Oracle 12.2 New features Multi-Tenancy

23rd of October 2017

Martin Jensen - Nordea

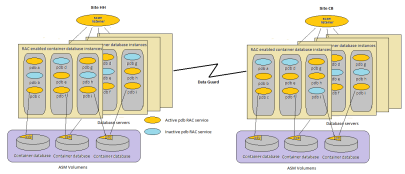
Agenda

- Overview
- Naming Standard
- Create Pluggable database / Cloning
- Create / start services for specific PDBs
- Backup / Restore
- Pluggable Database Flashback
- DataGuard setup
- Resource Profiles
- Security
- MAX_STRING_SIZE = STANDARD
- More Database Parameters
- AWR for individual Pluggable Databases

Overview

Oracle 12.2 have removed most restrictions from 12.1 for Multi-Tenancy, so now is the time where Nordea can fully utillize this new feature area, where Nordea currently holds an ULA license.

Basically a few large rac-enabled container databases are being populated with a number of pluggable databases.



Naming Standard

Container databases are typical referring to an SLA (Business criticality) as well as the category and site:

Physical_container_name ::= 'C'<business_criticality><sequence><category><standby_no><site>

Business_criticality ::= 'B' | 'H' | 'S' – for BCA, HA or Standard

Sequence ::= nnn – a 3 digit number

Category ::= 'S' | 'T' | 'D' | 'P' – Sandbox, Test, Demo or Production

Standby_no := n – a one digit number (not 0) to allow more standby databases.

Site ::= 'C' | 'H' | 'O' – CB, HH or OE

As an example CH003T2H is a container database covering HA test databases in HH, and has the number 003.

Naming Standard cont.

A pluggable database name is more like a logical construction which is the same on different cluster databases in a standby environment. As a convenient step, the category of the database is maintained at the pluggable database level even though it is actually also inherited from the container level. Because we cannot use the pluggable database name as a service for the application, we have some freedom naming the actual pluggable database.

Pluggable_name ::= 'P'<logical_database_name><clone_no><category>

Clone_no ::= nn – a two digit clone number of a pluggable – the first has the number '01'

As an example PALFA01S is a Pluggable sandbox database from the logical ALFA application system, and it's the first one.

Apart from the default service (equal to the pluggable_name) at least two dedicated services are created:

Application_service ::= <logical_database_name> -- used in TNS for applications accessing this database

Read_only_service ::= <logical_database_name>'_ro'

Create Pluggable Database / Cloning

A pluggable database may be created in a number of different ways

- **Creating a new Pluggable database**
- **Through a PLUGIN – potentially following a PLUGOUT**
- **Copy a Pluggable database / Clone**
- **Refresh a PDB Clone**
- **Relocate a PDB from one container to another**
- **Create a thin Snapshot Clone of a PDB**

Nordea

Create Pluggable Database / Cloning

Creating a new Pluggable database

Fron the root of a container database (CDB$ROOT) it is possiblle to create a new pluggable database, internally using the SEED database as the template:

```
CREATE PLUGGABLE DATABASE <pdb_name>
   ADMIN USER <pdb_admin_name> IDENTIFIED BY <passwd> ROLES=(CONNECT);

ALTER PLUGGABLE DATABASE <pdb_name> OPEN INSTANCES = ALL;

ALTER SESSION SET CONTAINER = <pdb_name>;

GRANT CONNECT, RESOURCE, DBA TO <pdb_admin_name>;
```

In Nordea the <pdb_admin_name> account name is chosen to be 'PDB_ADMIN' with connect, resource, dba roles.

Nordea

Create Pluggable Database / Cloning

Through a PLUGIN – potentially following a PLUGOUT

ALTER PLUGGABLE DATABASE <pluggable_database_name> CLOSE immediate instances = all;

ALTER PLUGGABLE DATABASE <pluggable_database_name> UNPLUG INTO '<pdb_name>.xml';

-- located in $ORACLE_HOME/dbs/<pdb_name>.xml

Now, let's drop the database (but keep the datafiles), and plugin the pdb into the same container again. After having validated that we are all green.

drop pluggable database <pdb_name> KEEP DATAFILES;

Nordea

Create Pluggable Database / Cloning

Through a PLUGIN – potentially following a PLUGOUT – Check if the plugged out database may be plugged in:

```
set serveroutput on
DECLARE
  compat BOOLEAN := FALSE;
BEGIN
  compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
    pdb_descr_file =>'<pdb_name>.xml',
    pdb_name => '<pdb_name>');
  if compat then
    DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? YES');
    else DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? NO');
  end if;
end;
/
```

Is pluggable compatible? YES

Nordea

Create Pluggable Database / Cloning

Through a PLUGIN – potentially following a PLUGOUT – Plug in:

Potential violations may be found here in the *pdb_plug_in_violations* view. If no violations are found the database may be plugged in and the related service started:

```
select message, action from pdb_plug_in_violations
where name='<pdb_name>'
order by message;
```

create pluggable database <pdb_name> using <pdb_name>.xml' NOCOPY;

srvctl start service -service <pdb_service_name> -db <container_database_name>

When a pluggable database is first created it is in *MOUNTED* mode before it is opened. Please consider to adjust some of the database parameters before the pluggable database is opened, such as setting *open_links* to 0 in order to control if any local materialized views should register remote materialize view logs

Alter system set open_links=0 scope=spfile sid='*';

Nordea

Create Pluggable Database / Cloning

Copy a Pluggable database from one container database to another.

Version 12.2 does allow a clone of a pluggable database to be created from one container database to another (with some restrictions) even as the source pluggable database is in *READ WRITE* mode and being used. The create pluggable database will (when it's opened) have a consistent content from the source pluggable from when all the data have been copied over to the new pluggable.

CREATE USER c##remote_clone_user IDENTIFIED BY <passwd> CONTAINER=ALL;

GRANT CREATE SESSION, CREATE, SYSOPER PLUGGABLE DATABASE TO c##remote_clone_user CONTAINER=ALL;

Create a database link in the target_cdb container pointing to the source_cdb, and test the link. Note that there seems to be an issue using the scan listener in this database link, which is why the VIP address is used (SR 3-15433875101: PDB Hot Clone fails with TNS error).

```
CREATE DATABASE LINK clone_link CONNECT TO c##remote_clone_user IDENTIFIED BY
    <passwd> USING
    '(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=<host>-
    vip.oneadr.net)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=<source_cdb>.oneadr.ne
    t)))';
```

Nordea

Create Pluggable Database / Cloning

Copy a Pluggable database from one container database to another – do the copy

Create the target_pdb as a clone of the source_pdb, without having to switch the source_pdb into *READ ONLY* mode, and take advantage of our usage on Oracle Managed Files (OMF).

CREATE PLUGGABLE DATABASE <target_pdb> FROM <source_pdb>@clone_link;

ALTER PLUGGABLE DATABASE <target_pdb> OPEN INSTANCES = (<list of instances>);

Check for possible errors:

```
select *
from PDB_PLUG_IN_VIOLATIONS
where name = '<target_pdb>'
order by time;
```

There will probably be some "Database option <option> mismatch: PDB installed version 12.2.0.1.0. CDB installed version NULL" errors, which may be ignored due to the following MOS: "OPTION WARNING Database option mismatch: PDB installed version NULL" in PDB_PLUG_IN_VIOLATIONS (Doc ID 2020172.1)

Nordea

Create Pluggable Database / Cloning

Refresh a PDB Clone

It is possible to create the clone of a pdb, and then at a later stage refresh the clone to get the latest content of the source pluggable database pushed to the clone. The procedure is very similar with normal cloning.

CREATE PLUGGABLE DATABASE <target_pdb> FROM <source_pdb>@clone_link REFRESH MODE MANUAL;

This may be a lengthy operation as large volumes of data would potentially need to be copied from one cluster to another. On the positive side however, the source pdb is still servicing the applications. So after the initial cloning has finished, it may be preferable to do a refresh to know more about the actual content in the cloned database.

ALTER SESSION SET CONTAINER = <target_pdb>;

ALTER PLUGGABLE DATABASE <target_pdb> REFRESH;

Before the new clone can be opened it need to be disconnected from the refresh mode:

ALTER PLUGGABLE DATABASE <target_pdb> REFRESH MODE NONE;

ALTER SESSION SET CONTAINER = CDB$ROOT;

ALTER PLUGGABLE DATABASE <target_pdb> OPEN INSTANCES = (<list of instances>);

Nordea

Create Pluggable Database / Cloning

Relocate a PDB from one container to another

It is possible to combine the previous *hot clone* mechanism to actually move a pluggable database from one container database to a different one, even in different clusters using the same kind of database link (clone_link) as used before:

CREATE PLUGGABLE DATABASE <target_pdb> FROM <source_pdb>@clone_link RELOCATE;

This will leave the <target_pdb> in the *RELOCATING* state, with all content from the <source_pdb> from the time the relocate operation ends. In order to open the relocated pdb, transfer the last minute changes and drop the old pdb, we need to execute the open request:

ALTER PLUGGABLE DATABASE <target_pdb> OPEN INSTANCES = (<list of instances>);

Nordea

Create Pluggable Database / Cloning

Create a thin Snapshot Clone of a PDB

It is possible to create pdb clones as thin (storage wise) pluggable databases, where the cloned pdb literally only have very little storage claims in the beginning of its life. As the original and the cloned pdb evolves over time, the storage claim of the cloned pdb will increase.

CREATE PLUGGABLE DATABASE <pdb_clone_name> FROM <pdb_name> SNAPSHOT COPY;

It is however required to use the so called 'sparse disk group' to have this working.

TBD

Nordea

Create / start services for specific PDBs

In order to avoid connection errors (ORA-01033: ORACLE initialization or shutdown in progress) when connecting to a pdb in a RAC, where some of the instances is up and some is only MOUNTED, we need to create explicit services for each PDB, see MOS 1998112.1: Connecting To a 12c RAC Pluggable Database Intermittently Fails With ORA-1033".

srvctl add service -service <explicit_service_name> -db <container_db_name> -role primary -pdb <pdb_name> -preferred "<list of cluster db instances>"

srvctl start service -service <explicit_service_name> -db <container_db_name>

srvctl status service -service <explicit_service_name> -db <container_db_name> -v

srvctl config service -service <explicit_service_name> -db <container_db_name> -v

add entry to tnsnames.ora using the explicitely create service name

Nordea

Backup / Restore

The strategy is to continue to run the Nordea backup procedures at the (container) database level using RMAN, and then to restore pluggable databases (if need be) from the container database backup. This way a new pluggable in a container database will automatically inherit the backup characteristics. And a pluggable database may be restored without disturbing other pluggable databases in the same container database.

Actually a full container database backup will group tablespaces from the same PDB into the same backup pieces in order to make a pdb restore faster. Newly created pluggable databases will have their first backup at the following incremental level 1 backup. Archivelog backups will not cover these newly created PDBs.

The general backup procedure remains at the container database level, where backups are taken locally on the site where the physical database is located:

- Incremental level 0 backup once every weekend
- Incremental level 1 backup once every day
- Archivelog backup every hour (or second hour)
- Block-change-tracking enabled
- RMAN compression in use

Nordea

Pluggable Database Flashback

From version 12.2 restore points may be used at the pluggable database level.

CREATE RESTORE POINT <restore_point_name>;

< do some work >

As we want to keep the pluggable instances exactly at the same positions and state as before a close, the following *save state* the following statement should be considered.

ALTER PLUGGABLE DATABASE <pdb_name> SAVE STATE;

Closing a PDB with non-default services may hang on oracle process connections

srvctl stop service -db <container_database_name> -service <pdb_service_name> -f

ALTER PLUGGABLE DATABASE <pdb_name> CLOSE IMMEDIATE INSTANCES = ALL;

FLASHBACK PLUGGABLE DATABASE <pdb_name>TO RESTORE POINT <restore_point_name>;

ALTER PLUGGABLE DATABASE <pdb_name> OPEN RESETLOGS;

srvctl start service -db <container_database_name> -service <pdb_service_name>

Nordea

DataGuard setup

At the Container database level *force logging* must be enabled. This setting is automatically inherited to all the pluggable databases in that container, and cannot be overwritten at the pluggable level.

Two special init.ora parameters are useful when pdbs are cloned or relocated into a container with standby: *standby_pdb_source_file_dblink* and *standby_pdb_source_file_directory*

TBE

Nordea

Resource Profiles

We define 4 different *resource profiles* in order to span PDBs from the very large (one pdb in the cdb) to the relatively small where one Container database instance may hold 64 PDBs in total. And we think 4 different resource profiles are manageable:

Resource Profile	SGA Target	SGA Min size	PGA Target	PGA Limit	Cpu Count	Max pdbs per cdb inst
rp1	1162M	581M	150M	450M	1	64
rp2	4650M	2325M	602M	1806M	2	16
rp3	18600M	9300M	2411M	7233M	4	4
rp4	74401M	37200M	9644M	28932M	14	1

SGA_MIN_SIZE must be equal or less than half the *SGA_TARGET*.

Due to Bug 26160154 the usage of the SGA for a pdb may be bigger that the *SGA_TARGET* for that pdb.

PGA_AGGREGATE_TARGET is 3 times lower than *PGA_AGGREGATE_LIMIT*

Nordea

Resource Profiles cont.

It is possible to limit the IOPS and Megabytes per second for each of the PDBs – obviously we need to test and verify which limit values are reasonable in order to utilize the underlying IO capacity the best, to be balanced with reasonable isolation between PDBs. Here is the initial settings:

Resource Profile	MAX_IOPS	MAX_MBPS
rp1	6500	70
rp2	25000	280
rp3	100000	1120
rp4	400000	4500

Nordea

Resource Profiles cont.

It is recommended to create each pluggable database with a reasonable maximum size, and then to allow for extensions when extra storage capacity is needed and acknowledged.

Resource Profile	MAXSIZE in G	MAX_SHARED_TEMP_SIZE in G
rp1	100	5
rp2	500	25
rp3	1000	50
rp4	5000	250

When needed (possibly after an "ORA-65114: space usage in container is too high" error), this size may be increased like this from the actual pluggable database:

ALTER PLUGGABLE DATABASE <pdb_name> STORAGE (MAXSIZE <size>);

Similar settings may protect the local filesystems, covering audit and diag areas, using *MAX_AUDIT_SIZE* and *MAX_DIAG_SIZE*

Nordea

Security

Audit settings

Each pdb will inherit the basic security settings from the container-database (audit_trail which is set to XML, EXTENDED) – refer to the Oracle database audit log management document.

User accounts and Roles

It is possible to create container database wide user accounts and roles. Such phenomena need to be prefixed with 'C##' (default of the COMMON_USER_PREFIX parameter). Such common user accounts are immediately available in existing (and coming pluggable databases) in the container database and in the different pluggable databases. Such users may get different default tablespaces and other preferences.

Database Profiles

Besides the existing database profiles delivered from Oracle (DEFAULT and ORA_STIG_PROFILE) , we have in Nordea the TECHNICAL_USER_PROFILE and the PERSONAL_USER_PROFILE database profiles now called C##TECHNICAL_USER_PROFILE and the C##PERSONAL_USER_PROFILE at the Root container level.

From the single-tenant databases we are used to have these database profiles using the *NORDEA_VERIFY_FUNCTION* password validation function, which is now available at the root container level as well.

Nordea

Security – cont.

Consider the new lockdown profile from 12.2 to increase stability. Such a lockdown profile will limit the available features on some of the pluggable databases in the container database.

CREATE LOCKDOWN PROFILE ndld_profile;

ALTER LOCKDOWN PROFILE ndld_profile DISABLE FEATURE = ('EXTERNAL_FILE_ACCESS','JAVA_OS_ACCESS','LOB_FILE_ACCESS');

ALTER LOCKDOWN PROFILE ndld_profile DISABLE STATEMENT = ('alter database');

ALTER SESSION SET CONTAINER = <pdb_name>;

ALTER SYSTEM SET PDB_LOCKDOWN = ndld_profile;

Nordea

MAX_STRING_SIZE = STANDARD

From 12.1 it is possible to extend the limit for VARCHAR2 columns from 4000 to 32k bytes. This is done by changing the MAX_STRING_SIZE from STANDARD to EXTENDED.

Setting the MAX_STRING_SIZE to extended in a pluggable database in a RAC environment is not straight forward. One would need to follow note "How to Increase the Maximum Size of VARCHAR2, NVARCHAR2, and RAW Columns in 12C Database using MAX_STRING_SIZE? (Doc ID 1570297.1)" and disable the *cluster_database* to false for the cluster database during the upgrade on the pluggable database.

It is recommended to keep the default (STANDARD) for this parameter.

Nordea

More Database Parameters

In order to offer maximum isolation between pluggable databases in the same container database, and to support restore points at the pdb level, each of the pluggable database should use *LOCAL UNDO*. LOCAL UNDO is OFF as default in version 12.2. This is also a prerequisite for *hot cloning* and *relocation.*

```
ALTER DATABASE LOCAL UNDO ON;
select * from database_properties
where property_name = 'LOCAL_UNDO_ENABLED';
```

The clause "close instance all" does not work (Doc ID 2062080.1) as this feature does require *PARALLEL_FORCE_LOCAL=FALSE*, which we do not want in general for other reasons. So for the container database we set *PARALLEL_FORCE_LOCAL=FALSE*, and TRUE for pluggable databases.

ENABLED_PDBS_ON_STANDBY Should be set to "*" to cover that all PDBs in a container database should be covered by Dataguard.

Using Bigfile as Default

```
ALTER PLUGGABLE DATABASE <pdb_name> SET DEFAULT BIGFILE TABLESPACE;
```

Nordea

AWR for individual Pluggable Databases

It is possible to have snaps generated at the Pluggable database level in order to offer AWRs at the container as well as the pluggable database.

AWR_PDB_AUTOFLUSH_ENABLED changed from default FALSE to TRUE, at the container level in order to support automatic AWRs per pdb. It is also possible to set this parameter at the pdb level in order to select specific pdbs for AWR support.

AWR_SNAPSHOT_TIME_OFFSET change from 0 to 1000000 at the container level. Setting this parameter to 1000000 helps to create AWR snapshots with different offset based on database names and avoids CPU spike in the system.

Also on the pluggable database the following workload repository setting need to be applied in order to automate the snap generation each hour.

```
BEGIN
 DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS (
   retention => 11520          -- 8 days,
-- setting this value to 0 will disable the snap creation at the pdb level
 , interval => 60                 --  one hour
 , topnsql => 'DEFAULT'
 , dbid => <dbid for the pdb>
  );
END;
/
```

Nordea

Be careful out there,

Oracle SME's @nordea.com