to specify a different encryption algorithm, you can do so using the ENCRYPTION clause:

```
CREATE TABLESPACE secure_tbs
DATAFILE '/u01/app/oracle/oradata/secure_tbs01.dbf' SIZE 100M
ENCRYPTION USING 'AES128';
```

Managing Encrypted Tablespaces

Viewing Encrypted Tablespaces

```
SELECT tablespace_name, encryptedts
FROM v$encrypted_tablespaces;
```

Example Output:

```
TABLESPACE_NAME            ENCRYPTEDTS
-------------------------- -----------
SECURE_TBS                 YES
USERS                      NO
```

Querying Tablespace Information

You can query various data dictionary views to get information about tablespaces:

- DBA_TABLESPACES: Provides information about all tablespaces.
- DBA_DATA_FILES: Provides information about data files associated with each tablespace.
- DBA_TEMP_FILES: Provides information about temporary files associated with temporary tablespaces.
- DBA_FREE_SPACE: Shows free space available in each tablespace.

Example Queries

To list all tablespaces:

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
```

To get detailed information about a specific tablespace:

```
SELECT *
FROM DBA_TABLESPACES
WHERE TABLESPACE_NAME = 'MY_TABLESPACE';
```

To check free space in tablespaces:

```
SELECT TABLESPACE_NAME, FILE_ID, BYTES, BLOCKS
FROM DBA_FREE_SPACE;
```

Specifying Tablespace for a Table

```
CREATE TABLE employees (
   employee_id NUMBER PRIMARY KEY,
   first_name VARCHAR2(50),
   last_name VARCHAR2(50),
   email VARCHAR2(100)
) TABLESPACE userspace;
```

Moving a Table to a Different Tablespace

```
ALTER TABLE employees MOVE TABLESPACE new_userspace;
```

Creating Users with Default Tablespaces

When creating a new user, you can specify a default tablespace for their objects and a temporary tablespace for their temporary data:

```
CREATE USER john IDENTIFIED BY password
DEFAULT TABLESPACE userspace
TEMPORARY TABLESPACE temp;

GRANT CREATE SESSION, CREATE TABLE TO john;
```

Querying Tablespace Information

To find out which tablespace a table is in, you can query the DBA_SEGMENTS view:

```
SELECT table_name, tablespace_name
FROM dba_segments
WHERE segment_type = 'TABLE'
AND table_name = 'EMPLOYEES';
```

Encrypting Existing Tablespaces

To encrypt an existing tablespace, you can use the ALTER TABLESPACE command. However, note that this operation requires moving the data to an encrypted tablespace, as direct encryption of an existing tablespace is not supported.

Example: Encrypting an Existing Tablespace
Create an Encrypted Tablespace:

```
CREATE TABLESPACE new_secure_tbs
DATAFILE '/u01/app/oracle/oradata/new_secure_tbs01.dbf' SIZE 100M;
```

Move the Data to the Encrypted Tablespace:

```
ALTER TABLE your_table MOVE TABLESPACE new_secure_tbs;
```

Drop the Original Tablespace (if needed):

```
DROP TABLESPACE old_tbs INCLUDING CONTENTS AND DATAFILES;
```

Key Management

Oracle TDE uses the Oracle Wallet to store encryption keys. In DBCS, the Oracle Wallet is managed automatically, but you can perform operations on the wallet as needed.

Example: Opening the Wallet

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY wallet_password;
```

Example: Closing the Wallet

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE;
```