Practice 16: Implementing Miscellaneous Topics

Practice Overview

In this practice, you will perform the following:

- Query the data owned by common users in multiple PDBs.
- Change the data owned by common users in multiple PDBs.
- Specify the Default Container target and test it.
- Apply an auditing policy in `PDB1`.

Practice Assumptions

- `CDB1` (in `srv1`) database is up and running.

Accessing Data in Multiple PDBs

A. Querying data owned by common users in multiple PDBs

In this section of the practice, you will learn on how to query tables owned by a common user and located in multiple PDBs.

As a common user, you will create a table in all the PDBs as well as in the root. This table imitates a table to store the audit login records of the users of a specific application in all the PDBs.

1. Verify that `USERS` tablespace exists in the root. Create it, if it does not exist.

```
sqlplus / as sysdba
SELECT NAME FROM V$TABLESPACE WHERE NAME='USERS' AND CON_ID=1;

-- if it doesn't exist, create it:
CREATE TABLESPACE USERS;
```

2. Create `USERS` tablespace in `PDB1`.

```
ALTER SESSION SET CONTAINER=PDB1;
CREATE TABLESPACE USERS;
```

3. Grant privileges to the common user `C##USER1` as shown in the following code block:

```
conn / as sysdba
GRANT CREATE SESSION TO C##USER1 CONTAINER=ALL;
GRANT CREATE TABLE TO C##USER1 CONTAINER=ALL;
GRANT ALTER SESSION TO C##USER1 CONTAINER=ALL;
GRANT SET CONTAINER TO C##USER1 CONTAINER=ALL;
GRANT UNLIMITED TABLESPACE TO C##USER1 CONTAINER=ALL;
```

4. As the common user `C##USER1` create `LOGIN_AUDIT` table in the root and as well as in the PDBs, as shown in the following code block:

```
conn C##USER1/oracle

CREATE TABLE LOGIN_AUDIT
(
   RECORD_ID    NUMBER,
   DB           VARCHAR2(5),
   USER_ID      VARCHAR2(5),
   LOGIN_TIME   DATE,
   LOGOFF_TIME  DATE,
   CONSTRAINT LOGIN_AUDIT_PK PRIMARY KEY(RECORD_ID)
)
TABLESPACE USERS;
```

```
ALTER SESSION SET CONTAINER=PDB1;

CREATE TABLE LOGIN_AUDIT
(
   RECORD_ID    NUMBER,
   DB           VARCHAR2(5),
   USER_ID      VARCHAR2(5),
   LOGIN_TIME   DATE,
   LOGOFF_TIME  DATE,
   CONSTRAINT LOGIN_AUDIT_PK PRIMARY KEY(RECORD_ID)
)
TABLESPACE USERS;

INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (11,'PDB1','USER1', SYSDATE-1.5, SYSDATE-1.4);
INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (21,'PDB1','USER2', SYSDATE-1.6, SYSDATE-1.5);
INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (31,'PDB1','USER3', SYSDATE-1.7, SYSDATE-1.5);
COMMIT;

ALTER SESSION SET CONTAINER=PDB2;

CREATE TABLE LOGIN_AUDIT
(
   RECORD_ID    NUMBER,
   DB           VARCHAR2(5),
   USER_ID      VARCHAR2(5),
   LOGIN_TIME   DATE,
   LOGOFF_TIME  DATE,
   CONSTRAINT LOGIN_AUDIT_PK PRIMARY KEY(RECORD_ID)
)
TABLESPACE USERS;

INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (12,'PDB2','USER4', SYSDATE-2.5, SYSDATE-2.4);
INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (22,'PDB2','USER5', SYSDATE-2.6, SYSDATE-2.5);
INSERT INTO LOGIN_AUDIT (RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME)
VALUES (32,'PDB2','USER4', SYSDATE-2.7, SYSDATE-2.5);
COMMIT;
```

5. Login as `C##USER1` to the root and query the table data from the PDBs using the `SELECT` statements as shown in the following code block:

```
conn C##USER1/oracle

col con$name format a8
ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-RR HH24:MI';

-- select all the rows from all the PDBs
SELECT RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME, CON_ID, CON$NAME
FROM CONTAINERS(LOGIN_AUDIT);

-- select all the rows from specific PDBs
SELECT RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME, CON_ID, CON$NAME
FROM CONTAINERS(LOGIN_AUDIT)
WHERE CON$NAME IN ('PDB1', 'PDB2');
```

B. Changing the data owned by common users in multiple PDBs

In this section of the practice, you will apply DML statements on a table owned by a common user and located in multiple PDBs.

Note: the demonstration in this practice was tested on Oracle database release 12.2.0.1.0. As the database response is different from what Oracle documentation states, the response of the database after applying a future patch set could be different than what is demonstrated in this section of the practice.

6. Login to the root as the common user `C#USER1` and insert a row in the table `LOGIN_AUDIT` in `PDB1` using the `CONTAINERS` clause. You need to include the keyword `CON_ID` in the insert statement to specify which PDB should the insert affect.

```
conn C##USER1/oracle
ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-RR HH24:MI';

INSERT INTO CONTAINERS(LOGIN_AUDIT) (CON_ID, RECORD_ID, DB, USER_ID, LOGIN_TIME,
LOGOFF_TIME)
VALUES (3,41,'PDB1','USER4', SYSDATE-2.5, SYSDATE-2.4);
COMMIT;

-- verify:
SELECT RECORD_ID, DB, USER_ID, LOGIN_TIME, LOGOFF_TIME, CON_ID, CON$NAME
FROM CONTAINERS(LOGIN_AUDIT)
WHERE CON$NAME IN ('PDB1') AND RECORD_ID=41;
```

7. Make an update on the record that you created in the previous step using the `CONTAINERS` clause. Delete the row afterwards.

```
UPDATE CONTAINERS(LOGIN_AUDIT) SET USER_ID='USER5'
WHERE CON_ID=3 AND RECORD_ID=41;
COMMIT;

DELETE CONTAINERS(LOGIN_AUDIT) WHERE CON_ID=3 AND RECORD_ID=41;
COMMIT;
```

8. Try deleting all the rows in `LOGIN_AUDIT` in `PDB1` and `PDB2` using single `DELETE` statement.

You should receive the `ORA-65319` error. According to the documentation, this is not the expected behavior but this is how it actually goes in the release 12.2.0.1.0.

```
DELETE CONTAINERS(LOGIN_AUDIT) WHERE CON_ID in (3,5);
```

C. Specifying the Default Container Target

In this section of the practice, you will examine using the default container for DML statements that use CONTAINERS clause.

9. Check out the value of the database property CONTAINERS_DEFAULT_TARGET.

If this property is not being set (the select statement below returns no row), the default container is the current container.

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES
WHERE PROPERTY_NAME='CONTAINERS_DEFAULT_TARGET';
```

10. Switch to PDB1 container and update all the rows in the table LOGIN_AUDIT.

Observe that the update statement affects the table in the current container.

```
conn C##USER1/oracle

ALTER SESSION SET CONTAINER=PDB1;
UPDATE CONTAINERS(LOGIN_AUDIT) SET USER_ID='USERx';

SELECT USER_ID FROM LOGIN_AUDIT;
ROLLBACK;
SELECT USER_ID FROM LOGIN_AUDIT;
```

11. Change the CONTAINERS_DEFAULT_TARGET to PDB2

Observe that the property is changed to the DBID of PDB2.

```
conn / as sysdba
ALTER DATABASE CONTAINERS DEFAULT TARGET = (PDB2);

SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES
WHERE PROPERTY_NAME='CONTAINERS_DEFAULT_TARGET';

SELECT NAME, DBID, CON_UID FROM V$PDBS WHERE NAME='PDB2';
```

12. Connect to the root as the user C##USER1, update all the rows in the LOGIN_AUDIT table using the CONTAINERS clause without specifying the PDB, and check out which PDB got affected by the statement.

```
conn C##USER1/oracle
UPDATE CONTAINERS(LOGIN_AUDIT) SET USER_ID='USERy';

-- the following commit is a must to see the output of UPDATE statement because
--  the transaction of the UPDATE statement is different from the current
--  transaction
COMMIT;

-- select from the table in the root:
SELECT USER_ID FROM LOGIN_AUDIT;
```

```
-- select from the table in pdb1:
SELECT USER_ID FROM CONTAINERS(LOGIN_AUDIT) WHERE CON_ID=3;

-- select from the table in pdb2:
SELECT USER_ID FROM CONTAINERS(LOGIN_AUDIT) WHERE CON_ID=4;
```

D. Using Auditing Policies in PDBs

In this section of the practice, you will examine how to create an auditing policy in a PDB. You will create a new audit policy, test the new policy, and examine how to retrieve the audit records.

Note: this practice is not to cover the auditing subject in Oracle Database. That subject is beyond the scope of the course. This section of the practice is to cover how Oracle database audit can be used in a PDB.

13. Login as `SYSDBA` in `PDB1` and create an audit policy for any `CREATE TABLESPACE` operation in `PDB1`. Enable the audit policy.

```
sqlplus sys/oracle@pdb1 as sysdba

CREATE AUDIT POLICY audit_tbs ACTIONS CREATE TABLESPACE;
AUDIT POLICY audit_tbs;

col user_name format A10
col policy_name format A10
col entity_name format a10
SELECT * FROM AUDIT_UNIFIED_ENABLED_POLICIES WHERE POLICY_NAME='AUDIT_TBS';
```

14. Create a new tablespace in `PDB1` and verify that the operation has been audited.

```
conn system/oracle@pdb1

CREATE TABLESPACE tbs1;

col dbusername format a12
col action_name format a20
col object_name format a12
col event_time format a19

SELECT DBUSERNAME, ACTION_NAME, OBJECT_NAME,
       to_char(EVENT_TIMESTAMP,'DD-MON-RR HH12:MI AM') EVENT_TIME
FROM   UNIFIED_AUDIT_TRAIL
WHERE  ACTION_NAME LIKE '%TABLESPACE%'
ORDER  BY EVENT_TIMESTAMP;
```

15. Drop the tablespace and the audit policy.

```
DROP TABLESPACE tbs1 INCLUDING CONTENTS AND DATAFILES;

conn sys/oracle@pdb1 as sysdba
NOAUDIT POLICY audit_tbs;
DROP AUDIT POLICY audit_tbs;
```

In the rest of this section of the practice, you will examine the auditing policies that are already enabled in the database.

16. Check out the enabled audit policies in PDP1.

```
conn sys/oracle@pdb1 as sysdba

col policy_name format a20
col user_name format a15

SELECT POLICY_NAME, USER_NAME FROM AUDIT_UNIFIED_ENABLED_POLICIES ORDER BY 1;
```

17. Check which actions are audited by ORA_SECURECONFIG audit policy.

Observe that among the operations that the policy audits is the "CREATE USER" operation.

```
col audit_option format a40

SELECT AUDIT_OPTION
  FROM AUDIT_UNIFIED_POLICIES
 WHERE POLICY_NAME ='ORA_SECURECONFIG' ORDER BY 1;
```

18. Create a local user in PDB1 and then drop it.

If the in-memory audit records have not been flushed to the dictionary tables, execute the DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL procedure.

```
conn system/oracle@pdb1

CREATE USER testuser IDENTIFIED BY oracle;
DROP USER testuser;
```

19. Connect to PDB1 and check if this operation is reported by the UNIFIED_AUDIT_TRAIL view.

```
conn sys/oracle@pdb1 as sysdba

col dbusername format a12
col action_name format a20
col object_name format a12
col event_time format a19

SELECT DBUSERNAME, ACTION_NAME, OBJECT_NAME,
       to_char(EVENT_TIMESTAMP,'DD-MON-RR HH12:MI AM') EVENT_TIME
FROM  UNIFIED_AUDIT_TRAIL
WHERE ACTION_NAME = 'CREATE USER' AND OBJECT_NAME='TESTUSER'
ORDER BY EVENT_TIMESTAMP;
```

20. Connect to the root and run the same query that you ran in the previous step.

You will observe that the query does not report the action this time. This is because this view reports the audit records of only the current container.

```
conn / as sysdba
```

21. Run the same query against the consolidated version of the view.

```
col dbusername format a12
col action_name format a20
col object_name format a12
col event_time format a19

SELECT CON_ID, DBUSERNAME, ACTION_NAME, OBJECT_NAME,
       to_char(EVENT_TIMESTAMP,'DD-MON-RR HH12:MI AM') EVENT_TIME
FROM  CDB_UNIFIED_AUDIT_TRAIL
WHERE ACTION_NAME = 'CREATE USER' AND OBJECT_NAME='TESTUSER'
ORDER BY EVENT_TIMESTAMP;
```

Summary

- You can use the `CONTAINERS` clause to query the data owned by a common user in multiple PDBs.
- A DML statement can be issued using the `CONTAINERS` clause, but it should affect a single PDB at a time. This is not the expected behavior and it might be fixed in future releases.
- Using the Default Container Target, you can control which PDB will be affected when you issue a DML with the `CONTAINERS` clause and you do not specify which PDB should be affected.
- You can create and apply an auditing policy in the CDB to audit the actions in all the PDBs. The view `CDB_UNIFIED_AUDIT_TRAIL` reports the auditing records of all the PDBs.