Oracle® Database

Upgrading PDBs in Parallel on the Same System

18c
E97286-01
July 2018

Oracle Database Upgrading PDBs in Parallel on the Same System, 18c

E97286-01

Primary Authors: Sunil Surabhi, Nirmal Kumar

Contributing Authors: Douglas Williams, Prakash Jashnani, Mark Bauer

Contributors: Roy Swonger, Byron Motta, Hector Vieyra Farfan, Carol Tagliaferri, Mike Dietrich

Contents

Preface

This guide provides a compilation of topics from the Oracle Database user assistance documentation that are collected to help you complete a specific use case scenario.

Use Case Scenario for this Document

Use this scenario document to assist you to upgrade the entire multitenant architecture to a later release, including `CDB$ROOT`, `PDB$SEED`, and all of the pluggable databases (PDBs) plugged into the CDB, using parallel processing to limit your downtime.

Prerequisites for this Scenario

- You have installed the new release Oracle Database software on your server.
- You have backed up your multitenant architecture.

Oracle recommends that you update your source and target databases to the most recent release update (Update), or release update revision (Revision) before starting CDB upgrades.

Outline for this Scenario

- **Upgrading Multitenant Architecture In Parallel**. Learn about and then apply the procedure for upgrading your CDB, and upgrading all or part of your PDBs on a CDB in parallel.
  1. Review about upgrading pluggable databases in parallel.
  2. Upgrade the multitenant CDB and PDB in parallel.

These steps correspond to the chapters and sections in this document.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

Upgrading Multitenant Architecture In Parallel

Use this technique to upgrade multitenant architecture Oracle Database releases (Oracle Database 12*c* Release 1 (12.1.0.1) and later by upgrading container databases (CDBs), and then upgrading multiple pluggable databases (PDBs) in parallel.

- About Upgrading Pluggable Databases (PDBs) In Parallel
  Using the In-Parallel technique, you can upgrade the CDB, and then immediately upgrade PDBs using parallel SQL processors.
- Upgrading Multitenant Container Databases In Parallel
  Use this technique to upgrade CDB$ROOT, PDB$SEED, and all PDBS in the CDB in one upgrade operation.

About Upgrading Pluggable Databases (PDBs) In Parallel

Using the In-Parallel technique, you can upgrade the CDB, and then immediately upgrade PDBs using parallel SQL processors.

Container databases (CDBs) can contain zero, one, or more pluggable databases (PDBs). By default, the Parallel Upgrade Utility (`catctl.pl`) updates the CDB and all of its PDBs in the same upgrade window. The Parallel Upgrade Utility uses the number of computer processing units (CPUs) to determine the maximum number of PDBs that are upgraded simultaneously. The number of PDBs that are upgraded in parallel is determined by dividing the parallel SQL process count (`-n` option) by the parallel PDB SQL process count (`-N` option).

> **Note:**
>
> You must plan your upgrade window to accommodate a common downtime for all of the database services that the PDBs on the CDB are providing.

Pluggable Database Upgrade Syntax

```
dbupgrade [-M] -n [-N]
```

- `-M` Specifies if CDB$ROOT is kept in upgrade mode, or if it becomes available when it completes upgrade:
  - If you run the Parallel Upgrade Utility with the `-M` parameter, then the upgrade places CDB$ROOT and all of its PDBs in upgrade mode, which can reduce total upgrade time. However, you cannot bring up any of the PDBs until the CDB and all of its PDBs are upgraded.
  - If you do not run the Parallel Upgrade Utility with the `-M` parameter, then CDB$ROOT is upgraded and restarted in normal mode, and the normal

background processes are started. After a successful upgrade, only CDB$ROOT is opened in read/write mode. All the PDBs remain in MOUNT mode. As each PDB is upgraded, you can bring each PDB online while other PDBs are still being upgraded.

- `-n` Specifies the number of in-parallel PDB upgrade processors.

  If you do not specify a value for `-n`, then the default for `-n` is the `CPU_COUNT` value.

  If you do specify a value for `-n`, then that value is used to determine the number of parallel SQL processes. The maximum value is unlimited. The minimum value is 4.

- `-N` Specifies the number of SQL processors to use when upgrading PDBs. The maximum value is 8. The minimum value is 1. If you do not specify a value for `-N`, then the default value is 2.
- The maximum PDB upgrades running concurrently is the value of `-n` divided by the value of `-N`.

The following is a high-level list of actions during the In Parallel PDB upgrade technique:

1. Make sure that your backup strategy is complete.
2. Run the Pre-Upgrade tool. Fix any issue that is reported.. The Pre-Upgrade Tool (`preupgrade.jar`) is shipped with the new Oracle database release.
3. Run the Parallel Upgrade Utility. In sequence, the following upgrades are carried out:
   a. Cycle 1: CDB$ROOT is upgraded to the new Oracle release
   b. Cycle 2 to Cycle x: PDB$SEED and PDBs are upgraded in parallel, with the number of cycles of upgrades as determined by the parameter settings you specify with `-n`.
4. Complete post-upgrade steps.

Example 1-1 Example of Multitenant Architecture Upgrade Using Defaults (No Parameters Set)

In this scenario, your `CPU_COUNT` value is equal to 24. If you do not specify a value for in-parallel PDB processors using the `-n` option, then the default value for in-parallel PDB processors (`-n`) is equal to 24. If you do not specify a value for `-N`, then the default value for the number of SQL processors (`-N`) is 2.

Result:

12 PDBs are upgraded in parallel (`CPU_COUNT` divided by 2, or 24 divided by 2.) There are 2 parallel SQL processes allocated for each PDB.

Example 1-2 Example of Multitenant Architecture Upgrade Using 64 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes

In this scenario you set the value of in-parallel PDB upgrade processors to 64 by specifying the option `-n 64`. You specify the value of parallel SQL processors to 4 by specifying the option `-N 4`.

Result:

16 PDBs are upgraded in parallel (64 divided by 4). There are 4 parallel SQL processes for each PDB.

Example 1-3 Example of Multitenant Architecture Upgrade Using 20 In Parallel PDB Upgrade Processors and 2 Parallel SQL Processes

In this scenario you se the value of in-parallel PDB upgrade processors to 20 by specifying the option `-n 20`. You specify the value of parallel SQL processors to 2 by specifying the option `-N 2`.

Result:

10 PDBs are upgraded in parallel (20 divided by 2). There are 2 parallel SQL processes for each PDB.

Example 1-4 Example of Multitenant Architecture Upgrade Using 10 In Parallel PDB Upgrade Processors and 4 Parallel SQL Processes

In this scenario you se the value of in-parallel PDB upgrade processors to 10 by specifying the option `-n 10`. You specify the value of parallel SQL processors to 2 by specifying the option `-N 4`.

Result:

2 PDBs are upgraded in parallel (10 divided by 4). There are 4 parallel SQL processes for each PDB.

Upgrading Multitenant Container Databases In Parallel

Use this technique to upgrade CDB$ROOT, PDB$SEED, and all PDBS in the CDB in one upgrade operation.

Oracle recommends that you use this approach if you can schedule downtime, because it provides a direct procedure for upgrades and simplicity of maintenance. Using this procedure upgrades in parallel all the PDBs in the multitenant architecture container database, depending on your server's available processors (CPUs).

> **Note:**
>
> When you upgrade the entire container using the In Parallel upgrade method, all the PDBs must be down. Perform the upgrade in a scheduled upgrade window so that you can bring all the PDBs down.

> **Caution:**
>
> - Always create a backup of existing databases before starting any configuration change.
> - You cannot downgrade a database after you have set the compatible initialization parameter to 12.1.0.2. A downgrade is possible for a pluggable database (PDB) only if the compatibility is set to 12.1.0.1 or later. There can still be additional downgrading restrictions .
> - Oracle strongly recommends that you upgrade your source and target databases to the most recent bundle patch or patch set update (BP or PSU) before starting an upgrade, and to the most recent release update before starting a downgrade.

1. Ensure that you have a proper backup strategy in place.
2. Open all PDBs.

   For example:

   ```
   SQL> alter pluggable database all open;
   ```

3. Run the Pre-Upgrade Information Tool (`preupgrade.jar`), using the following syntax:

   `/java -jar` *`$New_release_Oracle_home`*`/rdbms/admin/preupgrade.jar [TERMINAL|FILE| DIR outputdir] [TEXT|XML] [-c` *`InclusionListOfPDBs`*`] [-C` *`ExclusionListOfPDBs`*`]`
   Use space-delimitation for lists. On Linux and UNIX, define the list by placing the list inside single quotes: `'`. On Windows systems, define the list by placing the list inside double quotes `"`.

   For example, run the following command to run the Pre-Upgrade Information tool on PDBs PDB1 through PDB25, where you have set up an environment variable `$ORACLE_HOME_12.1` for your Oracle Database Oracle home in `/u01/app/oracle/product/12.1.0/dbhome_1/`, and you have set up an environment variable `$ORACLE_HOME_18.1` for your new Oracle Database Oracle home in `/u01/app/oracle/product/18.1.0/dbhome_1/`:
   Linux and UNIX:

   ```
   java -jar $ORACLE_HOME_18.1/rdbms/admin/preupgrade.jar \
   -c 'pdb1 pdb2 pdb3 pdb4 pdb5 pdb6 pdb7 pdb8 pdb9 pdb10 pdb11 pdb12 pdb13\
   pdb14 pdb15 pdb16 pdb17 pdb18 pdb19 pdb20 pdb21 pdb22 pdb23 pdb24 pdb25'
   ```

   Windows:

   ```
   java -jar %ORACLE_HOME_18.1%/rdbms/admin/preupgrade.jar \
   -c "pdb1 pdb2 pdb3 pdb4 pdb5 pdb6 pdb7 pdb8 pdb9 pdb10 pdb11 pdb12 pdb13\
   pdb14 pdb15 pdb16 pdb17 pdb18 pdb19 pdb20 pdb21 pdb22 pdb23 pdb24 pdb25"
   ```

   > **Note:**
   >
   > You must use Java 1.5 or later to run the Pre-Upgrade Information tool. By default, the Java releases in Oracle Database releases that you can upgrade directly support the tool.

4. Review any generated fixup scripts and log files.

   By default, if ORACLE_BASE is defined, then the fixup files are placed in one of the following paths:

   - Linux and UNIX:

     `$ORACLE_BASE/cfgtoollogs/db_unique_name/preupgrade`

   - Windows

     `%ORACLE_BASE%\cfgtoollogs\db_unique_name\preupgrade`

   If ORACLE_BASE is not defined, then fixup files are placed in one of the following paths:

   - Linux and UNIX:

     `$ORACLE_HOME/cfgtoollogs/db_unique_name/preupgrade`

   - Windows:

     `%ORACLE_HOME\cfgtoollogs\db_unique_name\preupgrade`

   On multitenant architecture Oracle Databases, the Pre-Upgrade Information Tool also creates a consolidated `preupgrade_fixups.sql` script. You can run the consolidated fixup script by using `catcon.pl`. The consolidated fixup script runs on every container that was open at the time that you ran the `preupgrade.jar` command.

5. Run the `preupgrade_fixups` script, or individual PDB scripts. The `preupgrade_fixups` SQL scripts resolve some of the issues reported by the `preupgrade` script.

   On multitenant environment Oracle Database deployments, you can run `preupgrade_fixups`*`pdb-name`*`.sql` scripts on the source database, where *`pdb-name`* is the PDB name. If you generate fixup scripts for PDBs, then the PDB name is added to the fixup filename.

   In addition to the individual PDB fixup scripts, you can use `catcon.pl` to run the consolidated `preupgrade_fixups.sql` script. The consolidated script runs on every container that was open at the time that you ran `preupgrade.jar`.

   > **Note:**
   >
   > Because $ is a reserved symbol on operating systems, the fixup script for PDB$SEED is `preupgrade_fixups_pdb_seed.sql`.

   Complete any other preupgrade tasks that the Pre-Upgrade Information Tool identifies.

6. (Conditional) for Oracle RAC databases, set the cluster database initialization parameter to false:

   For example;

   ```
   ALTER SYSTEM SET cluster_database=FALSE SCOPE=spfile;
   ```

7. Shut down the database in the old Oracle home.

   For example, where *`db_unique_name`* is your database name:

   ```
   $ srvctl stop database -d db_unique_name
   ```

8. Copy the `PFILE` or `SPFILE` from the old Oracle home to the new Oracle home
9. Connect with SQL*Plus:

   ```
   sqlplus / as sysdba
   ```

10. Bring the CDB$ROOT instance into upgrade mode:

    ```
    STARTUP UPGRADE
    ```

11. Bring all PDBs into upgrade mode:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN UPGRADE;
    ```

12. Check the status of PDBs to confirm that they are ready to upgrade:

    ```
    SHOW PDBS
    ```

    For all PDBs, ensure that the status is set to MIGRATE.

13. Exit from SQL*Plus, and change directory to the new Oracle home `$ORACLE_HOME/rdbms/admin`:

    ```
    SQL> EXIT
    $ ORACLE_HOME/bin
    ```

14. Start the upgrade using the Parallel Upgrade Utility (`catctl.pl`, using the shell command `dbupgrade`), where `-d` specifies the location of the directory:

    ```
    dbupgrade -d $ORACLE_HOME/rdbms/admin
    ```

    If you do not specify any parameters, then the Parallel Upgrade Utility runs the upgrade in parallel on the number of PDBs equivalent to the number of CPUs divided by 2. On a server with 64 CPUs, 64 divided by 2 equals 32 PDBs upgraded in parallel, carried out with two SQL processors for each PDB. CDB$ROOT remains in NORMAL mode for the duration of the upgrade.

15. Review the `upg_summary.log` to conf8irm that the upgrade was successful, and if necessary, review other logs.
16. Open all PDBs, so that you can recompile the databases:

    ```
    ALTER PLUGGABLE DATABASE ALL OPEN;
    ```

17. Exit from SQL*Plus, and change directory to the new Oracle home path `$ORACLE_HOME/rdbms/admin`:

    ```
    SQL> EXIT
    cd $ORACLE_HOME/rdbms/admin
    ```

18. Run the `catcon.pl` script and the `postupgrade_fixups.sql` script that is supplied with the new release Oracle Database.

    The following example shows the command strings for running `catcon.pl`, using the `-n` parameter to specify one parallel processor for each PDB, using the `-d` parameter to specify the path where the preupgrade script that you want to run is located, using the `-l` parameter to specify the location where you want the scripts to place log files, and using the `-b` flag to specify the log file prefixes for the `postupgrade_fixups.sql` script:

    ```
    $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -n 1 -d \
    $ORACLE_HOME/cfgtoollogs/cdbupgr/preupgrade -l /home/oracle/upgrdDBA -b \
    postupgrade_fixups postupgrade_fixups.sql
    ```

19. Run `postupgrade_fixups.sql`.

Non- CDB:

```
SQL> @rdbms/admin/postupgrade_fixups.sql
```

CDB:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b postupgradefixups -d '''.''' postupgradefixups.sql
```

20. Run `utlu122s.sql` to verify that there are no upgrade issues.

    Non-CDB:

    ```
    SQL> @rdbms/admin/utlu122s.sql
    ```

    CDB:

    ```
    $ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlu122s -d '''.''' utlu122s.sql
    ```

    When you use `catcon.pl` to run `utlu122s.sql`, the log file `utlu122s0.log` is generated. The log file provides the upgrade results. You can also review the upgrade report, `upg_summary.log`.

    To see information about the state of the database, run `utlu122s.sql` as many times as you want, at any time after the upgrade is completed. If the `utlu122s.sql` script returns errors, or shows components that do not have the status `VALID`, or if the version listed for the component is not the most recent release, then perform troubleshooting.

21. (Conditional) For Oracle RAC environments only, enter the following commands to set the initialization parameter value for `CLUSTER_DATABASE` to `TRUE`, and to start the Oracle RAC database, where *`dbname`* is the name of the Oracle RAC database:

    ```
    ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
    srvctl start database -db db_unique_name
    ```

    Your database is now upgraded.

> **Caution:**
>
> If you retain the old Oracle software, then never start the upgraded database with the old software. Only start Oracle Database using the start command in the new Oracle Database home.
>
> Before you remove the old Oracle environment, relocate any data files in that environment to the new Oracle Database environment.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about relocating data files

Index

M