Miscellaneous Topics on CDB and PDBs

By Ahmed Baraka

Objectives

By the end of this lecture, you should be able to:

- Query from a table in multiple containers
- Use the CONTAINERS clause with DML statements
- Specify the CONTAINERS DEFAULT TARGET
- Use Data Pump utility in a multitenant container
- Apply audit policies in a multitenant container

Querying from Multiple PDBs

- Use the CONTAINERS clause

```
SELECT op_id, op_user, op_desc, CON_ID, CON$NAME, CDB$NAME
FROM  CONTAINERS(APP_AUDIT);
 .. WHERE CON_ID IN (3,4); .. WHERE CON$NAME='PDB1';
```

- Prerequisites:
  - Beside the PDBs, the tables or views must exist in the root
  - The tables or views must be owned by the common user issuing the statement
  - Table definitions must match in all the PDBs (except if extra columns exist)

Querying a Table Owned by Local Users Across All PDBs

1. Connect to each PDB as a common user, and create a view accessing the local table:

```
CREATE OR REPLACE VIEW app_users AS SELECT * FROM hr.app_users;
```

2. Login to the root as the common user
3. Create an empty table of the same structure as the view
4. Query the views from all the PDBs using the CONTAINERS clause:

```
SELECT * FROM CONTAINERS(app_users);
```

Using Query Hints

- Be default, it applies only to the last aggregation step
- To push the hint to each PDB:

```
SELECT /*+ CONTAINERS(DEFAULT_PDB_HINT='FULL')*/
*
FROM CONTAINERS(APP_AUDIT) WHERE CON_ID IN (3,4);
```

Using the CONTAINERS clause with DML

- Use the CONTAINERS clause
- Same prerequisites of querying from PDBs apply
- Two options:
  - If no container specified, the default current container is affected.

```
UPDATE CONTAINERS(MYTABLE) SET .. ;
```

  - Specify the container with an equality condition on CON_ID

```
UPDATE CONTAINERS(MYTABLE) SET .. WHERE  CON_ID=3;
```

```
UPDATE CONTAINERS(MYTABLE) SET .. WHERE  CON_ID IN (3,4);
UPDATE CONTAINERS(MYTABLE) SET .. WHERE  CON_ID > 3;
```

Using the CONTAINERS clause with DML Examples

```
INSERT INTO CONTAINERS(APP_USERS)(USER_ID,USER_NAME,CON_ID)
VALUES ( 2,'USER 2', 3) ;
```

```
UPDATE CONTAINERS(APP_USERs) SET user_name ='updated'
WHERE  USER_ID=2 AND CON_ID=3;
```

```
DELETE CONTAINERS(APP_USERs) WHERE  USER_ID=2 AND CON_ID=3;
```

Specifying the CONTAINERS DEFAULT TARGET

- To specify the default container for DML statements in a CDB:

```
ALTER DATABASE CONTAINERS DEFAULT TARGET = (PDB2);
```

- The following DML will apply in PDB2:

```
UPDATE CONTAINERS(APP_USERs) SET user_name ='updated' ;
```

- To know the value of the property:

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


- To reset the property:

```
ALTER DATABASE CONTAINERS DEFAULT TARGET = NONE;
```

Using Data Pump with PDBs

- Export from a PDB and import into a PDB within the same CDB or from different CDBs
- Export from a PDB into a non-CDB, or vise versa
- CDB-wide Data Pump export or import is not supported
- Between a PDB and non-CDB, the following are supported:
  - Full database conventional or transportable export/import
  - Conventional or transportable tablespace export and import
  - Schema/Table export and import

Exporting from non-CDB and Importing into PDB: Example

1. Export non-CDB with FULL clause:

```
$ expdp system@HRDB FULL=Y DUMPFILE=hrdb.dmp
```

2. Create the data pump directory in the target PDB
3. Copy the dump filet to the directory
4. Create the tablespaces (if they are not there)
5. Import into the PDB with FULL option:

```
$ impdp system@HRPDB FULL=Y DUMPFILE=hrdb.dmp
```

Common Users Issues in Data Pump

Issue	Possible Resolution
Between PDBs: common user exists in the source PDB but does not exist in the destination PDB	- create the common user in the root - have the common user re-created as local users using the clause REMAP_SCHEMA=C##xxx :local_user_name
From a PDB to a non-CDB: common users will not be created in the non-CDB	- have the common user re-created as local users using the clause REMAP_SCHEMA=C##xxx :local_user_name

Auditing in CDB and PDBs

1. Connect to the root or the specific PDB
2. Create the audit policy:

```
CREATE AUDIT POLICY audit_changejob
 ACTIONS EXECUTE, GRANT ON hr.change_job_proc;
```

3. Enable the audit policy:

```
AUDIT POLICY audit_changejob;
```

4. View the enabled audit policies:

```
SELECT POLICY_NAME FROM AUDIT_UNIFIED_ENABLED_POLICIES;
```

Viewing the Audit Records

View	Description
UNIFIED_AUDIT_TRAIL	Displays all audit rows of the container you are connected to.
CDB_UNIFIED_AUDIT_TRAIL	Retrieves all audit rows of the CDB

```
SELECT CON_ID, DBUSERNAME, ACTION_NAME, EVENT_TIMESTAMP FROM
CDB_UNIFIED_AUDIT_TRAIL
ORDER BY 1,2,4 DESC;
```

Summary

In this lecture, you should have learnt how to:

- Query from a table in multiple containers
- Use the CONTAINERS clause with DML statements
- Specify the CONTAINERS DEFAULT TARGET
- Use Data Pump utility in a multitenant container
- Apply audit policies in a multitenant container