Oracle MultiTenant Pros and Cons

Overview

With the release of Oracle Database Server 12c, Oracle changed the architecture of the database significantly to support multiple databases within a single instance, or with a RAC cluster.

There are very few application program differences. However, the new architecture is a significant change, which has many potential impacts on the administration and ongoing operations of the database.

Oracle has made it optional to switch architecture by supporting the 'pre-12c' architecture.

Based on experience, there are a number of cautions and details that must be reviewed prior to using the new architecture in production. This paper and presentation demonstrates some of our findings and supports the upgrade to the new architecture as soon as feasible for the organization.

The paper is divided into the following area:

- licensing
- planning
- host preparation
- application considerations
- operations considerations

Now is the time to upgrade

Oracle publishes the Support Contract lifecycle dates for the database in the Lifetime Support Policy: Oracle Technology Products document found at http://www.oracle.com/us/support/lifetime-support/index.html . It is interesting to note that the Premier Support for all versions 11gR1 and older has ended, and Permier Support for 11gR2 ends in January 2015, in a very short time.

As a result, organizations wishing to retain Premier Support for their application databases must consider upgrading to Oracle 12c very soon.

However, within Oracle 12c, the two architectures can coexist on the same machine, even within the same ORACLE_HOME. Therefore the decision to upgrade to 12c should not be dependent on the use the architecture.

Licensing

There are two separate discussions around licensing: licensing the Multi-Tenant option; and licensing recommendations resulting from using multi-tenant capabilities.

Architecture vs Option

It is very important to note that there is a distinction between the architecture and the Enterprise Edition option.

Pro: the architecture is ubiquitous.

The multi-tenant architecture has been built into the database kernel, and is available (in 12.1.0.1) in all Editions: Personal Edition; Standard Edition One; Standard Edition; and Enterprise Edition. To date, 12.1.0.2 has only been released as Enterprise Edition.

The documentation refers to the single-database, single-instance architecture as "Pre-12c", implying that at some time in the future the multi-tenant architecture may become the only available choice.

When the Multi-Tenant architecture is used for a database instance, you automatically have

- a Container and the Root DB (CDB$ROOT)
- a Seed pluggable database (PDB$SEED)
- an Application pluggable database (PDB) may have been created.

Pro: both architectures can co-exist in a 12c ORACLE_HOME.

It is important to remember that the two architectures can coexist on the same machine, and even within the same ORACLE_HOME.

Licensing the option – Enterprise Edition only

Pro: the multi-tenant architecture is free with single tenants.

With the Enterprise Edition (only), multiple Application PDBs may be created. Without licensing the Multi-Tenant option, only one PDB may be used.

The Oracle Database 12c, 12.1.0.2 Licensing document at https://docs.oracle.com/database/121/DBLIC/options.htm#DBLIC2166 states "The multitenant architecture with one pluggable database (single tenant) is available in all editions without the Multitenant Option."

According to Oracle's official blog on the architecture, "The single tenant configuration does not require nor trigger the Multitenant licensed option. " https://blogs.oracle.com/Multitenant/entry/single_tenant_configuration .

Con: it is too easy to create a second tenant in Enterprise Edition, triggering license

The multi-tenant architecture becomes very useful when making duplicates, or when upgrading. These operations are very easy to perform, and are very fast, especially when the operation is simply creating a new PDB from and existing one.

However, if the operations are performed within a single instance, this may automatically trigger the need to license the option.

Con: The Multi-tenant license may be a significant cost

According to Oracle's Price List, most options are listed at 25%, 33% and 50% of core price: http://www.oracle.com/us/corporate/pricing/price-lists/index.html and must be licensed for the same CPU cores as the core database. The Multi-Tenant option lists at about 33%.

On the other hand, most customers will negotiate some discounts, or have some site- or enterprise-wide license agreement, so this may not be as significant in practice.

Pro: Multiple PDBs may reduce the total license cost

Oracle has a public benchmark that indicates the Multi-tenant configuration can reduce the overall memory and CPU requirements compared to using an equivalent number of non-CDB instances. The savings, achieved by eliminating duplicate background processes and eliminating redundant SGA requirements, are documented in http://www.oracle.com/technetwork/database/multitenant/learn-more/oraclemultitenantt5-8-final-2185108.pdf

The document indicates that a fully loaded multi-tenant environment can reduce the CPU required for background processes from 26% in a 254 instance environment to 3% in a 254 PDB environment. On the described 128 core machine, that represents a reduction of 30 cores. This implies that a smaller machine, and therefore fewer

licenses, could be used to handle the same load. Alternately, it implies that an existing machine can handle more load, thereby increasing it's useful life span.

Based on unpublished tests, for smaller PDB counts, the reduction is not as dramatic. On an 8 core machine with 64GB RAM, I was able to achieve 6 relatively light-load test non-CDB databases, or 10 PDBs, which represents an increase of slightly over 50%. Load and transaction rates, however, dramatically cut into that increase.

The savings in license is counter-balanced by the cost of the Multi-Tenant option.

Wish list: Initialization Parameter "MAX_PDBS"

Unfortunately, there is no explicit parameter that can be set to limit the number of (non-PDB$SEED) PDBs that exist within a database.

Licensing other Options – Enterprise Edition only

There are a large number of options now available with the Oracle Database Server 12c. These options are compatible with Oracle's Multi-tenant architecture and are described in the Licensing document, chapter 3. (See the Licensing document at https://docs.oracle.com/database/121/DBLIC/options.htm)

Pro: Consolidation

The multi-tenant architecture allows the easy addition and consolidation of Oracle databases across the enterprise.

Con: Future-proofing licenses

Best practice indicates that to ensure future compatibility with possible PDBs, **all** options should installed against the instance, even if not used. That allows the easy transition when adding or plugging a database that requires an option and ensures that the options are patched properly during quarterly CPU or PSU patching.

Unfortunately, unless documented in an agreement with Oracle, simply installing options without using them has occasionally triggered licensing during the License Management Review.

This implies either

- purchasing all options;
- documenting which options trigger license and under which conditions those triggers occur; or

- removing options at install time and administering the instances appropriately.

Con: Administratively separating which host has which options

In the last case, it should be remembered that Oracle generally requires all instances (at a specific Edition) on a machine to have the same license configuration Therefore the administration could require identifying which host has which options, and ensuring consolidation is performed based on the options.

Planning

Service Level Agreement

A Service Level Agreement (SLA), whether formal or informal, presents the requirements by the client in terms of availability, software version, patch cycle, and performance. The application vendor might impose additional restrictions.

Any consolidation effort must, first and foremost, be led by similarities in Service Level Agreements.

Traditionally, one ORACLE_HOME represented one SLA. When patching the Oracle software, typically all database and database instances in the same ORACLE_HOME would be patched at the same time.

With the Multi-tenant architecture, the SLA can, and should, be implemented at the instance or RAC level. Without additional effort, all PDBs need to be at the same software version and patch level.

Pro: Just because you can consolidate, doesn't mean you must

There are many situations in which consolidation is not recommended. If an application has high I/O, significant CPU or memory requirements, or the vendor insists on separation of instances, consolidation may be precluded.

Administering a multi-tenant database requires a significantly different approach and the commands are a superset of those which many DBAs are familiar.

According to the paper previously cited, the overhead of using the Multi-tenant architecture for a single active database is minimal. Therefore there is basically no cost to using the architecture. By using the architecture for single PDB environments as well as multiple PDB environments, the DBA skill set is consistent, reducing the chance of command set errors.

Pro: SLA consistency and resource planning

It becomes easy to create silos of environments that have similar SLAs, monitoring resources and being able to quickly add a new PDB within an existing silo based on resource requirements and resource availability.

Pro: One instance per machine is achievable; Resource Manager is usable

One significant strength of Oracle Database is that the Oracle database kernel is capable of managing and balancing resources within the scope of an instance. Oracle has had the DBRMS (Database Resource Management System) available for use since at least Oracle9i. Unfortunately, when a second instance is added to the machine, inter-instance conflicts on CPU and I/O resources cannot easily be arbitrated.

With Multi-tenant architecture, it is possible to use the DBRMS to manage resources across multiple databases within one instance. It is also possible to keep only one instance for up to 254 databases.

Pro: Migration and Upgrades are potentially faster, with less down-time

There are two separate considerations to upgrade or patch: upgrade from pre-12c to 12c Multi-tenant; upgrade or patch a Multi-tenant PDB.

In the first case, when the source database is at 11.2.0.4 or higher, it is possible to create the PDB plug-in configuration using the DBMS_PDB package, as described in the Oracle Database 12c Administrators Guide, Chapter 38. Using that information, it is possible to plug the database in to the CDB. Applying the post plug-in $ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql script then completes the process. This process is generally faster than performing a full non-CDB upgrade.

When upgrading a database, it is possible to unplug it, plug it in to a second CDB to keep it operational, upgrade the software as well as the core of first CDB, move the database to the upgraded CDB, and apply the upgrade to the PDB. Since most patched and upgrades involve the core data dictionary, which is in the CDB, the potential outage can be significantly reduced.

Con: It is necessary to understand the SLA

If multiple PDBs are to be hosted within one CDB, the SLA for each PDB must be understood in order to plan the patch, upgrade, and maintenance cycle. Taking down the instance will shut down all PDBs that are hosted in the CDB.

However, it may be possible to migrate the PDBs to a different CDB – a relatively fast operation – to minimize any potential outage.

Con: Effective migration is best on shared storage

Often the host is upgraded or patched, or does not have sufficient resources to handle two CDBs concurrently. In that case, it may be desirable to move the PDBs to a different host. This requires the files for the PDB to be visible o the second host.

To avoid moving the files across the network, some form of shared storage – clustered ASM, shared RAW devices, or NFS - may be used.

Therefore, during planning, proposed resource limitations should be analyzed to determine whether shared storage should be used and how it will be made available.

Con: One one character set for entire instance

The container, and all PDBs, must use the same character set. It is therefore recommended that all organizations consider moving to Unicode.

Host preparation

P*ro: it is easy to maximize resource usage by adding more PDBs*

When properly configured with sufficient memory, shared memory, suitable shared storage, process and file limits, and so on, adding PDBs is generally a fast and easy operation. Chapter 38 of the DB Administrator's Guide describes the methods to create and cline PDBs from a variety of sources.

Con: the 'ultimate' configuration should be set up early to avoid down time.

However, if the CDB has not been configured to handle the resource requirements of the combination of PDBs that are to be managed, it will be necessary to shut down, reconfigure the host and/or the instance, and restart the CDB and all the PDBs.

The ability to shut down the CDB may be problematic if there are multiple PDB, as the outage may involve multiple user communities.

In a Linux environment, SHMEM, Huge Pages, Semaphores, ulimits for number of processes and open files should all be evaluated assuming the entire machine will be used for Oracle database configuration only,.

Pro: one host, two CDBs, downtime minimized

Patching and upgrading in a non-CDB environment consists of two stages:

- patching of upgrading the software
- patching or upgrading the data dictionary

Both of these stages take significant time, and both generally require the ORACLE_HOME and all related database instances to be down.

One technique to minimize down time s to use a second ORACLE_HOME, ensure that is patched, take over the database instance from that patched ORACLE_HOME in 'UPGRADE' mode and upgrade the dictionary. This can still result in a significant outage.

With the CDB, that technique is extended by creating an 'empty' instance operated from the second, patched, ORACLE_HOME, which has the root dictionary upgraded. It is very easy and relatively fast to unplug a PDB from the original instance and plug it into the patched CDB instance. Running Datapatch will then complete the upgrade for the PDB. See **Datapatch: Database 12c Post Patch SQL Automation (Doc ID 1585822.1)**

Application considerations

Pro: PDBs are accessed ONLY through SERVICE connection

There are two ways to access a PDB:

- Using a common user and setting the container within the session;
  sqlplus c##hans/password
  alter session set container='PDBORCL';

- Conecting directly to the service provided by the container
  sqlplus test/test@//myhost:1521/pdborcl.forbrich.ca

Applications generally will use the second method.

Note that the SID form (sqlplus test/test@//myhost:1521:pdborcl) is not acceptable and will usually result in an error UNLESS the 'USE_SID_AS_SERVICE' SQLNET.ora parameter is set as described in https://docs.oracle.com/database/121/NETRF/listener.htm#NETRF2090

Pro: CDB automatically registers opened PDBs

Once the CDB container is made available, opening and closing individual PDBs will register and deregister the PDB with the listener automatically.

The 'ALTER SYSTEM REGISTER' will force a PDB to be registered when run from the PDB or all PDBs to be registered when run from the ROOT container.

Con: TNSNAMES.ora is not updated by DBCA

Although the DBCA may be used to create, alter, plug and unplug, and drop PDBs, it does not update the TNSNAMES.ora. This is consistent with the operation in previous versions of the DBCA, but can easily be a surprise to those who create the database directly from the Universal Installer, which creates an entry for the CDB but no PDBs.

The TNSNAMES.ora in the ORACLE_HOME is intended primarily for use by administrative clients and for operations in which the database instance itself is a client, such as DBLINKS.

Con: SQLPlus scripts may need to be updated

Many sqlplus scripts for applications may still be configured either to use the SID, TNSNAMES.ora alias or operating system authentication.

All scripts that use these techniques will likely need to be changed.

Con: SID access no longer works to get to the application level

There are still many scripts and applications that are configured to access the database instance using the SID rather than SERVICE. This is a carryover from the many blogs that demonstrated how to connect with Oracle8i and Oracle9i, especially through Java or .Net, before the use of SERVICE was popular.

This implies that many 3-tier applications need to be reviewed to verify they are correctly configured.

- *Correct:* user/password@//host:port/service
- *Wrong*: user/password@//host:port:SID

Pro: Application access will be upgraded to use SERVICE

Application administrators are encouraged to review and correct the access method.

Operations considerations

Many operations considerations are reviewed in Oracle's White Paper at http://www.oracle.com/technetwork/database/multitenant-wp-12c-1949736.pdf

There are impacts due to:

- The new set of administrative VIEW layer;
- The new V$ views and references in nearly every other V$ view;
- The different action of the views depending on context (ROOT vs PDB);
- The difference between Common and Local users and roles;
- The new OS roles;
- The use of a common UNDO, REDO and FLASHBACK area;
- The optional separation of TEMPORARY tablespace;
- The differences in BACKUP, RESTORE and RECOVERY;
- The need to understand the Resource Monitor;
- The impacts due to the common SGA related to performance monitoring and tuning;
- Other related areas.

Con: It is necessary to know the Context

Views can have a very different response, and commands can have a very different action, depending on whether they are invoked from the ROOT or from a PDB.

For SQLPlus, the use of a glogin.sql that sets the context is possibly recommended. One such script and example is at http://sateeshv-dbainfo.blogspot.de/2013/08/12c-display-cdbpdb-name-in-sql-prompt.html

Con: Experienced DBAs, there have a significant muscle-memory relearning curce

We all have favorite commands that roll off the finger tips quickly and easily, borne of many years of experience.

Based on experience with 12c Multi-tenant environment, it can take considerable time to get used to the small extra steps to get the correct, or context-relevant , information

Pro: The Enterprise Manager Database Control is replaced by EM Express

The DB Control, which has been available since Oracle 10g, is replaced by the Enterprise Manager Express. EM Express is based on a much lighter Application Express framework, which has a significantly lower CPU requirement.

Con: EM Express is severely limited in capability

EM Express is not, nor is it intended to be, a complete replacement for the DB Control. It is a light weight administrative utility that can handle many of the basic admin functions as described in the documentation at https://docs.oracle.com/database/121/ADMQS/em_manage.htm#ADMQS003

Pro: Enterprise Manager Cloud Control understands Multi-tenant

The complete replacement for the DB Control is the Cloud Control. The ability to install the Cloud control, at no extra charge or basic functionality, is included with the properly licensed Oracle Database, as described in the Cloud Control License document at https://docs.oracle.com/cd/E24628_01/doc.121/e24474/toc.htm in Chapter 1.

Pro: Administrative views are extended to provide CDB and PDB information

There are many changes in the DBA_, ALL_ and USER_ views as well as new CBD_ described in the https://docs.oracle.com/database/121/REFRN/toc.htm Reference guide.

The same guide describes the changes in the V$ views as well. It may take some time to adjust scripts to these new views.

Pro: CDB_ views support Administrators separation of duties

The contents of the CDB_ views change depending on whether the administrator is in t he root container or in a PDB, and whether the userid is considered a common user or only a local user. This will initially cause some confusion until the administrator is used to the differences.

Pro: Oracle 12c encourages more separation of duties

In previous versions on Unix and Linux, Oracle has encouraged us to NOT log on as OS user oracle by using the setgid capabilities.

In a nutshell, the OS userid that is part of the 'dba' group has access to all relevant Oracle administrative commands except for those related to software upgrades.

With 12c, Oracle has added several new groups and roles – Backup, Data Guard, and Key Management. While not strictly for Multi-tenant, the use of these roles together

with local PDB administration userids, significantly enhances the ability to delegate administrative sub-tasks for those organizations that require such separation.

It is also possible for the DBA to spin off some administration of PDBs to the end user or developer community, while retaining the ability to manage key stores, backups and standby operations.

Pro: Common UNDO, REDO and FLASHBACK RECOVERY area

The Multi-tenant environment uses common areas and mechanisms for each of UNDO, REDO and FLASHBACK RECOVERY.

This can simplify administration, as it only needs to be sized for the accumulated sum of the UNDO, REDO and RECOVERY area needs.

Con: Common UNDO, REDO and FLASHBACK RECOVERY area

However, it can allow one PDB to overwhelm others in terms of use of these areas.

The size of these areas can now appear to be significant, even though it is simply the sum of the sizes that would be needed for individual instances.

Con: Recover using Redo Logs gets 'interesting'

This has some interesting implications specifically related to recovery operations and 'moving' PDBs between containers.

In particular, since all REDO is placed in the same common REDO logs, backup and recovery must, by definition, include all archive logs even if there is not relevant data in those log files. This could lead to slower recovery, and in particular point-in-time recovery operations.

In addition, if a PDB has been moved to another container, it appears that point in time recovery to before that move will not be possible.

Con: Flashback Database is whole container only

Similarly, the Flashback Database capability is for the entire instance. It appears that flashback on a PDB level is not yet possible. This implies that organizations that have become used to FLASHBACK DATABASE may need to be prepared to revert to standard Point In Time recovery.

Pro: TEMPORARY tablespace may be common or local

The Temporary tablespace and tablespace group may be common if desired. However, if applications make heavy use of temporary tablespace, these can be set up locally to avoid cross-application contention issues.

Pro: RMAN understands PDB

It is possible to take backups and perform many RMAN operations against individual PDBs. However, the archive logs will include the contents of all PDBs.

The RMAN manual describes several implications and aspects of PDB operations at https://docs.oracle.com/database/121/BRADV/rcmcnctg.htm

Con: RMAN backups for a PDB contain transactions for all PDBs

This has a potential security implication if the PDB backup is to be sent off site as a means to ship data outside of the organization.

Pro: RMAN is now effective

Oracle has made Database Resource Management (DBRM) available since Oracle8i. However, since many hosts had multiple instances, and since DBRM is not able to communicate between instances, many organizations seem to have concluded that they will not use the Resource manager

With Oracle Database 12c Multi-tenant, the need for multiple instances on a host may be dramatically reduced or eliminated and DBRM can be an effective way to provide resource management and resource balancing.

Conclusion

Over all, Oracle's Multi-tenant architecture and associated option is well worth considering. It is my belief that the non-CDB architecture will eventually be removed.

However, there is a significant learning curve, and there are a number of considerations that must be evaluated for the organization before adoption. I believe that this will require at least 4 months, and perhaps 6 months, of effort to become used to the multi-tenant architecture.

Since the architecture is available with the installation of the software, but not invoked until the database is created, and since non-CDB and CDB architecture database instances can be hosted – even from the same ORACLE_HOME – this is not a mutually exclusive decision.

In a number of areas, in spite of the apparent increase in cost due to licensing the option, the benefits are significant and may, in an ROI analysis, demonstrate that the architecture and option should be adopted.

Recommendation: Proceed – cautiously.