# Lab 7.1: Creating and Managing Common and Local Users with SQL Developer

## Objective:

To create common users in the CDB (CDBLAB) and assign them SYSDBA, SYSOPER, and SYSBACKUP privileges. To create a local user in PDBLAB1 with SYSBACKUP and additional privileges, and demonstrate the scope of these privileges.

## Prerequisites:

- Oracle Database is installed and running.
- CDBLAB is created with PDBLAB1, PDBLAB2, and PDBLAB3.
- SQL Developer is installed and configured to connect to the Oracle database.

## Steps:

### 1. Create Common Users in the CDB

1. **Launch SQL Developer:**
   - Open SQL Developer and connect to the CDBLAB as SYSDBA.
2. **Create Common User C##ADMIN with SYSDBA Privilege:**
   - Navigate to `Connections` and select your CDBLAB connection.
   - Open a SQL Worksheet for CDBLAB and run the following commands:

```
CREATE USER C##ADMIN IDENTIFIED BY admin_password CONTAINER=ALL;
GRANT SYSDBA TO C##ADMIN CONTAINER=ALL;
```

3. **Create Common User C##OPER with SYSOPER Privilege:**
   - Repeat the above steps to create C##OPER.

```
CREATE USER C##OPER IDENTIFIED BY oper_password CONTAINER=ALL;
GRANT SYSOPER TO C##OPER CONTAINER=ALL;
```

4. **Create Common User C##BACKUP with SYSBACKUP Privilege:**
   - Repeat the above steps to create C##BACKUP.

```
CREATE USER C##BACKUP IDENTIFIED BY backup_password CONTAINER=ALL;
GRANT SYSBACKUP TO C##BACKUP CONTAINER=ALL;
```

5. **Verify and Edit Users in the DBA Pane:**
   - Navigate to `View` -> `DBA`.
   - Ensure you have a connection to CDBLAB in the DBA pane.
   - In the `DBA` pane, expand `Security` -> `Users`.
   - Right-click on `Users` under `CDBLAB` and select `Refresh`.
   - Verify that `C##ADMIN`, `C##OPER`, and `C##BACKUP` are listed.

## 2. Verify Common Users in PDBs

1. **Connect to PDBLAB1 as C##ADMIN:**
    - In SQL Developer, create a new connection:
        - Connection Name: `C##ADMIN_PDBLAB1`
        - Username: `C##ADMIN`
        - Password: `admin_password`
        - Connection Type: `Basic`
        - Role: `SYSDBA`
        - Hostname: `localhost`
        - Port: `1521`
        - Service Name: `pdblab1`
2. **Test and Save the Connection:**
    - Click the `Test` button to verify the connection.
    - Click `Save` and then `Connect`.
3. **Verify Privileges:**
    - Run the following query to check the privileges:

    ```
    SELECT * FROM SESSION_PRIVS;
    ```

    - Ensure `SYSDBA` privilege is listed.
4. **Repeat for C##OPER and C##BACKUP:**
    - Create connections for `C##OPER` and `C##BACKUP` to `PDBLAB1`, `PDBLAB2`, and `PDBLAB3`, and verify their privileges.

## 3. Create Local User in PDBLAB1

1. **Connect to PDBLAB1 as SYS:**
    - Use the SYS connection to PDBLAB1 in SQL Developer.
2. **Create Local User in PDBLAB1:**
    - Navigate to `Users` under `PDBLAB1` and select `Create User...`.

    ```
    CREATE USER local_user IDENTIFIED BY local_password;
    GRANT CONNECT, RESOURCE, SYSBACKUP, CREATE TABLE, CREATE VIEW TO local_user;
    ```

3. **Test Local User Privileges:**
    - Create a new connection for `local_user` in SQL Developer:
        - Connection Name: `LOCAL_USER_PDBLAB1`
        - Username: `local_user`
        - Password: `local_password`
        - Connection Type: `Basic`
        - Hostname: `localhost`
        - Port: `1521`

        - Service Name: `pdblab1`
    - Verify the connection and run the following commands to test the privileges:

```
CREATE TABLE test_table (id NUMBER);
CREATE VIEW test_view AS SELECT * FROM test_table;
```

4. **Attempt Operations on Other PDBs:**
    - Try to connect to `PDBLAB2` and `PDBLAB3` using `local_user` and verify that the operations fail.

### 4. Summary and Verification

1. **Verify Common Users in PDBLAB1, PDBLAB2, and PDBLAB3:**
    - Connect to each PDB using the common users and verify their privileges.
2. **Verify Local User in PDBLAB1:**
    - Ensure that `local_user` has the necessary privileges only in `PDBLAB1`.

### SQL Scripts Summary:

```
-- Create Common Users
CREATE USER C##ADMIN IDENTIFIED BY admin_password CONTAINER=ALL;
GRANT SYSDBA TO C##ADMIN CONTAINER=ALL;

CREATE USER C##OPER IDENTIFIED BY oper_password CONTAINER=ALL;
GRANT SYSOPER TO C##OPER CONTAINER=ALL;

CREATE USER C##BACKUP IDENTIFIED BY backup_password CONTAINER=ALL;
GRANT SYSBACKUP TO C##BACKUP CONTAINER=ALL;

-- Create Local User in PDBLAB1
CREATE USER local_user IDENTIFIED BY local_password;
GRANT CONNECT, RESOURCE, SYSBACKUP, CREATE TABLE, CREATE VIEW TO local_user;
```

### Connection Details:

- **Common Users:**
    - Username: C##ADMIN, C##OPER, C##BACKUP
    - Password: admin_password, oper_password, backup_password
    - Service Name: CDBLAB, pdblab1, pdblab2, pdblab3
- **Local User in PDBLAB1:**
    - Username: local_user
    - Password: local_password
    - Service Name: pdblab1

By following these steps, you'll have successfully created common and local users, assigned appropriate privileges, and verified their accessibility within the Oracle CDB and PDB environment using SQL Developer.

# Addendum: Creating and Assigning User Profiles in Oracle

## Objective:

To create and assign user profiles to control resource consumption, manage account status, and password expiration, and use Oracle-supplied password functions in profiles.

### Steps:

### 1. Create a Profile to Control Resource Consumption

1. **Connect to PDBLAB1 as SYS:**
   - Use SQL Developer to connect to PDBLAB1 as the SYS user with SYSDBA privileges.
2. **Create a Profile:**
   - Create a profile named `DEV_PROFILE` to control resource consumption.

```
CREATE PROFILE DEV_PROFILE LIMIT
    SESSIONS_PER_USER 2
    CPU_PER_SESSION 10000
    CPU_PER_CALL 1000
    CONNECT_TIME 60
    IDLE_TIME 10
    LOGICAL_READS_PER_SESSION 1000
    LOGICAL_READS_PER_CALL 100;
```

### 2. Manage Account Status and Password Expiration

1. **Create a Profile for Password Management:**
   - Create a profile named `PASSWORD_MANAGEMENT_PROFILE` to manage account status and password expiration using Oracle-supplied password functions.

```
CREATE PROFILE PASSWORD_MANAGEMENT_PROFILE LIMIT
    FAILED_LOGIN_ATTEMPTS 3
    PASSWORD_LIFE_TIME 30
    PASSWORD_REUSE_TIME 365
    PASSWORD_REUSE_MAX 5
    PASSWORD_VERIFY_FUNCTION ORA12C_VERIFY_FUNCTION
    PASSWORD_LOCK_TIME 1
    PASSWORD_GRACE_TIME 10;
```

2. **Assign Profiles to Users:**
   - Assign the created profiles to the user `dev_user`.

     ```
     CREATE USER dev_user IDENTIFIED BY dev_password;
     ```

     Then

```sql
ALTER USER dev_user PROFILE DEV_PROFILE;
ALTER USER dev_user PROFILE PASSWORD_MANAGEMENT_PROFILE;
```

### 3. Verify the Assigned Profiles

1. **Check the Assigned Profiles:**
   - Verify the profiles assigned to `dev_user`.

```sql
SELECT username, profile FROM dba_users WHERE username = 'DEV_USER';
```

2. **Check the Profile Details:**
   - Verify the details of the profiles assigned.

```sql
SELECT * FROM dba_profiles WHERE profile = 'DEV_PROFILE';
SELECT * FROM dba_profiles WHERE profile = 'PASSWORD_MANAGEMENT_PROFILE';
```

### 4. Test the Profiles

1. **Test Resource Consumption Limits:**
   - Connect to PDBLAB1 as `dev_user` and perform operations to test resource consumption limits.

```sql
-- Connect to PDBLAB1 as dev_user and perform operations
-- Example: Multiple sessions, long-running queries, etc.
```

2. **Test Password Management:**
   - Change the password and perform multiple login attempts to test password management.

```sql
-- Connect to PDBLAB1 as dev_user and change password
ALTER USER dev_user IDENTIFIED BY new_password123 REPLACE dev_password;

-- Perform multiple failed login attempts to test FAILED_LOGIN_ATTEMPTS
```

### Summary:

By following these steps, you will have created and assigned profiles to control resource consumption and manage account status and password expiration. The profiles will help in enforcing resource limits and security policies for the `dev_user`.

This addendum provides practical experience in using Oracle profiles to manage user resources and security, enhancing the overall database administration skills.

# Addendum: Creating Tables, Inserting Data, and Creating a Developer User

## Objective:

To create a table and insert data in PDBLAB1, then create a developer user with specific read and write permissions for that table. Finally, write a Java program to read and write data to the table.

### Steps:

### 1. Create a Table and Insert Data in PDBLAB1

1. **Connect to PDBLAB1 as SYS:**
    - Use SQL Developer to connect to PDBLAB1 as the SYS user with SYSDBA privileges.
2. **Create a Table:**
    - In SQL Developer, create a new table called `EMPLOYEES`.

```
CREATE TABLE EMPLOYEES (
    EMP_ID NUMBER PRIMARY KEY,
    NAME VARCHAR2(50),
    DEPARTMENT VARCHAR2(50),
    SALARY NUMBER
);
```

3. **Insert Data into the Table:**
    - Insert sample data into the `EMPLOYEES` table.

```
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES (1, 'John Doe', 'IT', 70000);
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES (2, 'Jane Smith', 'HR', 60000);
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES (3, 'Alice Johnson', 'Finance', 75000);
COMMIT;
```

### 2. Create a Developer User with Specific Permissions

1. **Create a Developer User:**
    - In SQL Developer, create a new user for the developer.

```
CREATE USER dev_user IDENTIFIED BY dev_password;
```

2. **Grant Necessary Permissions:**
    - Grant the necessary permissions for the developer user to read and write to the `EMPLOYEES` table.

```
GRANT CONNECT TO dev_user;
GRANT CREATE SESSION TO dev_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON EMPLOYEES TO dev_user;
```

### 3. Write a Java Program to Read and Write Data

1. **Install the Oracle JDBC Driver:**

- Download the Oracle JDBC driver (ojdbc8.jar) using `wget` and place it in a known directory, such as `/opt/oracle/jdbc`.

```
mkdir -p /opt/oracle/jdbc
cd /opt/oracle/jdbc
wget https://download.oracle.com/otn-pub/otn_software/jdbc/1918/ojdbc8.jar
```

2. **Java Program:** (this is optional)

- Write a Java program (inthe same /opt/oracle/jdbc directory) to connect to PDBLAB1 and perform read and write operations on the `EMPLOYEES` table.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ManageEmployees {

    // Database connection details
    private static final String URL =
"jdbc:oracle:thin:@localhost:1521/pdblab1";
    private static final String USER = "dev_user";
    private static final String PASSWORD = "dev_password";

    public static void main(String[] args) {
        try {
            // Load Oracle JDBC Driver
            Class.forName("oracle.jdbc.driver.OracleDriver");

            // Connect to the database
            Connection connection = DriverManager.getConnection(URL, USER,
PASSWORD);

            // Insert a new employee
            insertData(connection, 4, "Bob Brown", "Marketing", 65000);

            // Read and display data
            System.out.println("Data in EMPLOYEES table:");
            readData(connection);

            // Close the connection
            connection.close();
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    private static void readData(Connection connection) throws SQLException {
        String query = "SELECT * FROM EMPLOYEES";
        PreparedStatement stmt = connection.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();

        while (rs.next()) {
            System.out.println("EMP_ID: " + rs.getInt("EMP_ID") +
                               ", NAME: " + rs.getString("NAME") +
                               ", DEPARTMENT: " + rs.getString("DEPARTMENT") +
                               ", SALARY: " + rs.getDouble("SALARY"));
        }

        rs.close();
        stmt.close();
    }

    private static void insertData(Connection connection, int empId, String
name, String department, double salary) throws SQLException {
        String query = "INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT,
SALARY) VALUES (?, ?, ?, ?)";
        PreparedStatement stmt = connection.prepareStatement(query);
        stmt.setInt(1, empId);
        stmt.setString(2, name);
        stmt.setString(3, department);
        stmt.setDouble(4, salary);
        stmt.executeUpdate();
        stmt.close();
    }
}
```

3. **Compile and Run the Java Program:** (Optional)

   - Ensure the `ojdbc8.jar` file is in the same directory as your Java program or set the CLASSPATH to include the `ojdbc8.jar`.

```
javac -cp .:/opt/oracle/jdbc/ojdbc8.jar ManageEmployees.java
java -cp .:/opt/oracle/jdbc/ojdbc8.jar ManageEmployees
```

## Summary:

By following these steps, you will have created a table in PDBLAB1, inserted data, and created a developer user with specific read and write permissions. The Java program will demonstrate how to connect to the database and perform basic CRUD operations on the table.

This addendum provides practical experience in managing database objects and users, as well as integrating database operations with a Java application.