Oracle Server 12c - Anger Management Guide

Hans Forbrich
Forbrich Consulting Ltd.
St. Albert, Alberta, Canada

Keywords: **DB12c, Features, Scripts**

Introduction

Oracle database 12 C introduces a number of new and exciting changes, especially in conjunction with the container database. A number of these changes have required a fundamental architectural revision. In order to accommodate this architectural difference, Oracle has lifted a long-standing practice of ensuring backward compatibility.

This paper explores some of the changes and the impacts of which we have become aware. We have found that some of these changes have been unexpected and have caused us to revisit a number of the scripts on which we have depended for a number of versions.

That is not to say that all scripts and environments will immediately break. However, being aware of these changes can make troubleshooting significantly the easier.

This is not a comprehensive list, however it does identify a number of the areas of functionality that we have had to investigate and correct.

Remember: The whole is still the sum of the parts ... know before you start

Oracle Database 12c includes the Multi-tenant architecture and capability, which has generated a lot of interest in part because it is so new. Oracle's Concepts documentation, which contains a very comprehensive description of the architecture can be found at http://docs.oracle.com/cd/E16655_01/server.121/e17633/cdbovrvw.htm#CEGDAADB

In summary, the architecture has been changed to allow multiple databases to run in one instance. This ist he exact opposite of RAC, which supports multiple instance for one database.

It is important to note that the multi-tenant architecture is included with the base Standard Edition and Enterprise Edition license. According to the online Licensing document, you must obtain additional license for the ability to have more than one container (other than root and seed).

As a result of this, we have been testing the database 12c Standard Edition in single container mode for one of our clients. So far we have been pretty satisfied with the compatibility and have not noticed any serious difficulties in this mode of operation.

However, when using multiple containers, in other words using the multitenant option for the enterprise edition, we have observed a number of significant changes that are caused us to slow down and study the impacts carefully.

One instance per host

The first of these impacts is that we truly should plan on a maximum of one database instance per host operating system.For the purposes of this discussion we will assume a Linux environment, although most of the principles are consistent across any UNIX, and many principles will be consistent across Windows as well.

In the past it was fairly common to have multiple instances on one host for variety of reasons including: maximizing use of the host hardware capabilities; and minimizing licenses required for the hardware.

Switching between instances was generally accomplished by changing our Oracle_SID environment variable, for example using the oraenv script provided by Oracle.

In the multitenant architecture, we have one instance and many databases plugged into that instance. This has some implications at installation and initial configuration time.

SGA allocation

Although the SGA will be generally smaller than the sum of the SGAs previously used, we need to plan ahead and size the SGA for the maximum number of databases expected throughout the lifetime. While it is very simple to snap in a new container without shutting down any of the other databases, it would cause an unacceptable outage to have to shut down all databases to reconfigure the SGA to accommodate a new container.

If the operating system supports DISM, then we must allocate MEMORY_ MAX_TARGET properly during initial root container creation. We can then throttle the actual memory usage using the MEMORY_TARGET parameter.

An alternative to this would be using the PGA_AGGREGATE_TARGET and the SGA_MAX_SIZE and SGA_TARGET.

In our test and dev environments, we have allocated the memory max target to approximately 80% of available RAM, and the memory target to approximately 70% of available RAM. So far this is working reasonably well and has avoided instance shutdowns.

Undo allocation

The multitenant environment uses one UNDO per instance to cover all possible databases. From what we have seen, the total UNDO usage is approximately the sum of the individual database UNDO activity. It is of course much easier to add disk to the undo tablespace on-demand, however it was a surprise to realize child being the UNDO tablespace would need to be.

It will become much more important when configuring multitenant environment to have a documented UNDO requirement that includes such parameters as: undo retention requirements; flashback query requirements; and of course transaction rates used in traditional calculation.

UNDO management is done from the root container, never from the pluggable database.

TEMP allocation

The multitenant architecture allows us to use one temporary tablespace, or tablespace group, for all databases within the instance. This is the default operation.

Similar to UNDO sizing considerations, the temporary tablespace size will need to accommodate all database activity, for all containers. However, again, the actual size can be adjusted at runtime without impacting the operation of individual databases or the entire instance.

However if necessary, a temporary tablespace can be created for any pluggable database.

Temporary table space management for the default, instance wide, table space is done from the root container. However, the management for a specific pluggable database, when used, is done from that specific container.

We believe that there is a possibility for confusion here, especially since temporary space management is often done under crisis conditions. Significant documentation, and an administrators logbook, will likely be very important for this aspect as well.

Summary

The multitenant architecture - while very interesting and, from initial investigation, very reliable - will require some significant changes in traditional DBA practices.

For example, memory allocation must be considered for the entire instance lifecycle, with all potential pluggable databases. This may have secondary impacts on shared memory configuration managed by the system administrator.

While the SYSTEM and SYSAUX tablespaces for each database are independent of the instance, the UNDO and TEMP are common to all databases in the instance. Depending on the configuration of the DBA team, managing UNDO and TEMP will have to be carefully coordinated.

Service, Service, Service

Back in Oracle8i, Oracle introduce the concept of database „Service“. Many of us initially looked at it as simply being a way to automatically register an instance with the listener.

However, a better way of looking at this „service" is as a virtual instance. A service can be turned off and on, can be relocated from one instance to another, and in the case of RAC it can be distributed across multiple instances. From Oracle10g (and perhaps earlier) to 11gR2, and Oracle instance could support approximately 120 services.

The concept of resource management within an instance is very closely tied to the concept of service. The DBA can create new services, remove, start and stop existing services.

With the multitenant architecture, each pluggable database has its own service. When a databases created or started, its services also started. And conversely when a database is stopped within the multitenant architecture, its services also stopped.

In the following listing, the instance cbd1 has several services, PDB1 and PDB2 being pluggable databases.

```
[oracle@localhost ~]$ lsnrctl status
LSNRCTL for Linux: Version 12.1.0.1.0 - Production on 05-NOV-2013 04:18:07

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 12.1.0.1.0 - Production
Start Date                04-NOV-2013 23:59:06
Uptime                    0 days 4 hr. 19 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter  File  /u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/localhost/listener/alert/log.xml
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=8080))(Presentation=HTTP)
(Session=RAW))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=localhost)(PORT=5501))
(Security=(my_wallet_directory=/u01/app/oracle/product/12.1.0/dbhome_1/admin/cdb1/xdb_wallet))
(Presentation=HTTP)(Session=RAW))

Services Summary...
Service "cdb1" has 1 instance(s).
   Instance "cdb1", status READY, has 1 handler(s) for this service...
Service "cdb1XDB" has 1 instance(s).
   Instance "cdb1", status READY, has 1 handler(s) for this service...
Service "pdb1" has 1 instance(s).
   Instance "cdb1", status READY, has 1 handler(s) for this service...
Service "pdb2" has 1 instance(s).
   Instance "cdb1", status READY, has 1 handler(s) for this service...
The command completed successfully
[oracle@localhost ~]$
```

ORACLE_SID

Traditionally in a clinic's or UNIX environment we set the ORACLE_SID environment variable to point to the instance that we wish to administer.

With a multitenant environment, this is still true. However accessing the database instance using a command like

```
sqlplus / as sysdba
```

will automatically bring us to the ROOT container. That is likely not the container that we wish to administer.

Applications cannot run in the root container. Users cannot be created in the root container. The root container is primarily used to: start and stop the instance; managed instance level resources such as memory, UNDO and TEMP; and create, start, stop, and drop pluggable databases.

In our experience, many scripts have been written to access the database instance strictly on the value of the Oracle_SID, using OS authentication. Most of these scripts will likely have to be revisited to ensure they work correctly.

TNSNAMES.ORA

One of the most frustrating things is when creating or dropping a pluggable database, the TNS names.Ora file is not updated automatically.

This means that the DBA creating a new pluggable database will need to add a TNS names.Ora entry such as the following:

```
PDB1 =
  (DESCRIPTION =
     (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
     (CONNECT_DATA =
       (SERVER = DEDICATED)
       (SERVICE_NAME = pdb1)
     )
   )
```

Once that entry has been created, access to the correct database would be by using the TNS alias. Note that by default in a Linux environment, the /etc/oratab is also not updated with the PDB name, and therefore the ORACLE_SID is set to point to the instance name and the TNS alias is used to get to the correct database.

```
[oracle@localhost ~]$ echo $ORACLE_SID
 cdb1
[oracle@localhost ~]$ sqlplus sys/oracle@pdb1 as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Tue Nov 5 04:38:33 2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL>
```

We've observed that there is considerable confusion, especially for manual operations, because we are used to the close tie between the SID and the database name. On several occasions we have gone into the database and ended up using the wrong container inadvertently.

The only possible workaround we found is to get into the habit, especially in SQL*Plus, of using the show container commands: SHOW CON_NAME; AND SHOW CON_ID

```
SQL> SHOW CON_NAME
  CON_NAME
 ------------------------------
 PDB1

 SQL> SHOW CON_ID CON_NAME
```

```
  CON_ID
 ------------------------------
 3
  CON_NAME
 ------------------------------
 PDB1

SQL>
```

JDBC and Connection Pools

Most of our customers have JDBC-based applications. These may be framework-based JEE applications which often use JNDI to provide appropriate JDBC lookup, or straight JDBC client/server type applications in which the JDBC connect string is hardcoded or passed by a parameter.

Possibly it because Oracle documentation has traditionally shown the connect string as

jdbc:oracle:thin:@host:port:sid

nearly every application uses a reference to the SID. As observed earlier, the said is synonymous with the root container, which is NOT the desired connection.

Most applications will therefore need to use the connect string similar to

jdbc:oracle:thin:@host:port/service

And virtually all JEE applications will need the JNDI lookups adjusted accordingly.

SUMMARY

While the multitenant architecture promises to be very important from a cost savings and resource management perspective, it will take some time getting used to, primarily because of the renewed emphasis on the service, or virtual instance.

Many applications and DBA scripts will need to be reviewed to ensure that the appropriate service access is being used rather than the SID access.

Something big: at least some column names in the data dictionary views

One surprise is that Oracle has decided to expand a number of the data dictionary column names, in particular a variety of identifiers, from 30 bytes or characters out to at least 128. Unfortunately, although the identifier is output at 128 characters, we still cannot create tables, views, columns, and other DB objects with identifiers of greater than 30 characters in length

Any scripts that depended on the 30 character format for output will of course now have an additional wrap or formatting problem. We ran across this when trying to get information about our various instances using summary reports that we would mail to ourselves. The first mailing was quite a shock where the identifier contents rolled off the edge of the paper.

Table 1 lists the column names for USER_* Advisor-related views that have changed from 30 bytes to 128 bytes.

Column Name	**Found in Views**
ADVISOR_NAME	USER_ADDM_FINDINGS.
DIRECTIVE_NAME	USER_ADDM_TASKS.
EXECUTION_NAME	USER_ADDM_TASK_DIRECTIVES.
EXECUTION_TYPE	
INSTANCE_NAME	
LAST_EXECUTION	
SOURCE	
TASK_NAME	
ADVISOR_NAME	USER_ADVISOR_ACTIONS.
EXECUTION_NAME	USER_ADVISOR_DIR_TASK_INST.
EXECUTION_TYPE	USER_ADVISOR_EXECUTIONS.
INSTANCE_NAME	USER_ADVISOR_EXEC_PARAMETERS.
PARAMETER_NAME	USER_ADVISOR_FDG_BREAKDOWN.
TASK_NAME	USER_ADVISOR_FINDINGS.
	USER_ADVISOR_JOURNAL.
	USER_ADVISOR_LOG.
	USER_ADVISOR_OBJECTS.
	USER_ADVISOR_PARAMETERS.
	USER_ADVISOR_RATIONALE.
	USER_ADVISOR_RECOMMENDATIONS.
COLUMN_NAME	USER_ADVISOR_SQLA_COLVOL.
PARAMETER_NAME	USER_ADVISOR_SQLA_REC_SUM.
TABLE_NAME	USER_ADVISOR_SQLA_TABLES.
TABLE_OWNER	USER_ADVISOR_SQLA_TABVOL.
TASK_NAME	USER_ADVISOR_SQLA_WK_MAP.
WORKLOAD_NAME	USER_ADVISOR_SQLW_COLVOL.
	USER_ADVISOR_SQLW_JOURNAL.
	USER_ADVISOR_SQLW_PARAMETERS.
	USER_ADVISOR_SQLW_TABLES.
	USER_ADVISOR_SQLW_TABVOL.
PARSING_SCHEMA_NAME	USER_ADVISOR_SQLA_WK_STMTS.
SOURCE	USER_ADVISOR_SQLA_WK_SUM.
SQLSET_NAME	USER_ADVISOR_SQLW_STMTS.
TASK_NAME	USER_ADVISOR_SQLW_SUM.
USERNAME	USER_ADVISOR_SQLW_TEMPLATES.
WORKLOAD_NAME	
EXECUTION_NAME	USER_ADVISOR_SQLPLANS.
OBJECT_NAME	USER_ADVISOR_SQLSTATS.
OBJECT_OWNER	
QBLOCK_NAME	
TASK_NAME	
ADVISOR_NAME	USER_ADVISOR_TASKS.
EXECUTION_TYPE	USER_ADVISOR_TEMPLATES.
LAST_EXECUTION	
SOURCE	
TASK_NAME	

Table 1

The standard object data dictionary view columns have also changed to display out to 128 characters, although we are not able to create tables or views with identifiers that long.

This includes all object identifiers for any of the following: OBJECT_NAME, VIEW_NAME, TABLE_NAME, INDEX_NAME, COLUMN_NAME.

Any variant on Username, such as OWNER, USERNAME, GRANTOR, GRANTEE also has expanded to 128 characters.

Summary

A SQL*Plus scripts that use the SQL*Plus 'COLUMN … FORMAT' capability may produce unexpected formatting issues. A number of environments use the scripts to do simple reporting, or mail environmental status to the DBA, possibly also to management, many of these types of scripts will have to be reviewed for simple format sizing changes.

Perhaps the effort in this particular conversion might become the incentive to switch over to Oracle's Enterprise Manager 12c, which has an impressive set of reports. It also includes incident management system, available with the base license of the database, no additional charge.

In conclusion

The Oracle 12 C architectural changes, especially multitenant environment, introduce some exciting new possibilities for cost savings and operations. However since the architectural changes are not guaranteed to be backward compatible, the DBA is advised to plan on a longer test cycle to get used to the smaller changes that are likely to crop up at the most surprising location.

Contact address:

Name	**Hans Forbrich**
Company	Forbrich Consulting Ltd.
Address	3-11 Bellerose Drive, PMB 117
Postal code, city	St. Albert, Alberta, T8N 3H9 Canada
Phone:	+1 780 499 6244
Email	hans@forbrichcomputing.ca
Internet:	hansforbrich@blogspot.ca