Hands-On-Lab

February 28, 2016

Oracle Database 12c Release 2 (12.2.0.1) - Upgrade, Migration & Consolidation

Roy Swonger
Vice President and Product Manager
ST – Database Utilities
ORACLE Corporation

Mike Dietrich
Master Product Manager
ST – Upgrade Development Group
ORACLE Corporation

The **4 parts** of the Lab:

1. **Upgrade** an Oracle 11.2.0.4 database (SID: **UPGR**) to Oracle 12.2.0.1
2. **Plug in** the upgrade **UPGR** database into an existing Oracle 12.2.0.1 container database (SID: **CDB2**)
3. **Migrate** an Oracle 11.2.0.4 database (SID: **FTEX**) to Oracle 12.2.0.1 using Full Transportable Export/Import into a new pluggable database **PDB2**

[optional]:

4. **Work** with Multitenant databases and implement new Oracle Database 12c features

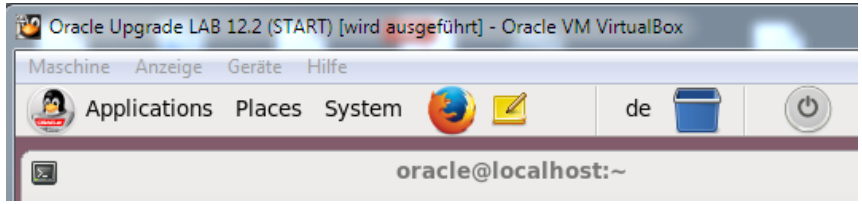
Before you can start you may have to setup a few things and make yourself familiar with the environment

Setup Tasks

Keyboard Layout

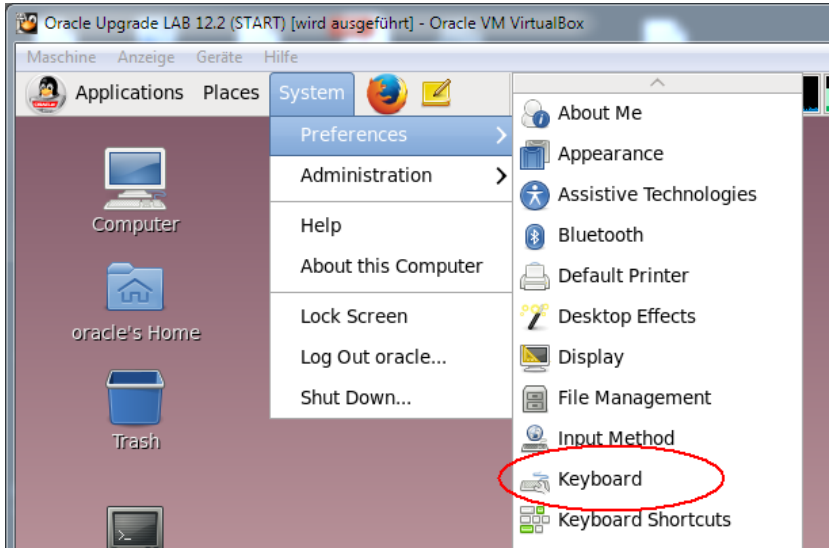
The default keyboard layout may be German or US English. **If you would like a different layout,** then you can add another keyboard as follows:

1. Login to the Linux desktop (user/password are both “oracle”)
2. If you want to change the keyboard's layout (default is: US) to German please just CLICK ONCE on the tiny "us" symbol next to the trash bin:



Note: it may take two attempts to change the keyboard layout, as you may notice that it reverts to the old value when you right-click on the desktop the first time.

3. If you'd like to do more general changes please enter SYSTEM ➔ KEYBOARD and choose your desired keyboard layout:



!!! IMPORTANT !!! THINGS TO KNOW AND UNDERSTAND !!!

All **passwords** are set to: **oracle**

Switch environments for instance: `. cdb2` (type in on shell prompt: <dot> <blank> cdb2)

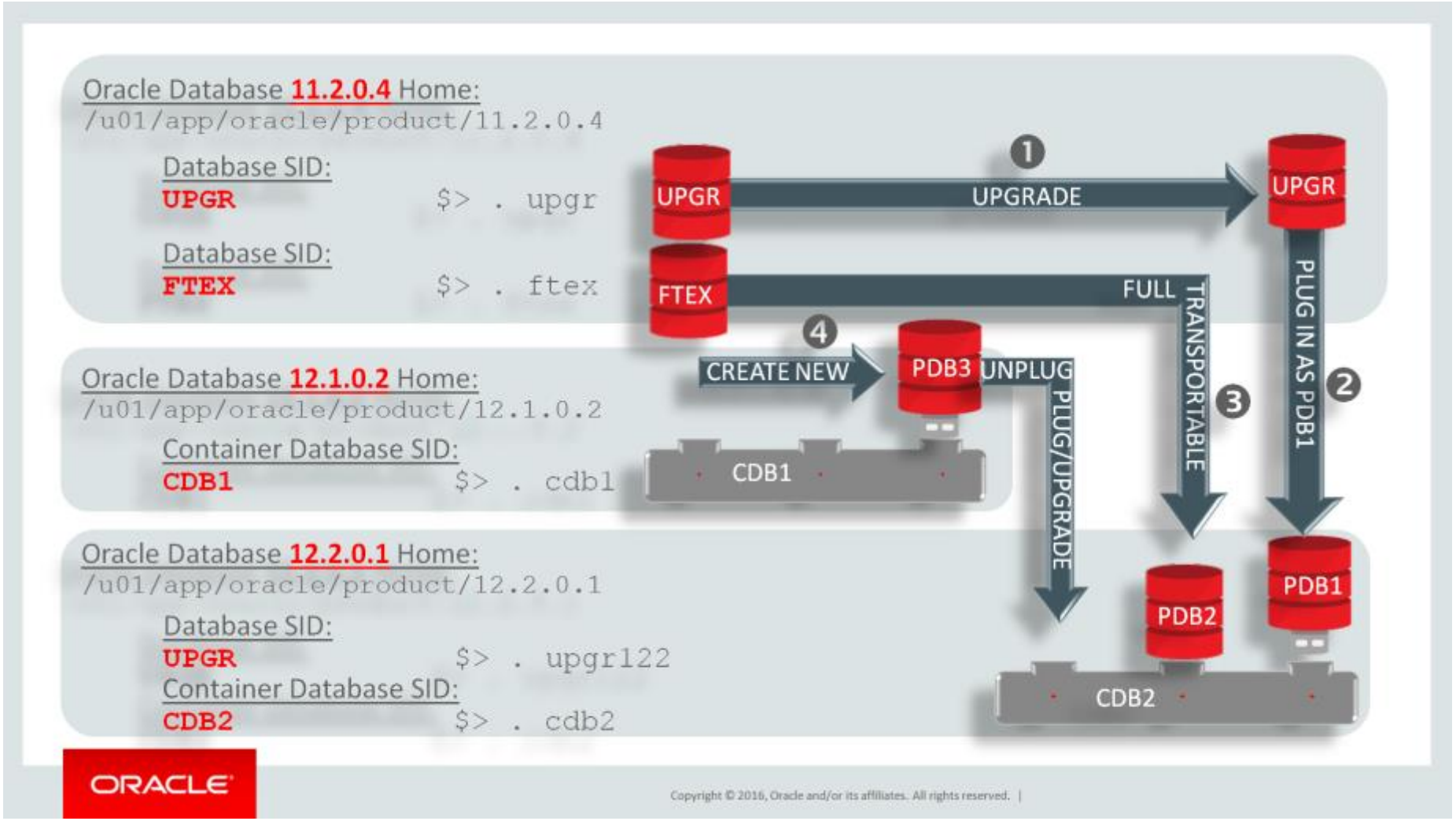
There is an environment variable $OH12 defined for convenience. This points to the 12.2.0.1 Oracle Home, and is used several times in part 1 of the lab.

Dark gray background, white characters mean: Execute on the command prompt (OS shell)

Light gray background, black characters mean: Execute in SQL*Plus

IMPORTANT !!! System Overview - The numbers on the picture describe part 1-4 of the Hands-On-Lab !!!

This is included on your desktop as the file background.jpg. It may be helpful to open this on a separate desktop for reference.



HOL – Part 1– Upgrade the Oracle 11.2.0.4 database UPGR to Oracle Database 12.2.0.1

Database files location:	/u02/oradata/UPGR
Initialization parameter and password file location:	/u01/app/oracle/product/11.2.0.4/dbs
Listener configuration:	/u01/app/oracle/product/12.2.0.1/network/admin

Tasks HOL Part 1

Your task in HOL Part 1 will be a simple and straightforward database upgrade to Oracle Database 12c. Everything is installed already. Your Oracle Database 11*g* Release 2 (11.2.0.4) database (SID: **UPGR**) is startup already and ready to go. Now follow all steps and upgrade it.

You will use the new pre-upgrade check tool preupgrade.jar which will examine your UPGR database. This script gets shipped with the new Oracle 12c home in /u01/app/oracle/product/12.2.0.1/rdbms/admin

You will then prepare your UPGR database for the upgrade to Oracle Database 12c and upgrade it. The database will stay in place and doesn't get moved to another location.

Remarks:

For this hands-on lab we have provided easy commands to switch environments! You can switch between environments on the bash shell prompt typing ((don't type "$>"!!!) in every Terminal/xterm:

SID: UPGR – Oracle 11.2.0.4 home	SID: UPGR – Oracle 12.2.0.1 home
`$> . upgr` `<dot> <space> upgr` for the Oracle 11.2.0.4 environment with your database to be upgraded (SID: UPGR)	`$> . upgr122` `<dot> <space> upgr12` for the Oracle 12c environment with your database to be upgraded (SID: UPGR)

*** START HERE *** Command Line Upgrade from Oracle 11.2.0.4 to Oracle 12.1.0.2 ***

In this section you'll execute the **new preupgrd.sql** check script, verify the output, execute some commands and a fixup script and prepare a new spfile for the upgrade. Then you'll copy the spfile and the password file to the new Oracle Database 12c home.

SID: UPGR Oracle 11.2.0.4 home	SID: UPGR Oracle 12.2.0.1 home

Execute pre-upgrade preparation steps
 	
1. **Open one xterm** ("Terminal" icon) and start SQL*Plus.
 	
`. upgr`
`sqlplus / as sysdba`
 	
(throughout the lab you can also use the simple shortcut "**s**" instead of typing "sqlplus / as sysdba")
 	
2. **Open a second xterm** side by side and **run the new preupgrade.jar** in your 11.2.0.4 environment:
 	
`. upgr`
`java -jar $OH12/rdbms/admin/preupgrade.jar TERMINAL TEXT`
 	
3. **Verify the output (scroll from top to bottom)** and make necessary changes in SQL*Plus (in other xterm)
 	
Execute the preupgrade_fixups.sql:	

```
@/u01/app/oracle/cfgtoollogs/UPGR/preupgrade/preupgrade_fixups.sql
```

Adjust the spfile parameters but take a copy of it before for safety reasons:

```
create pfile from spfile;

alter system set processes=300 scope=spfile;
alter system set sga_target= 998244352 scope=spfile;
```

Because part 2 of the lab uses Oracle Multitenant you must also raise COMPATIBLE:

```
alter system set compatible='12.2.0' scope=spfile;
```

The preupgrade.jar output displays a message about moving audit data from system.aud$ to sys.aud$ because Oracle Label Security is installed.
Move the AUD$ table now using the `olspreupgrade.sql` script from the Oracle 12c home from SYSTEM to SYS:

```
@$OH12/rdbms/admin/olspreupgrade.sql
```

Gather dictionary stats prior to the upgrade:

```
EXECUTE dbms_stats.gather_schema_stats('SYS');
```

4. You can **rerun** `preupgrade.jar` if you want to check whether all tasks have been completed. Just be aware that the spfile parameters still get displayed as you fixed them in the spfile which is not visible to the database instance at this moment.

5. Shutdown the UPGR database:

```
shutdown immediate
exit
```

6. Copy your new spfile file into the Oracle 12c home's `dbs` (`$OH12/dbs`) directory:

```
cp $ORACLE_HOME/dbs/spfileUPGR.ora $OH12/dbs/
```

SID: UPGR Oracle 11.2.0.4 home	SID: UPGR Oracle 12.2.0.1 home

Execute all parallel upgrade steps

Now you'll upgrade your UPGR database to Oracle Database 12c using the new parallel upgrade scripts. Furthermore you'll recompile and check for invalid objects before/after the upgrade.

1. Switch to the Oracle 12.2 environment and create a new password file first

```
. upgr122
cd $ORACLE_HOME/dbs
orapwd file=orapwUPGR force=y format=12
```

Put in "`oracle`" as password to ease your life in the lab.
Then start SQL*Plus:

```
sqlplus / as sysdba
```

2. **Bring the UPGR database into UPGRADE mode**

```
startup upgrade
exit
```

3. **Upgrade the UPGR database** with the parallel upgrade script
   Start the new parallel upgrade – it will be driven by a PERL script catctl.pl outside of SQL*Plus and execute in 4 parallel threads – in maximum you could run with 8 parallel threads by specifying the parameter option **-n 8**

```
cd $ORACLE_HOME/rdbms/admin
$ORACLE_HOME/perl/bin/perl catctl.pl -l /home/oracle catupgrd.sql
```

   You will now see >100 phases listed – some can act in parallel, others get executed serially. This will now take up to 15-30 minutes depending on your system. If you wonder about the RESTART phases: those happen if timing dependencies make it necessary to cleanup something internally. Logfiles will be written to /home/oracle as you specified this directory with the `-l` option

   Once the upgrade is finished it will shutdown the database and in the next phase you'll restart it in normal mode.

IF YOUR MACHINE IS WELL EQUIPPED WITH RAM/CPU YOU MAY DO TASKS IN PARALLEL AND START WITH PART 3 (FULL TRANSPORTABLE EXPORT).
GOTO PAGE 15 – HOL PART 3

SID: UPGR Oracle 11.2.0.4 home	SID: UPGR Oracle 12.2.0.1 home

	Finalize the upgrade with all required post upgrade steps
	
	During this part you'll finalize the upgrade with recompilation, postupgrade_fixups.sql and the time zone adjustment to TZ V26. Startup the database – post upgrade it is shutdown:
	
	1. In your xterm, **STARTUP t**he UPGR database and **recompile** everything::
	
	`. upgr122`
	`sqlplus / as sysdba`
	`startup`
	`@?/rdbms/admin/utlrp.sql`
	
	2. **Gather dictionary stats post upgrade:**
	
	`exec dbms_stats.gather_dictionary_stats;`
	`exec dbms_stats.gather_fixed_objects_stats;`
	
	3. **Adjust Time Zone** settings – you may look into the scripts taken from MOS Note: 1585343.1:
	
	`@/home/oracle/DST/upg_tzv_check.sql`
	`@/home/oracle/DST/upg_tzv_apply.sql`
	
	4. **Execute the `postupgrade_fixups.sql`:**
	
	`@/u01/app/oracle/cfgtoollogs/UPGR/preupgrade/postupgrade_fixups.sql`
	`exit`
	
	5. **Update your `/etc/oratab` file manually:**
	
	`vi /etc/oratab`

	Update the line:
	
	`UPGR:/u01/app/oracle/product/11.2.0.4:Y`
	
	to:
	
	`UPGR:/u01/app/oracle/product/12.2.0.1:Y`

*** COMPLETED *** Tasks HOL Part 1 ***

HOL – Part 2 – Plug in UPGR into CDB2, an Oracle Database 12.2.0.1 container database

Database files location:	/u02/oradata/UPGR
Initialization parameter and password file location:	/u01/app/oracle/product/12.2.0.1/dbs
Listener configuration:	/u01/app/oracle/product/12.2.0.1/network/admin

Tasks HOL Part 2

Oracle Multitenant Option is a way to **consolidate** several independent databases into one large **Container Database**. The CDB$ROOT is the administrative layer and contains absolutely no user or application data. The PDBs that are plugged into the CDB contain the user and application data. With Oracle Database 12c Release 2 you can have up to 4096 PDBs within one CDB.

Applications and clients will connect to the PDB just as they would connect to a non-CDB. The entire CDB/PDB shares one SGA, one set of background processes, one redo log stream.

In HOL Part 2 you will plug in the already upgraded UPGR database as a new pluggable database PDB1 into the already existing Container Database CDB2. The data files of UPGR will stay in place.

SID: UPGR – Oracle 12.2.0.1 home	SID: CDB2 – Oracle 12.2.0.1 home
`$> . upgr122` <dot> <space> upgr122 for the Oracle 12.1.0.2 environment with your database to be plugged in later (SID: UPGR)	`$> . cdb2` <dot> <space> cdb2 for the Oracle 12c environment connecting to the Container Database (SID: CDB2)

*** START HERE *** Plug in UPGR into CDB2 ***

In this section an XML description file for UPGR will be created and used to plug UPGR into CDB2 as new PDB1. Finally sanity operations will have to be done to assimilate UPGR finally as PDB2.

Please note: There's no ALTER PLUGGABLE DATABASE … RECONVERT command available. To migrate a database back into a stand-alone database either Data Pump, Transportable Tablespaces or similar techniques will need to be used.

SID: UPGR Oracle 12.2.0.1 home	SID: CDB2 Oracle 12.2.0.1 home

Prepare the UPGR database for plug in
 	
1. Switch to the UPGR Oracle 12.2.0.1 environment:
`. upgr122`
`sqlplus / as sysdba`
 	
2. **Start** the UPGR database in **read-only** mode:
`shutdown immediate`
`startup open read only;`
 	
3. Generate the XML description file – this file will contain the information describing the database structure. To create it the database UPGR has to be in read only mode:
`exec DBMS_PDB.DESCRIBE('/tmp/pdb1.xml');`
 	
4. Shutdown the database
`shutdown immediate`
`exit`	

SID: UPGR Oracle 12.2.0.1 home	SID: CDB2 Oracle 12.2.0.1 home

	Prepare the UPGR database for plug in
	
	1. Switch to the CDB2 Oracle 12.2.0.1 environment:
	
	`. cdb2`
	`sqlplus / as sysdba`
	
	2. The 12.2.0.1 Container Database CDB2 should already be started. If not, start it now:
	
	`startup`
	
	3. Check plug in compatibility first:
	
	`SET SERVEROUTPUT ON`
	
	`DECLARE`
	`compatible CONSTANT VARCHAR2(3) := CASE`
	`DBMS_PDB.CHECK_PLUG_COMPATIBILITY( pdb_descr_file => '/tmp/pdb1.xml',`
	`pdb_name => 'PDB1') WHEN TRUE THEN 'YES' ELSE 'NO'`
	`END;`
	`BEGIN`
	`DBMS_OUTPUT.PUT_LINE(compatible);`
	`END;`
	`/`
	
	4. Now plug in the database with its new name **PDB1** – from this point there's no UPGR database anymore. In a real world environment, you would have a backup or use a backup/copy to plug in. In our lab the database UPGR will stay in place and become PDB1 as part of CDB2.
	Please use the proposed naming as the FILE_NAME_CONVERT parameter and TNS setup have been done already.

Use the NOCOPY option for this lab to avoid additional copy time and disk space usage.

```
create pluggable database PDB1 using '/tmp/pdb1.xml' nocopy tempfile
reuse;
```

5. Connect to this new PDB1 and perform sanity operations:

```
alter session set container=PDB1;
@?/rdbms/admin/noncdb_to_pdb.sql
```

Sanity operations required inside the PDB to connect the PDB with the CDB correctly. Therefore, run the script `noncdb_to_pdb.sql` – this may take approximately 10-20 minutes to complete due to recompilations. If the script didn't get executed the PDB1 would open in restricted mode only.

6. Now the database UPGR is plugged in – but not open yet. It will need to be started.

```
startup
show pdbs
exit
```

7. To connect to the consolidated PDB1 from the command prompt the following command syntax needs to be used:

```
sqlplus "sys/oracle@pdb1 as sysdba"
```

As an alternative you could use the EZconnect syntax:

```
sqlplus "sys/oracle@//localhost:1521/pdb1 as sysdba"
```

*** COMPLETED *** Tasks HOL Part 2 ***

HOL – Part 3 – Migrate FTEX database with Full Transportable Export/Import into PDB2

Database files location:	/u02/oradata/FTEX
Pluggable database files location:	/u02/oradata/CDB2/pdb2
Initialization parameter and password file location:	/u01/app/oracle/product/11.2.0.4/dbs
Listener configuration:	/u01/app/oracle/product/12.2.0.1/network/admin

Tasks HOL Part 3

Full Transportable Export/Import is a new Oracle Database 12c upgrade and migration feature combining the speed of Transportable Tablespaces with the ease-of-use of Data Pump taking care of all metadata and non-transportable data.

Your task in HOL Part 3 will be to use Full Transportable Export/Import to migrate the existing Oracle 11.2.0.4 database FTEX into a new PDB2 which will belong to the container database CDB2. Please stay with the proposed names (PDB2) as the TNS setup has been set up already to allow connections etc.

This feature works independent of Oracle Multitenant and platform and can be used to migrate cross Endianness as well. Source database version has to be at least Oracle 11.2.0.3, target version needs to be at least Oracle 12.1.0.1. For cross-platform migrations RMAN backups with CONVERT operations will be necessary.

SID: FTEX – Oracle 11.2.0.4 home	SID: CDB2 – Oracle 12.2.0.1 home
`$> . ftex` `<dot> <space> ftex` for the Oracle 11.2.0.4 environment with your database to be plugged in later (SID: FTEX)	`$> . cdb2` `<dot> <space> cdb2` for the Oracle 12c environment connecting to the Container Database (SID: CDB2)

*** START HERE *** Migrate FTEX with Full Transportable Export/Import into PDB2 ***

The first task in the lab will be to provide an empty database – something we would do for a full import or for transportable tablespaces as well. But in this specific case we want to consolidate, and therefore pre-create an empty PDB2 (a Pluggable Database) inside the already existing CDB2 (the Container Database).

SID: FTEX Oracle 11.2.0.4 home	SID: CDB2 Oracle 12.2.0.1 home

	Provision PDB2 from PDB$SEED:
	
	1. Switch to the Oracle 12c **CDB2 environment**:
	
	`. cdb2`
	`sqlplus / as sysdba`
	
	`startup` **[May not be necessary if the database is already started]**
	
	2. **Create a new pluggable database PDB2**:
	
	The easiest way to create an empty PDB is to clone it from the template PDB called **PDB$SEED** which exists in every container database. The location to create it is defined by the init parameter `PDB_FILE_NAME_CONVERT` or by specifying file_name_convert when creating the new PDB. It is. For part 1 of the lab we used the init parameter to create CDB1. For this part we will specify that the files for PDB2 should be placed in the `/u02/oradata/CDB2/pdb2` directory.
	
	Create an empty PDB by cloning the PDB$SEED:
	
	`create pluggable database PDB2 admin user adm identified by adm`
	`file_name_convert=('/oradata/CDB2/pdbseed', '/oradata/CDB2/pdb2');`
	
	This will take 1-2 minutes.

	Start the new pluggable database PDB2:
	
	`alter session set container=pdb2;`
	`startup`
	
	3. Create a directory object and a database link inside the PDB2 – you will need this for the full transport operation - the directory `/u02/oradata/CDB2/mydir` has been precreated as well for Data Pump
	
	`create directory mydir as '/u02/oradata/CDB2/mydir';`
	`grant read, write on directory mydir to system;`
	`create public database link SOURCEDB connect to system identified by oracle using 'FTEX';`
	`exit`

SID: FTEX Oracle 11.2.0.4 home	SID: CDB2 Oracle 12.2.0.1 home

Prepare the FTEX database for the Full Transportable Export/Import
 	
In order to run the Full Transportable operation we must set all data tablespaces into read-only mode. This is the same procedure we would follow for a regular transportable tablespace operation. Once the tablespace is in read-only mode we can copy the file(s) to the target location
 	
1. Switch to the Oracle 12c UPGR environment:
 	
`. ftex`
`sqlplus / as sysdba`
 	
2. **Start** the **FTEX** database and switch data tablespaces (here: USERS) into **read-only** mode:	

```
startup
alter tablespace users read only;
exit
```

3. Copy the files to the target location

```
cp /u02/oradata/FTEX/users01.dbf /u02/oradata/CDB2/pdb2
```

SID: FTEX Oracle 11.2.0.4 home	SID: CDB2 Oracle 12.2.0.1 home

Data migration via Full Transportable Export/Import from FTEX into PDB2

The Data Pump import will be run through the database link you created earlier – thus no need for an export or a dumpfile. Data Pump will take care of everything (currently except XDB and AWR) you need from the system tablepaces and move views, synonyms, trigger etc over to the target database (in our case: PDB2).

1. Switch to the Oracle 12c **CDB2 environment**:

```
. cdb2
```

2. **Execute the Full Transportable Export/Import with Data Pump**

```
impdp system/oracle@pdb2 network_link=sourcedb version=12 full=y \
transportable=always metrics=y exclude=statistics directory=mydir \
logfile=pdb2.log \
transport_datafiles='/u02/oradata/CDB2/pdb2/users01.dbf'
```

In case copy&paste does not work we have prepared a par file in /home/oracle/IMP.

	The PDB2 is open and ready to use after the Transport migration has completed: `sqlplus "system/oracle@PDB2"`

*** COMPLETED *** Tasks HOL Part 3 ***

❹ [optional]

HOL – Part 4 – Create PDB3 in Oracle 12.1.0.2 and upgrade via plug out/in to Oracle 12.2.0.1

Database files location:	/u02/oradata/CDB1/pdb3
Pluggable database files location:	/u02/oradata/CDB2/pdb3
Initialization parameter and password file location:	/u01/app/oracle/product/12.1.0.2/dbs
Listener configuration:	/u01/app/oracle/product/12.2.0.1/network/admin

Tasks HOL Part 4

One technique to upgrade pluggable databases in a Multitenant environment is unplug-plugin. This approach gives a lot of control over a pluggable database upgrade but requires manual steps, similar to the command line upgrade.

In this part of the HOL a new PDB will be created in an Oracle 12.1.0.2 CDB1 and upgraded via unplug/plugin into the Oracle 12.2.0.1 CDB2.

SID: CDB1 – Oracle 12.1.0.2 home	SID: CDB2 – Oracle 12.2.0.1 home
`$> . cdb1` `<dot> <space> cdb1` for the Oracle 12.1.0.2 environment with your database to be plugged in later (SID: CDB1)	`$> . cdb2` `<dot> <space> cdb2` for the Oracle 12.2.0.1 environment connecting to the Container Database (SID: CDB2)

*** START HERE *** Create PDB3, upgrade it to Oracle 12.1.0.2 via plug out/in ***

The first task in the lab will be to provide an empty database – something we would do for a full import or for transportable tablespaces as well. But in this specific case we want to consolidate, and therefore pre-create an empty PDB2 (a Pluggable Database) inside the already existing CDB2 (the Container Database).

SID: CDB1 ➔ PDB3 Oracle 12.1.0.2 home	SID: CDB2 ➔ PDB3 Oracle 12.2.0.1 home

Create a new pluggable database PDB3:
 	
1. Switch to the Oracle 12.1.0.2 **CDB1 environment**:
 	
`. cdb1`
`sqlplus / as sysdba`
 	
2. **Start** the **CDB1** container database – it has no PDBs yet (except for PDB$SEED):
 	
`startup`
 	
3. **Create a new pluggable database PDB3 and start it**:
 	
`create pluggable database PDB3 admin user adm identified by adm`
`file_name_convert=('/u02/oradata/CDB1/pdbseed',`
`'/u02/oradata/CDB1/pdb3');`
This will take 1-2 minutes.
 	
`alter session set container=pdb3;`
`startup`	

4. **Open a second xterm** side by side and **run the new preupgrade.jar** in your 12.1.0.2 environment on this container PDB3 only:

```
. cdb1
java -jar $OH12/rdbms/admin/preupgrade.jar -c 'pdb3' TERMINAL TEXT
```

5. **Verify the output (scroll from top to bottom)** and make necessary changes in SQL*Plus (in other xterm) including dictionary statistics – then execute the preupgrade_fixups.sql:

```
@/u01/app/oracle/cfgtoollogs/CDB1/preupgrade/preupgrade_fixups.sql
```

6. Switch to the CDB$ROOT layer, **close** the pluggable database PDB3 and **unplug** it

```
alter session set container=CDB$ROOT;
alter pluggable database PDB3 close;
alter pluggable database PDB3 unplug into '/tmp/pdb3.xml';
drop pluggable database PDB3 keep datafiles;

shutdown immediate
exit
```

SID: CDB1 ➔ PDB3 Oracle 12.1.0.2 home	SID: CDB2 ➔ PDB3 Oracle 12.2.0.1 home

	Plug in the PDB3 into CDB2 and upgrade it to Oracle 12.2.0.1:
	
	1. Switch to the Oracle 12.2.0.1 **CDB2 environment**:
	
	`. cdb2`
	`sqlplus / as sysdba`

2. Execute the **Plug In Check a**nd **check PDB_PLUG_IN_VIOLATION**S :

```
SET SERVEROUTPUT ON

DECLARE
  compatible CONSTANT VARCHAR2(3) := CASE
DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  pdb_descr_file => '/tmp/pdb3.xml',
  pdb_name => 'PDB3')
  WHEN TRUE THEN 'YES' ELSE 'NO'
END;
BEGIN
DBMS_OUTPUT.PUT_LINE(compatible);
END;
/


select message, status from pdb_plug_in_violations where type like
'%ERR%';
```

3. Plug in the PDB3 into CDB2

```
create pluggable database pdb3 using '/tmp/pdb3.xml'
file_name_convert=('/u02/oradata/CDB1/pdb3', '/u02/oradata/CDB2/pdb3');
```

4. Open PDB3 in UPGRADE mode and upgrade it

```
alter pluggable database PDB3 open upgrade;
exit

cd $ORACLE_HOME/rdbms/admin
$ORACLE_HOME/perl/bin/perl catctl.pl -c 'PDB3' catupgrd.sql
```

5. Recompile after upgrade

```
sqlplus / as sysdba

alter session set container=PDB3;
startup
@?/rdbms/admin/utlrp.sql
show pdbs
```

6. **Verify the output (scroll from top to bottom)** and make necessary changes in SQL*Plus (in other xterm) including dictionary statistics – then execute the preupgrade_fixups.sql:

```
@/u01/app/oracle/cfgtoollogs/CDB1/preupgrade/postupgrade_fixups.sql
```

7. Exit from SQL*Plus:

```
exit
```

NN	SID: CDB2 ➔ PDB3 Oracle 12.2.0.1 home

Finally a few CDB/PDB exercises
First test will introduce you to the new CDB views. Therefore we create a simple table and check its visibility within the dictionary views

1. Connect directly to **PDB1 in the Oracle 12.2.0.1 environment**:

```
. cdb2
sqlplus "sys/oracle@//localhost:1521/pdb1 as sysdba"
```

2. Create a table and insert data:

```
create table HOL (col1 number);
insert into HOL values (1);
commit;
```

3. Connect directly to **PDB2**:

```
alter session set container=PDB2;
```

4. Create a table and insert data:

```
create table HOL (col1 number);
insert into HOL values (2);
commit;
```

5. Connect directly to **PDB3**:

```
alter session set container=PDB3;
```

6. Create a table and insert data:

```
create table HOL (col1 number);
insert into HOL values (2);
commit;
```

7. Connect directly to **CDB$ROOT**:

```
alter session set container=cdb$root;
```

8. Query the data from the CDB_Views:

```
select CON_ID, SUBSTR(TABLE_NAME,1,10) TNAME from CDB_TABLES WHERE TABLE_NAME='HOL';
```

You'll see that each table HOL within a certain PDB is visible to the CDB$ROOT. But if you'd repeat the exercise within each of the PDBs you'll see just the contents on a PDB level. Recognize the CON_ID which represents where an object exists.

*** COMPLETED *** Tasks HOL Part 4 ***

Thank your for completing our Upgrade, Migrate & Consolidate to Oracle Database 12c Hands-On-Lab.

If you have further questions you may please download the +500 slide deck containing almost everything about upgrades and migrations. And always feel free to contact us directly.

http://blogs.oracle.com/UPGRADE

Thanks and successful upgrades!

Roy Swonger & Mike Dietrich & The Database Upgrade Team