Becoming a Multitenant DBA

Arup Nanda
Longtime Oracle DBA

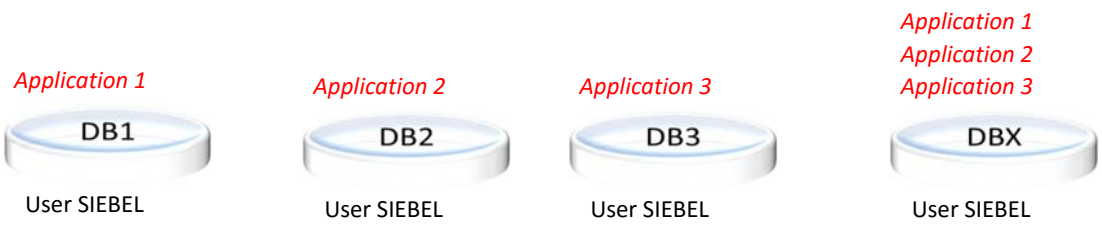
Three Big Questions

1. What do I need to learn *extra*?
2. Do I *need* to change anything?
3. *May* something break?

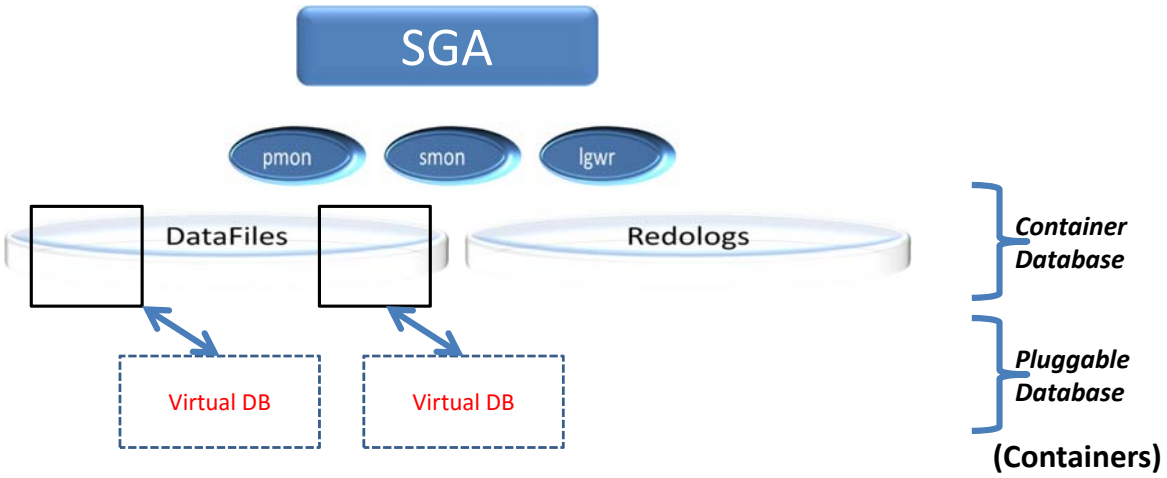
Agenda

- Quick primer on Oracle Multitenant
- Developer Activities Affected
- DBA Activities Affected
- Tips and Tricks

Vexing Problem of Database Consolidation



Enter: Pluggable Database



DBA_USERS

```
SELECT 1 AS CON_ID, NAME
FROM USER$@cdb
UNION ALL
SELECT 2 AS CON_ID, NAME
FROM USER$@pdb1
UNION ALL
SELECT 3 AS CON_ID, NAME
FROM USER$@pdb2
```

Root
CON_ID=1

PDB2
CON_ID=3

PDB3
CON_ID=4

What's Different

- CDB specific:
  - Alert Log
  - Redo Logs
  - Undo Tablespaces
  - SGA
  - ADR (Automatic Diagnostic Repository)
  - Characterset
  - Block size
  - Most pfile parameters
- PDB specific:
  - Additional datafiles (including system)
  - Some PDB-specific parameters

Parameters can be different

- View V$PARAMETER column ISPDB_MODIFIABLE shows if a parameter is modifiable

```
select name, value from v$parameter where
ispdb_modifiable = 'TRUE'
```

- Example:
  - `parallel_degree_policy` is PDB modifiable
  - `result_cache_max_size` is not PDB modifiable
- PDB does not have a SPFILE
- These parameters are stored in `PDB_SPFILE$`

Creating PDBs

- (In CDB) Logon to SQL*Plus as SYSDBA

```
SQL> create pluggable database PLUG1 admin user plug1admin
identified by plug1admin;
```


Basics of PDBs

Is this a CDB? Or a non-CDB?

```
SQL> select cdb from v$database;

CDB
---
YES
```

How many PDBs?

```
SQL> select name from v$pdbs;

NAME
---------------
PDB$SEED
PDBORCL
PLUG1
```

```
SQL> desc v$pdbs
 Name                          Null?    Type
 ----------------------------- -------- ---------------
 CON_ID                                 NUMBER
 DBID                                   NUMBER
 CON_UID                                NUMBER
 GUID                                   RAW(16)
 NAME                                   VARCHAR2(30)
 OPEN_MODE                              VARCHAR2(10)
 RESTRICTED                             VARCHAR2(3)
 OPEN_TIME                              TIMESTAMP(3) WITH TIME Z
 CREATE_SCN                             NUMBER
 TOTAL_SIZE                             NUMBER
 BLOCK_SIZE                             NUMBER
 RECOVERY_STATUS                        VARCHAR2(8)
 SNAPSHOT_PARENT_CON_ID                 NUMBER
```

Service Name

- Creates a default service in the database (CDB) with the same name as PDB
- Listener listens to this service:

```
$ lsnrctl status

…
Service "plug1" has 1 instance(s).
  Instance "cdborcl1", status READY, has 1 handler(s) for this
service...
```

- But that service is not *defined* in the database

```
SQL> show parameter service
NAME           TYPE  VALUE
-- ----------- ----- -------
service_names string CDBORCL
```


How do you Connect to a PDB?

- By default it connects to the “Root” container.

```
$ sqlplus / as sysdba
```

- Three ways to connect. First approach:

The SET CONTAINER Clause

```
SQL> alter session set container = PLUG1;
```

Connect to Service Name

Put in TNSNAMES.ORA file, SERVICE_NAME = PLUG1

```
PLUG1 =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = TCP)(HOST = host1)(PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = PLUG1)
    )
  )
```

```
$ sqlplus scott/tiger@plug1
```


Connecting from Applications

- Connect through a connect string:
  `$ sqlplus scott/tiger@mydb`
  where `mydb` is a connect string in `TNSNAMES.ORA`

before

```
mydb =
  (DESCRIPTION =
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=host1)
      (PORT=1521)
    )
    (CONNECT_DATA =
      (SID = MYSID)
    )
  )
```

after

```
mydb =
  (DESCRIPTION =
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=host1)
      (PORT=1521)
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PLUG1)
    )
  )
```

When Connect String is not Present

What to do when client connect on the server directly?

```
sqlplus scott/tiger
```

Put in TNSNAMES.ORA file, SERVICE_NAME = PLUG1

Set TWO_TASK environment variable

```
$ export TWO_TASK=PLUG1
$ sqlplus scott/tiger
```


Which PDB am I in?

- From SQL*Plus

```
SQL> show con_name
```

- From any session

```
sys_context('USERENV','CON_NAME')
```

DBA views

At root

```
SQL> select tablespace_name
from dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
UNDOTBS2
UNDOTBS3
UNDOTBS4
NEWURBANCODE
```

At PLUG1

```
SQL> select tablespace_name from
dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
TEMP
URBANCODE
URBANCODE1
```


All PDB Views

- Prefixed with CDB_ instead of DBA_

```
select tablespace_name from
DBA_TABLESPACES

TABLESPACE_NAME
------------------------------
NEWURBANCODE
SYSAUX
SYSTEM
TEMP
UNDOTBS1
UNDOTBS2
UNDOTBS3
UNDOTBS4
USERS
```

```
select tablespace_name from CDB_TABLESPACES

TABLESPACE_NAME
------------------------------
NEWURBANCODE
SYSAUX
SYSAUX
SYSAUX
SYSTEM
SYSTEM
SYSTEM
TEMP
TEMP
TEMP
UNDOTBS1
UNDOTBS2
UNDOTBS3
UNDOTBS4
URBANCODE
URBANCODE
URBANCODE1
URBANCODE1
USERS
```

CON_ID column shows the container the data belongs to.

More Space

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PLUG1                          READ WRITE NO
```

- Two additional containers

V$ Views

- Some V$ Views show the same data regardless of where you are connected
  - Examples:
    - V$DATABASE
    - V$LOGFILE
- But most show values specific for the PDB  IMPORTANT

At root

```
select value from v$sysstat s,
v$statname n
where n.name = 'parse time cpu'
and n.statistic# = s.statistic#;

     VALUE
----------
     86637
```

At plug1

```
select value from v$sysstat s,
v$statname n
where n.name = 'parse time cpu'
and n.statistic# = s.statistic#;

     VALUE
----------
        49
```

Opening and closing PDBs

- Open in current instance only:
```
SQL> alter pluggable database PLUG1 open;
```
- Open in current instance read only:
```
SQL> alter pluggable database PLUG1 open read only;
```
- Open all PDBs in current instance only:
```
SQL> alter pluggable database all open;
```
- Open in all instances:
```
SQL> alter pluggable database PLUG1 open instances=all;
```
- Or, you can be in the PDB
```
SQL> alter session set container = plug1;
SQL> startup
```


Shutdown PDBs

- Shutting down PDBs do not shut down the CDB
- Close in current instance only:
```
SQL> alter pluggable database PLUG1 close;
```
- Close in current instance immediately:
```
SQL> alter pluggable database PLUG1 close immediate;
```
- Close all PDBs in current instance only:
```
SQL> alter pluggable database all close;
```
- Close in all instances:
```
SQL> alter pluggable database PLUG1 close instances=all;
```
- Or, you can be in the PDB
```
SQL> alter session set container = plug1;
SQL> shutdown [immediate]
```

Individualized Instances of PDBs

- PDBs can be opened on selected instances

```
select name, inst_id, OPEN_MODE
from gv$pdbs;

NAME                           INST_ID OPEN_MODE
------------------------------ ---------- ----------
PDB$SEED                            1 READ ONLY
PDB$SEED                            4 READ ONLY
PLUG1                               1 READ WRITE
PLUG1                               4 MOUNTED
PLUG2                               1 READ WRITE
PLUG2                               4 MOUNTED
SARPRD                              1 READ WRITE
SARPRD                              4 READ WRITE
```

Note: PLUG1 is opened on one instance and just mounted on the other.

Service Name

- Default service created: PDB name
- You can create additional services

```
select name, con_name
from v$active_services

NAME             CON_NAME
---------------- ----------
plug1            PLUG1
pdborcl          PDBORCL
cdborclXDB       CDB$ROOT
cdborcl          CDB$ROOT
SYS$BACKGROUND   CDB$ROOT
SYS$USERS        CDB$ROOT
```

Tip: Do NOT Use Default Service

- Default service
  - is not managed by srvctl
  - can't be brought down
  - starts as soon as the PDB comes up
  - If you move the PDB to a different CDB (with a different name), you can use the same service name; so apps do not need to change
- How to create a service for that specific PDB

```
$ srvctl add service -d cdborcl -s newplug3 -pdb plug3 -
preferred "cdborcl1" -available "cdborcl2"
```


Bug

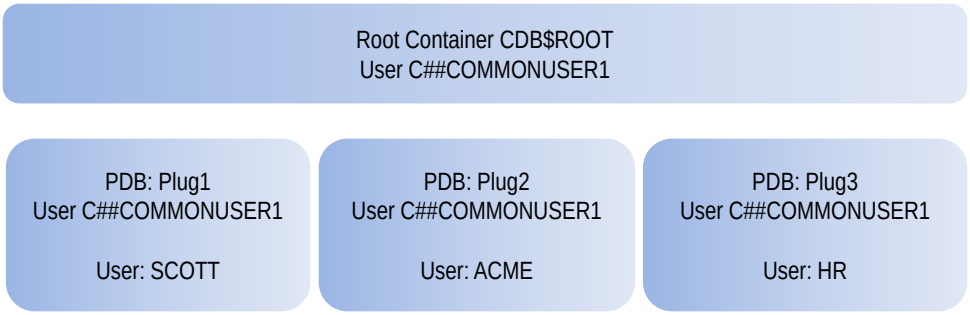
- Default service pointed to root container

```
select name, con_name
from v$active_services

NAME             CON_NAME
---------------  ---------
plug1            PLUG1
pdborcl          PDBORCL
cdborclXDB       CDB$ROOT
cdborcl          CDB$ROOT
SYS$BACKGROUND   CDB$ROOT
SYS$USERS        CDB$ROOT
```

```
select name, con_name
from v$active_services

NAME             CON_NAME
---------------  ---------
plug1            CDB$ROOT
pdborcl          PDBORCL
cdborclXDB       CDB$ROOT
cdborcl          CDB$ROOT
SYS$BACKGROUND   CDB$ROOT
SYS$USERS        CDB$ROOT
```

This can't be altered. So all apps pointing to the PLUG1 service will point at the root container, resulting in not finding the right data, users, etc.

Local/Common User



Database Parameter `common_user_prefix = C##`

Cloning

- Clone from an existing PDB:
  1. Bring up the source PDB in read only mode
  2. `SQL> create pluggable database plug2 from plug1;`
  3. Creates the default service plug2 automatically.
  4. Create a non-default service name for this PDB
- Can clone a remote PDB too

  `SQL> create pluggable database plug3 from plug1@mylink;`
- Can clone only metadata

  `SQL> create pluggable database plug4 from plug1 no data;`

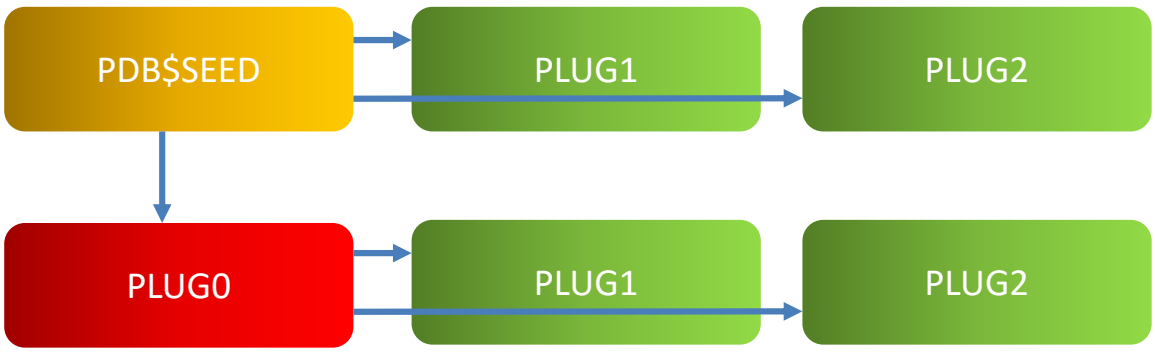
Subsetting

- You can clone only a few user tablespaces

```
SQL> create pluggable database plug3 from plug1
user_tablespaces = ('URBANCODE1');
```

- Very useful in creating subsets for prod-to-non-prod moves, or dividing too large PDBs into smaller ones.

Tip for Creation of PDBs



Transporting

1. If the PDB is open, you should close it.
```
SQL> alter pluggable database plug1 close;
```
2. Create the meta-information on the PDB in an XML file.
```
SQL> alter pluggable database plug1 unplug into
'plug1_meta.xml';
```
3. Copy this file and all the datafiles of plug1 to the target server.
4. On the target server, connect to the CDB with SYSDBA privilege
```
$ sqlplus sys/oracle as sysdba
```
5. Execute this:
```
SQL> create pluggable database newplug1 using
'plug1_meta.xml';
```


History of PDBs

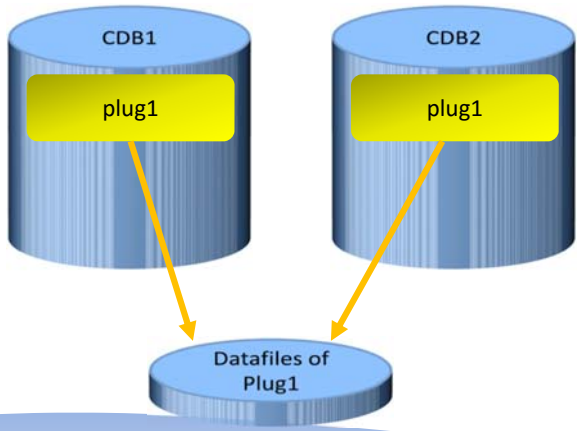
- View CDB_PDB_HISTORY shows all operations

```
select pdb_name, operation, op_timestamp, cloned_from_pdb_name
from cdb_pdb_history;

PDB_NAME   OPERATION        OP_TIMEST CLONED_FRO
---------- ---------------- --------- ----------
PDB$SEED   UNPLUG           07-JUL-14
PDB$SEED   PLUG             11-OCT-14 PDB$SEED
PLUG1      CREATE           01-JUN-16 PDB$SEED
PDB$SEED   UNPLUG           07-JUL-14
PDB$SEED   PLUG             11-OCT-14 PDB$SEED
PLUG1      CREATE           01-JUN-16 PDB$SEED
PLUG2      CLONE            10-JUN-16 PLUG1
```

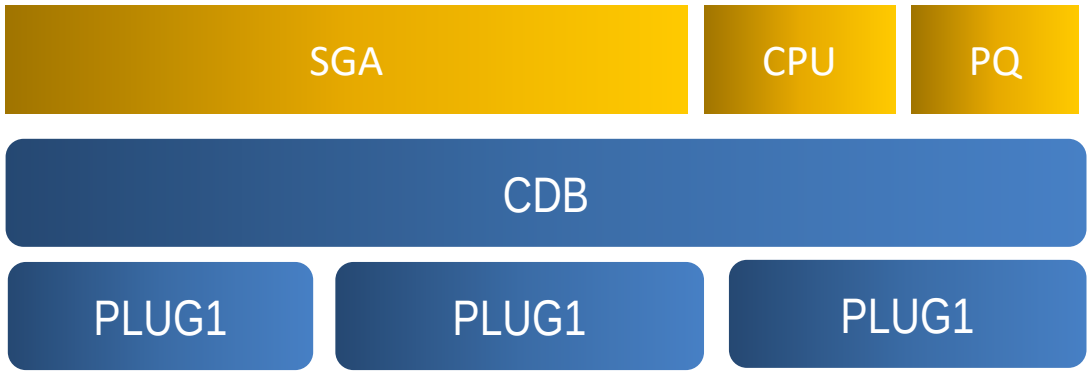
DB Links

- No Change. `SQL> create database link plug1 using 'plug1'`
- Can connect to
  - Root container
  - PDB
- Can't do "`alter session set container ...`" so root-link is not practical
- But useful for V$ views

Transportable Read Only Database



Will they all play along nicely?



Resource Manager

```
dbms_resource_manager.create_cdb_plan_directive (
                plan                  => 'dayplan1',
                pluggable_database    => 'plug1',
                shares                => 2,
                utilization_limit     => 100,
                parallel_server_limit => 100
        );

dbms_resource_manager.create_cdb_plan_directive (
                plan                  => 'dayplan1',
                pluggable_database    => 'plug2',
                shares                => 1,
                utilization_limit     => 50,
                parallel_server_limit => 50
        );
```

Data Pump

- Remember, directories are visible only in a PDB
  - So, in expdp or impdp, use user/pw@plug1
- SERVICE_NAME parameter doesn't help

```
$ expdp u/p directory=tmp_dir cluster=no
service_name=plug1 schemas=UCRELEASE
```

- OR, use TWO_TASK

```
$ export TWO_TASK=plug11
$ expdp u/p …
```


Backup/Recovery

- You can backup the entire CDB
  - RMAN> connect target /
  - RMAN> backup database;
- OR, individual PDBs
  - RMAN> backup pluggable database plug1;
- OR, use RMAN directly
  - RMAN> connect target=sys/oracle@plug1
  - RMAN> backup database;
- You can restore entire CDB or individual PDBs

Point in Time Recovery of PDB

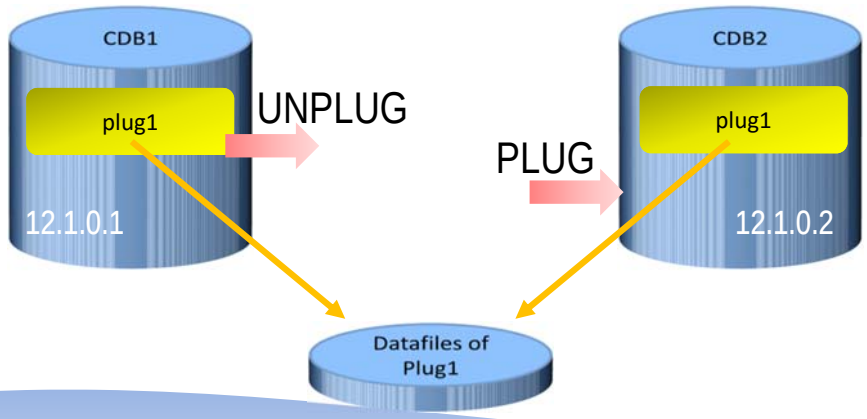
- You can do PITs of individual PDBs leaving the rest in their place.

```
RMAN> run {
2>       set until time '08-MAR-16';
3>       restore pluggable database plug1;
4>       recover pluggable database plug1;
5> }
```

- Creates an auxiliary instance, creates that PDB only and plugs it in.

Upgrade

- Update the entire CDB. All individual PDBs will be upgraded
- Transport a specific PDB from one CDB to another at a higher version



Golden Gate

As Common User:

```
SQL> alter system set enable_goldengate_replication = true;
```


Golden Gate Setup

- In CDB
- Create a common user

```
SQL> create user C##GGADMIN identified by ggadmin;
SQL> exec dbms_goldengate_auth.grant_admin_privilege
('C##GGADMIN', container => 'ALL')
SQL> grant dba to C##GGADMIN container = ALL;
```

- Repeat for all PDBs.

```
SQL> alter session set container = plug1;
SQL> alter pluggable database add supplemental log data;
```

Golden Gate Configuration (ggsci)

```
GGSCI (source) 1> dblogin userid acme@plug1, password acme
GGSCI (source as acme@CDB1/PLUG1) 2> add schematrandata
plug1.acme
GGSCI (source) 3> edit params extora
   EXTRACT EXTORA
   USERID C##GGADMIN@CDB1, password ggadmin
   RMTHOST remotehost1, MGRPORT 7809
   RMTTRAIL ./trails
   DDL INCLUDE MAPPED
   LOGALLSUPCOLS
   UPDATERECORDFORMAT COMPACT
   TABLE PLUG1.ACME.*;
   TABLE PLUG2.ACME.*
```


Naming Convention Tip

- CDB Names
  - C<Seq#><AppName>
  - C2SAR
- PDB Name
  - Make it unique across the enterprise
  - P1SAR
  - Makes it easy to plug in to any CDB without renaming
  - Allows you to open multiple PDBs read only

Caveats

- Non-CDBs may be deprecated
- Some features not supported on CDBs (as of 12.1)
  - Heat Map, Automatic Data Optimization
  - Change Notification
  - Client Side Result Cache
- Real Application Testing only for CDB; not PDB

Summary

- CDB/PDBs are transparent to the applications
- Most functions for the DBA stay the same without change
  - The scripts will work
  - Backup/recovery works
- Pay close attention to dynamic performance views as their meanings could change
- DBA_ views shows PDB specific data. CDB_ views show all PDBs
- Create a non-default service name for the PDB; do not use the default one.

Thank You!

Blog: arup.blogspot.com
Tweeter: @ArupNanda
Facebook.com/ArupKNanda