ORACLE COOKBOOK

Oracle Database 12c New Features

I. Multitenant Database

Technical Solution Consulting, Oracle Korea
Version : 1.0

Document Control

Change Record

Date	Author	Version	Change Reference

Reviewers

Name	Position

Distribution

Copy No.	Name	Location
1		
2		
3		
4		

Contents

DOCUMENT CONTROL 2

I. ARCHITECTURE OF MULTITENANT DATABASE 4

MULTITENANT DATABASE 5
The benefits of Multitenant Database 5
Designed for the Cloud 5
Efficient Consolidation 5
Rapid Provision and Cloning 5
Fast Upgrade and Patching 5
Manage Many Databases As One 5
Multitenant Database Architecture 6
Multitenant Database Architecture Diagram 6
Container Database(CDB) 7
Pluggable database(PDB) 9
Users 10
Privilege and Role 10
Priority 12
Listener 13

Ⅱ. BASIC TASKS ON CDB AND PDB 14

Connect to the CDB root or to a PDB 14
Create a new PDB from the seed PDB 15
Create a directory for new datafile and tempfile 15
Manage the CDB and the PDBs 15
Manage the CDB 15
Manage the PDBs 16
Rename a PDB 18
Manage the storage in a CDB and its PDBs 18
Manage security in PDBs 22
Manage the Common and local users 22
Manage the common and local roles 23
Manage the common and local privileges 23
Drop PDBs 25

III. CLONE PLUGGABLE DATABASE 26

Clone Pluggable Database Overview 26
Cloning Types 26
Prepare the Source PDB to Clone 28
Set the source PDB in READ ONLY mode 28
Create a directory for the new clone PDB 28
Configure OMF to the directory of the clone PDB 29
Clone the PDB Within the CDB 29
Set the Source PDB Back to Open Mode 29
Clean Up the Environment 29

I. Architecture of Multitenant Database

Multitenant Database

Oracle Database 12c Enterprise Edition option인 Oracle Multitenant는 application에 영향을 주지 않고 여러 database들을 손쉽게 통합 할 수 있도록 설계되어 있다. 이러한 multitenant database는 다수의 데이타베이스가 가지는 모든 이점을 하나로 집중시킴으로써 리소스의 낭비를 최소화 할 뿐 아니라 우선순위를 효율적으로 관리할 수 있다. 또한 oracle Multitenant는 RAC와 ADG를 포함한 오라클의 option에도 최고의 호환성을 제공하며 빠른 업그레이드와 프로비저닝을 제공한다.

The benefits of Multitenant Database

Designed for the Cloud

Oracle Multitenant는 Cloud system의 핵심을 그대로 담고 있다. admin user가 manage하는 multinenant contatiner database(CDB)의 Root Container에 Pluggable database를 plug함으로써 분산되어 있는 많은 데이타베이스의 리소스들을 하나로 통합하여 관리할 수 있다. 따라서 고객들은 Oracle database 12로 산재되어 있는 데이타베이스를 private cloud로 손쉽게 통합하여 통합된 데이타베이스들의 리소스를 효과적으로 관리할 수 있다.

Efficient Consolidation

전통적으로 많은 IT회사들은 데이터베이스 통합을 위해 가상화와 클러스터링 기술을 사용해오고 있다. 데이터베이스를 통합하는 가장 일반적인 방법은 application을 재개발하여 스키마를 통합하는 방법에 주의를 기울여 왔다. 하지만 이러한 방법은 결국 높은 개발비용과 관리비용, 제한된 Consolidation density를 초래하였다. 오라클은 Multitenant database container에 데이터베이스들을 plugging시킴으로써 어플리케이션단의 변경없이 Consolidation process를 단순화하였다.

이러한 12c의 새로운 개념의 아케텍처는 오직 Container Database만이 memory와 background process 관리함으로써 통합에 주의를 기울여 온 IT기업들이 가지고 있던 보안에 대한 문제 없이 확장성과 consolidation density를 최소화 하였다.

Rapid Provision and Cloning

테스팅, 개발, 문제진단 등을 위한 빠른 프로비저닝과 데이타베이스 복제는 많은 IT기업에게는 큰 도전일 수 있다. 왜냐하면 특히 개발 및 테스트를 구성하기 위해서는 적게는 수십대에서 많게는 수백대의 데이터베이스가 요구된다. 이 경우 서버관리자들은 이기종 서버들간에 데이타베이스를 이동하거나 복제, 생성하기 위해서 많은 시간과 노력이 필요한다. 12c의 경우, SQL 한 문장(One command)으로 데이터베이스 프로비저닝과 복제를 할 수 있다.

.

Fast Upgrade and Patching

많은 데이타베이스 관리자들에게 업그레이드와 패치적용은 데이타베이스를 운영하는데 중요한 부분이다. 현재까지는 업데이트와 패치들은 모두 개별 대이테베이스에 각각 하나씩 적용이 되었다. 하지만 오라클의 multinenant는 Root CDB에만 업데이트 및 패치를 적용한 뒤 각각의 PDB에 일괄적으로 적용함으로써그 과정이 단순화되었을 뿐 아니라 굉장히 빠른 속도로 적용가능하다.

하지만 어떤 데이타베이스 관리자는 기존 데이타베이스의 특정 일부분을 update나 patch하고 싶거나 모든 pluggable database들을 모두 동시에 적용하고 싶진 않을 수 있다. 이럴경우에도 new updated multitenant container database를 생성하고 업데이트나 패치가 필요하지 않은 데이타베이스는 기존 container database에서 unplug하여 기존의 plug되어 있는 database만을 적용할 수 있다.

Manage Many Databases As One

multitenant database를 통합할 때의 명백한 이점은 수많은 데이터베이스를 통합적으로 관리하는 것이다. 뿐만 아니라 산재되어 있는 데이타베이스들을 한 번의 수행으로 백업을 완료할 수도 있다. 과거에는 다른 지역에 위치한

data center로 데이타를 백업 할 경우 standby database system에 별도의 셋업이 필요했지만, multitenant database의 경우에는 모든 pluggable database들을 replicate하면 백업이 완료된다.

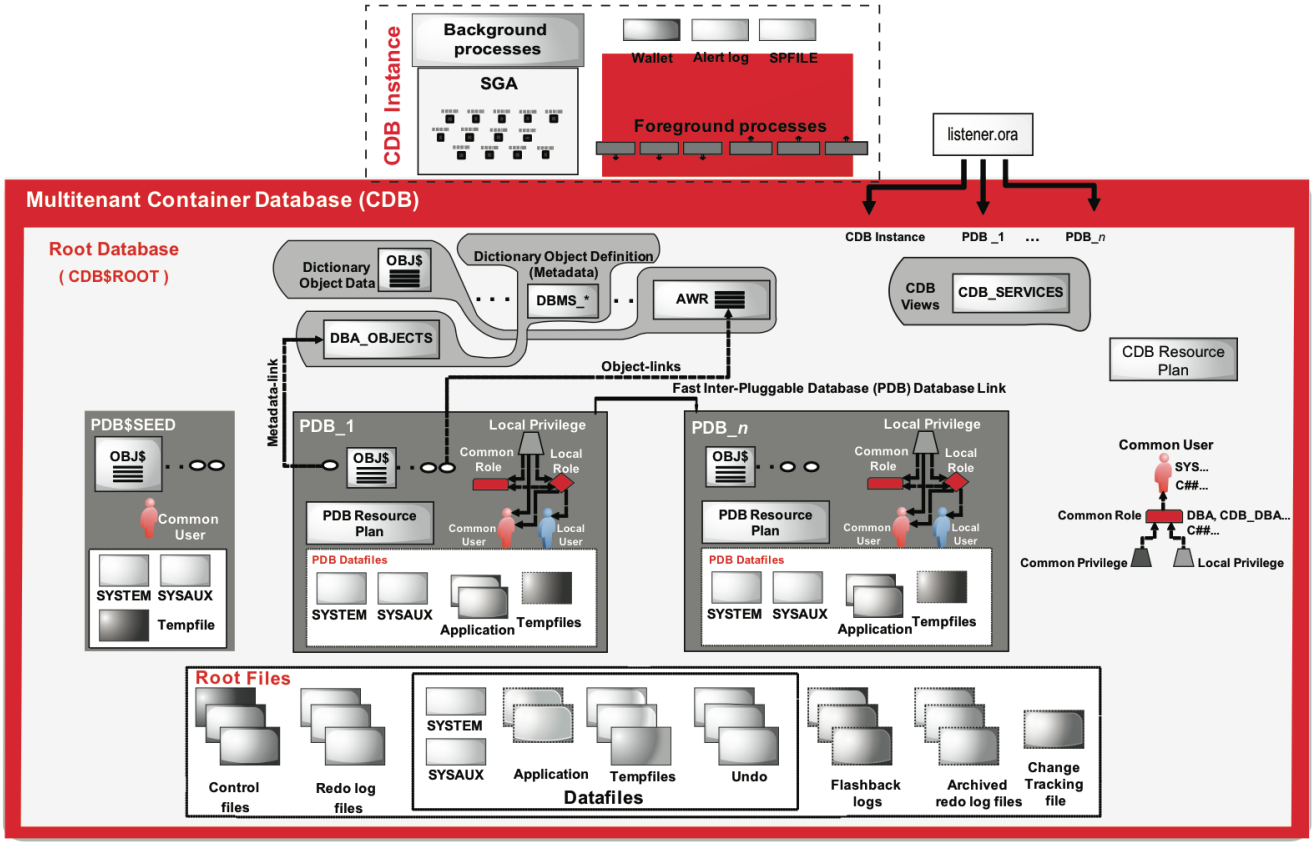
Multitenant Database Architecture

Oracle 12c는 많은 super database안에 많은 sub database를 가질 수 있는 새로운 아키텍처이다. super database를 공식적으로 multitenant database라는 용어로 사용하였고 이를 줄여서 CDB라고 부른다. Sub database는 pluggable database로, 줄여서 PDB라고 부른다. 따라서 하나의 CDB 내의 다수의 PDB가 하나의 instance를 공유하는 구조가 12c의 multitenant database이다.

관리의 편리성을 위해 각각의 데이터베이스에 Numbering을하여 쉽게 구분하였다.

DATABASE TYPE	Container ID
Multitenant Container Database or CDB	CON_ID 0
CDB$ROOT	CON_ID 1
PDB$SEED	CON_ID 2
PDB_1	CON_ID 3
PDB_2	CON_ID 4
PDB_n	CON_ID n

Multitenant Database Architecture Diagram

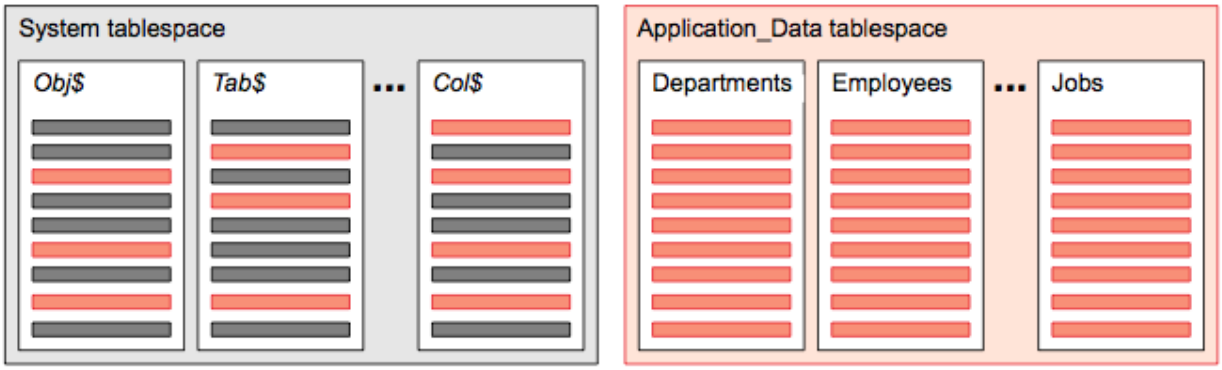


<그림 1-1> Total Multitenant Database Architecture Diagram

Container Database(CDB)

Data dictionary

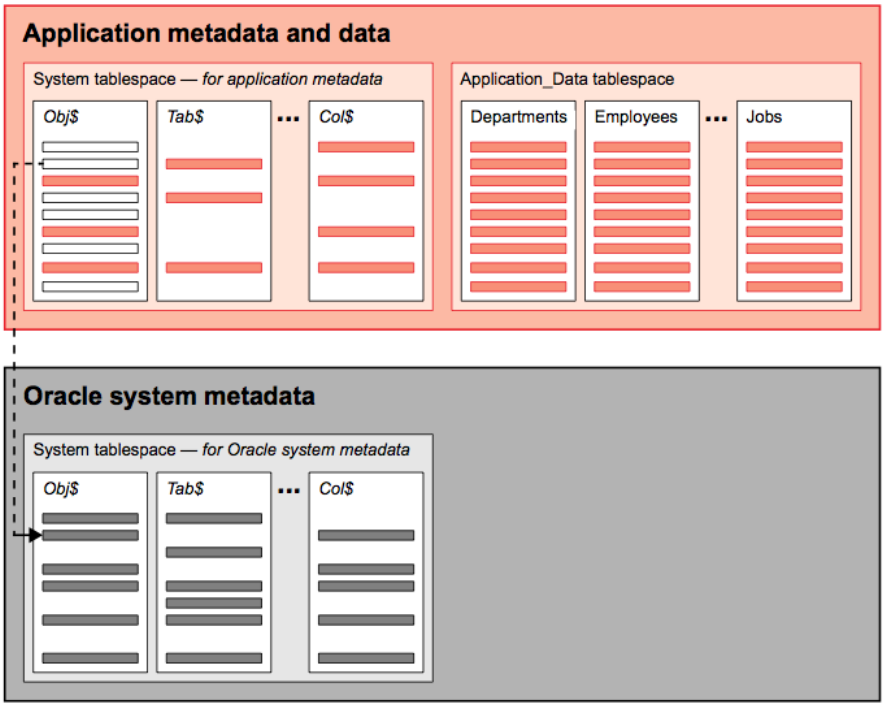
Pre-12c의 경우, user application data가 insert되면 application의 종속성을 유지하기 위해 insert된 row에 대한 metadata가 system tablespace에 horizontally 추가된다. 즉 고객이 데이터를 추가하기 되면 data dictionary metadata내에는 system metadata와 user data에 대한 metedata가 공존한다.(아래의 그림 참조) 따라서 Non-CDB구조에서는 User$, Edition$, TS$등에 의해 지정되는 objects들의 이름 뿐 아니라 SysAuth$, ObjAuth$ table에 저장된 user와 role의 이름은 유일한 값으로 해당 object를 대표하기 때문에 2개 이상의 application이 Non-CDB에 설치될 경우 병합이 발생할 수도 있고 보안의 문제 또한 발생할 수 있다.



<그림 1-2> pre-12c Data Dictionary Architecture

12c의 multitenant architecture는 이에 대한 문제를 해결하고자 data dictionary에 대한 가상화를 실현하였다. System dictionary에 공존하던 metadata를 논리적 파티셔닝[1]하여 system metadata(ROOT)와 user_data metadata(PDB)로 분리하였다. 즉, 각 PDB내에는 독립적인 metadata를 두고 ROOT medata dictionary에는 Oracle system metadata와 각 PDB의 metadata를 참조하는 값만이 기록된다. 따라서 실제로는 2개의 data dictionary table(CDB and PDB)이 존재하지만, 고객이 dictionary view를 조회해보면 두 개의 테이블이 하나로 통합되어 보여진다.

[1] data dictionary를 horizontally partition하여 logical tagging schema가 physically하게 수행된다.



<그림 1-3> 12c Data Dictionary Architecture

이렇듯 12c의 data dictionary 가상화는 CDB내에 독립된 PDB를 plug-in할 때 PDB에 대한 metadata만 System data dictionary에 plug-in되기 때문에 실시간에 가까운 성능으로 다수의 Pluggable database를 구성할 수 있다.

즉, ROOT CONTAINER에 DATA DICTIONARY정보를 한 곳에 집중시킴으로써 많은 PDB들이 Plug-in되었을 때 발생할 수 있는 중복을 줄였다. PDB의 System Metadata를 CDB에 별도로 저장하고 User data와 user metadata만을 가지기 때문에 PDB의 경량화를 실현하였다.

---

non-CDB의 data dictionary의 경우 약 700M이상의 공간이 필요하지만, Multitenant Database의 경우 약 200M가 필요하다.

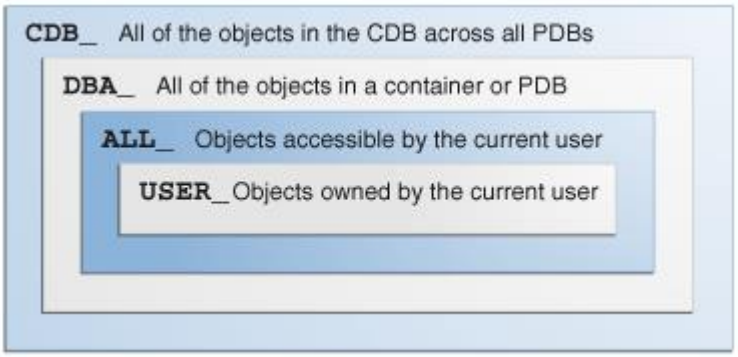
User application dictionary(PDB)와 Oracle system dictionary(CDB)는 복수의 링크 구조를 가진다.

- Data dictionary links
  Metadata-link
  CDB와 PDB의 Dictionary Object metadata를 연결한다. 예를들어 OBJ$ dictionary table(DBA_OBJECTS)의 metadata는 root container에만 기록되고 PDB는 pointer정보로 CDB내의 PDB system metadata를 연결한다.

- Object-link
  CDB와 PDB간 DICTIONARY OBJECT DATA(not metadata)를 연결한다. 예를들어 AWR DATA는 ROOT에 기록이 되고 각각의 PDB는 root의 AWR data를 참조한다.

각각의 독립된 PDB 내에 저장되는 User application data 및 datafile의 name&path는 PDB 내에서 유일한 값이어야 하고 해당 tablespace와 segment의 이름은 user metadata(PDB) 뿐만 아니라 참조하고 있는 system metadata(ROOT)와도 동일해야 한다.

- Relationship of Dictionary views

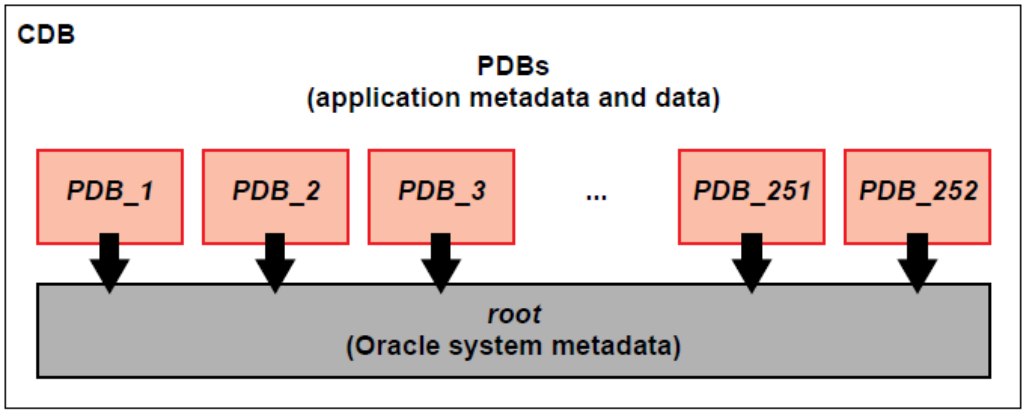


<그림 1-4> Relationship of Dictionary views

Pre- 12c의 경우, system metadata를 조회하기 위해 DBA_ view를 사용하였다. Multitenant database는 각 PDB에 대한 metadata를 조회할 수 있는 DBA_ view 뿐 아니라 특정권한이 있는 common user가 CDB_ view를 조회한다면 CDB 뿐 아니라 PDB에 대한 metadata정보도 한꺼번에 조회할 수 있다.

CDB$ROOT

CDB$Root는 Root Container라고도 불리며 Tablespaces, datafiles, oracle system에 대한 metadata를 가지는 meta-database이다. User application data 없이 data dictionary를 공유한다는 것에는 CDB와 비슷한 점이 많지만 container는 foreground process는 각각의 PDB들에 의해 생성되는 session을 연결하여 리소스를 최적화 한다는 것에서 root보다는 superclass로써 정의된다.



<그림 1-5> Relationship of CDB$Root and PDBs

Pluggable database(PDB)

PDB는 하나의 CDB의 root에서 unplug해 다른 CDB의 root로 plug-in하기 때문에 pluggable database로 정의된다. 즉 하나의 Container Database에 여러 많은 Pluggable database를 가질 수 있고 PDB는 12c 이하 버전의 기존 DB(이하 non-CDB) 에 대해서도 모두 호환 가능하다. 각 세션들은 PDB에 접속이 가능하고 Application 레벨에서는 기존 database처럼 동일하게 접속이 가능하다. 또한 System Admin이 Database를 보았을 때 하나의 이미지로만 보이고 다수의 pluggable database는 Resource Manager에 의해 resource를 관리할 수 있도록 설계되어 있다.

PDB$SEED

PDB를 복제할 때 사용되는 일종의 Template.(최대 250개의 PDB 복제 가능)
SYSTEM, SYSAUX, TEMPFILE로 구성

PDB의 특징

- 신규 PBD생성이나 기존 PBD복사 시 매우 빠른 Provisioning 이 가능하다. redeployment (Unplug 와 Plug) 이 매우 빠르다 (약 13초). 따라서 패치나 업그레이드가 매우 빠르다. (낮은 버젼의 CDB 에서 unplug해서 상위 버전의 CDB로 plug-in)
- PDB-PDB , CDB-PDB 간에 Access가 구분되어 있어 보안을 보장한다.
- 기존 DB 를 consolidation 할 때 보다 더 많은 DB를 consolidate할 수 있다.
- 단일화 되므로 중앙 집중적으로 관리 가능하고 DR 이나 Backup & Recovery 를 위한 관리 포인트도 줄어든다.
- Scale-out 이 용이하고 CDB-CDB 형식의 RAC를 구성하여 가용성이 보장된다.

PDB 생성방법

- PDB$SEED로부터 생성 (Cloning 했을 경우에는 Clone 한 DB의 속성을 그대로 상속받는다).
- non-CDB 로부터 생성
- 기존 PDB로부터 Cloning
- 다른 CDB에서 unplug 하여 새로운 CDB에 plug in

Users

CDB에서 local user와 common user로 구분된다.

Common user

- ROOT Container에는 common user를 만드는 순간 plug-in 된 각 PDB들에 자동적으로 생성되어 PDB의 접속권한이 생성된다.(root 뿐 아니라 다른 PDB에서도 common user가 보임)
- Common user는 CDB의 admin 역할을 할 뿐 아니라 한번의 작업으로 모든 PDB에 replicate되어 관리적 편의성을 제공한다.
- 적절한 권한이 주어질 경우, CDB를 startup하거나 PDB를 open하는 등의 작업이 가능하다.
- 권한이 있는 common user는 PDB간 접근이 가능하다.
- 권한은 권한을 부여한 컨테이너 안에서만 유효하다.

Local user

- PDB 외부에서는 참조가 불가능하다.
- 해당 스키마가 정의된 PDB에만 접근가능하다.
- Local user 정보는 각 PDB의 dictionary에 저장되고 저장된다.

Privilege and Role

Non-CDB와 같이 CDB에서도 role과 privilege는 user에 의해서 별도의 제한 없이 부여되었다. 하지만, CDB에서 role과 privilege를 common과 local로 분리하여 privilege를 줄 수 있도록 하였고 또한 이는 한 user나 role에 종속되는 것이 아니라 sql구문에 옵션을 붙여 독립적으로 수행되도록 하였다. 따라서 privilege를 위해서는 다음과 같은 옵션을 수행하여야 한다. . 따라서 common role을 가진 user일지라도 특정 PDB에는 privilege가 없을 수 있다.

Privilege type	Option	Coverage
Local privilege	CONTAINER = CURRENT	CURRENT CONTAINER
Common Privilege	CONTAINER = ALL	CURRENT CONTAINER, FUTURE CONTAINER

※GRANTOR가 특정 ROLE이나 PRIVILEGE를 부여하기 위해서는 반드시 'GRANT OPTION'이 필요하다.

<Test Script> - Granting Roles and Privileges

12c의 경우, Privilege와 role이 분리되어 있다. Privilege는 role에 대한 dependency가 없다는 것을 스크립트를 통해 알아보도록 한다.

Root Container의 common user인SYSTEM계정으로 로그인 후 COMMON USER(c##dba)생성한다.
Root container로 CREATE SESSION 권한을 준다.

```
cdb01$SYS> connect system/welcome1@cdb01
Connected.
cdb01$SYSTEM> create user c##dba identified by welcome1 container=all;

User created.

Elapsed: 00:00:00.13
cdb01$SYSTEM> grant create session to c##dba;

Grant succeeded.

Elapsed: 00:00:00.01
```

CONTAINER=ALL 구문으로 c##admin이라는 common role를 생성한다.
생성 후 SELECT ANY TABLE 권한을 c##admin role에 부여한다. (실제 권한을 줄 때 CONTAINER=ALL구문을 사용하지 않으면 자동적으로 local user인 root container에 권한이 지정된다)
마지막으로 생성한 role을 c##dba에 부여한다. (c##admin은 common role이다. 설령 pdb에 접속하여 CONTAINER=CURRENT 구문을 주더라도 해당 ROLE은 지정될 수 없다.)

```
cdb01$SYSTEM> create role c##admin container=all;

Role created.

Elapsed: 00:00:00.02
cdb01$SYSTEM> grant select any table to c##admin;

Grant succeeded.

Elapsed: 00:00:00.01
cdb01$SYSTEM> grant c##admin to c##dba;

Grant succeeded.

Elapsed: 00:00:00.01
```

하나의 pluggable database에 접속한다. 하지만 다음과 같은 에러메시지가 나온다. 왜냐하면 CREATE SESSION구문을 ROOT CONTAINER에서 생성했기 때문에 PDB에서는 접속이 불가하다.
시스템 계정으로 PDB에 접속해 CONNECT, RESOURCE role을 common user인 c##dba에게 준다.
(CONTAINER=ALL이라는 구문이 없으므로 PDB01 LOCAL에 주어진다.)

```
cdb01$SYSTEM> connect c##dba@pdb01
Enter password:
ERROR:
ORA-01045: user C##DBA lacks CREATE SESSION privilege; logon denied


cdb01$SYSTEM> connect system@pdb01
Enter password:
Connected.
pdb01$SYSTEM> grant connect,resource to c##dba;

Grant succeeded.

Elapsed: 00:00:00.01
```

특정 테이블에 대해 조회를 시도해보지만 c##dba는 pdb01에 대해 테이블조회에 대한 특정 권한이 없기 때문에 에러메시지가 나오게 된다. SELECT ANY TABLE이라는 권한이 주어졌지만 이는 ROOT에 대해서만 형성이 되어 있어 PDB01에는 해당 권한이 없다.

```
pdb01$SYS> connect c##dba@pdb01
Enter password:
Connected.
pdb01$C##DBA>  SELECT COUNT(*) FROM hr.employees;
select * from hr.employees SELECT COUNT(*) FROM hr.employees
                           *
ERROR at line 1:
ORA-00942: table or view does not exist


Elapsed: 00:00:00.00
```

따라서 Root Container로 접속해 SYSTEM계정으로 CONTAINER=ALL구문으로 COMMON PRIVILEGE를 부여한다.

```
cdb01$SYSTEM> grant select any table to c##admin container=all;

Grant succeeded.

Elapsed: 00:00:00.13
```

PDB01에 접속하여 다시 조회를 해보지만 역시 에러가 발생한다. 그 이유는 위에서 c##admin role은 local root container의 c##dba계정으로 주어졌기 때문이다.(role을 부여할 때 CONTAINER=ALL을 사용하지 않았다)

```
pdb01$C##DBA> SELECT COUNT(*) FROM hr.employees
  2  ;
SELECT COUNT(*) FROM hr.employees
                     *
ERROR at line 1:
ORA-00942: table or view does not exist


Elapsed: 00:00:00.00
```

다시 Root Container의 SYSTEM계정으로 접속 후 all container에 적용되도록 CONTAINER=ALL구무능ㄹ 사용하여 C##ADMIN role을 c##dba에게 부여한다.
이제 pdb01로 접속하여 조회해보면 결과값을 추출해 낼 수 있다.

```
cdb01$SYSTEM> GRANT c##admin TO c##dba
  CONTAINER=ALL;  2

Grant succeeded.

Elapsed: 00:00:00.02
cdb01$SYSTEM> connect c##dba@pdb01
Enter password:
Connected.
pdb01$C##DBA> select count(*) from hr.employees;

  COUNT(*)
----------
       107

Elapsed: 00:00:00.00
```

Priority

CDB를 생성하게되면 ROOT CONTAINER에 CDB RESOURCE PLAN이 생성되는데 이는 각각의 PDB의 우선순위가 정의되고 각 PDB의 RESOURCE PLAN에는 LOCAL RESOURCE에 대한 우선순위가 정의되어 있다.(그림 1-1참조)

Listener

Listener생성 시 CDB Listener가 생성되며 이는 ROOT CONTAINER를 LISTEN한다. 다른 PDB의 경우 PDB가 plug-in될 때 listener는 CDB_Service view를 참조해 listener에 pdb listener를 자동생성한다.(그림 1-1참조)

II. Basic tasks on CDB and PDB

Connect to the CDB root or to a PDB

CDB는 단지 컨테이너로서 database를 구분하는 저장소의 역할이기보다는 CDB라는 서비스를 생성하는 것이다. 따라서 Plug-In되는 PDB의 이름과 동일한 이름의 서비스 이름을 사용한다. 따라서 PDB로 세션 접속 시 서비스 이름으로 간편하게 접속하며 세션접속정보는 tnsnames.ora에 기록이 된다.

```
SQL*Plus: Release 12.1.0.1.0 Production on Tue Jul 9 16:42:37 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

cdb01$SYS> show con_name

CON_NAME
------------------------------
CDB$ROOT
cdb01$SYS> show con_id

CON_ID
------------------------------
1
```

서비스 가능한 root와 PDB조회

```
cdb01$SYS> select name, con_id from v$active_services order by 1;


NAME                                                             CON_ID
---------------------------------------------------------------- ----------
SYS$BACKGROUND                                                        1
SYS$USERS                                                             1
cdb01                                                                 1
cdb01XDB                                                              1
pdb01                                                                 3

Elapsed: 00:00:00.00
```

Tnsnames.ora에 PDB service가 등록되어 있다면, Easy Connect를 이용해 PDB에 연결한다.

```
cdb01$SYS> conn sys/welcome1@localhost:1521/pdb01 as sysdba
Connected.
localhost:1521/pdb01$SYS> show con_name

CON_NAME
------------------------------
PDB01
localhost:1521/pdb01$SYS> show con_id

CON_ID
------------------------------
3
```

Create a new PDB from the seed PDB

seed PDB에서 새로운 PDB를 생성한다. CDB 구성 후 PDB$Seed는 template형태로 자동적으로 CDB에 생성된다.

Create a directory for new datafile and tempfile

```
mkdir /u01/app/oracle/oradata/cdb1/pdb3
```

```
cdb01$SYS> create pluggable database pdb3
  2  admin user odb3_admin identified by oracle role=(CONNECT, RESOURCE)
  3* file_name_convert=('/u01/app/oracle/oradata/cdb01/pdbseed','/u01/app/oracle/oradata/cdb1/pdb3')
Pluggable database created.
```

CDB내의 모든 PDB들의 service name, open_mode, status를 확인한다.

```
cdb01$SYS> select name, open_mode from v$pdbs;


NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          MOUNTED
PDB3                           MOUNTED

Elapsed: 00:00:00.00
cdb01$SYS> cdb01$SYS> select name, con_id from v$active_services order by 1;


NAME                                                             CON_ID
---------------------------------------------------------------- ----------
SYS$BACKGROUND                                                        1
SYS$USERS                                                             1
cdb01                                                                 1
cdb01XDB                                                              1
pdb01                                                                 3
pdb3                                                                  4

6 rows selected.

Elapsed: 00:00:00.01
```

새롭게 생성된 PDB의 open_mode는 MOUNTED를 유지한다.
이제 다음 섹션에서 어떻게 PDB를 OPEN하는지, PDB의 NEW SERVICE는 어떻게 생성되는지 살펴본다.
DATAFILE LIST OF PDB

```
cdb01$SYS>  select name from v$datafile where con_id=4;

NAME
------------------------------------------------------------------
/u01/app/oracle/oradata/cdb1/pdb3/system01.dbf
/u01/app/oracle/oradata/cdb1/pdb3/sysaux01.dbf
```

Manage the CDB and the PDBs

CDB를 startup / shutdown, PDB를 open/close하는 방법을 테스트해본다.

Manage the CDB

가장먼저 root가 SYSDBA권한인지 확인한다
Shutdown CDB -> ALL PDB CLOSE -> CONTROL FILE DISMOUNT -> INSTANCE DOWN

```
cdb01$SYS> show con_name

CON_NAME
------------------------------
CDB$ROOT
cdb01$SYS> shutdown immediate
Database closed.
Database dismounted.
```

```
ORACLE instance shut down.
```

CDB를 startup

instance를 처음으로 start하면 control file이 mount되고 마지막으로 root container만 open된다. 본 operation은 SYSDBA 혹은 SYSBACKUP 권한이 요구된다.

```
cdb01$SYS> STARTUP
ORACLE instance started.

Total System Global Area 3707904000 bytes
Fixed Size                  2294936 bytes
Variable Size             855640936 bytes
Database Buffers         2835349504 bytes
Redo Buffers               14618624 bytes
Database mounted.
Database opened.
```

PDB가 open_mode인지 확인

(여기서 PDB01이 open되어 있다면, database trigger를 이용해 자동으로 pdb를 오픈하도록 설정된 경우이다.)

```
cdb01$SYS> STARTUP
ORACLE instance started.

Total System Global Area 3707904000 bytes
Fixed Size                  2294936 bytes
Variable Size             855640936 bytes
Database Buffers         2835349504 bytes
Redo Buffers               14618624 bytes
Database mounted.
Database opened.
cdb01$SYS> select name, open_mode from v$pdbs;


NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          MOUNTED
PDB3                           MOUNTED
```

Manage the PDBs

CDB가 start-up한 뒤 어떻게 한꺼번에 모든 PDB들을 open하는지 살펴보겠다.

```
cdb01$SYS> alter pluggable database pdb01 open;

Pluggable database altered.

Elapsed: 00:00:04.24
cdb01$SYS> select name, open_mode from v$pdbs;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          READ WRITE
PDB3                           MOUNTED
```

Open all PDBs at once

```
cdb01$SYS> alter pluggable database all open;


Pluggable database altered.

Elapsed: 00:00:06.93
cdb01$SYS> cdb01$SYS> select name, open_mode from v$pdbs;


NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          READ WRITE
PDB3                           READ WRITE
```

Close a PDB

```
cdb01$SYS> cdb01$SYS> alter pluggable database pdb01 close immediate;

Pluggable database altered.

Elapsed: 00:00:01.34
cdb01$SYS> select name, open_mode from v$pdbs;


NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          MOUNTED
PDB3                           READ WRITE
```

Close all PDBs at once.

```
cdb01$SYS> cdb01$SYS> alter pluggable database all close immediate;

Pluggable database altered.

Elapsed: 00:00:00.31
cdb01$SYS> select name, open_mode from v$pdbs;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          MOUNTED
PDB3                           MOUNTED
```

CDB instance가 시작될 때 PDB들은 보통 자동적으로 open되지 않는다. 개발환경에서는 instance와 함께 모든 PDB들이 한꺼번에 open되는 것이 편리하다. CDB가 시작될 때 모든 PDB가 자동적으로 open되는 trigger를 생성해본다.

- 모든 PDB들이 자동적으로 open되는 trigger를 확인하기 위해 CDB를 shutdown/startup

```
cdb01$SYS> create or replace trigger Sys.After_Startup after startup on database
begin
   execute immediate 'alter pluggable database all open';
end After_Startup;
/
Trigger created.

Elapsed: 00:00:00.16
cdb01$SYS> cdb01$SYS> shutdown immediate


Database closed.
Database dismounted.
ORACLE instance shut down.
cdb01$SYS> cdb01$SYS> cdb01$SYS> startup


ORACLE instance started.

Total System Global Area 3707904000 bytes
Fixed Size                  2294936 bytes
Variable Size             855640936 bytes
Database Buffers         2835349504 bytes
Redo Buffers               14618624 bytes
Database mounted.
Database opened.
cdb01$SYS> select name, open_mode from v$pdbs;



NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          READ WRITE
PDB3                           READ WRITE
```

```
Elapsed: 00:00:00.01
cdb01$SYS> cdb01$SYS> cdb01$SYS>
```

Rename a PDB

PDB를 restricted mode로 open한다.

```
cdb01$SYS> alter pluggable database pdb3 close immediate;

Pluggable database altered.

Elapsed: 00:00:00.24
cdb01$SYS> alter pluggable database pdb3 open restricted;


Pluggable database altered.

Elapsed: 00:00:00.94
cdb01$SYS> cdb01$SYS> select name, restricted from v$pdbs;

NAME                           RES
------------------------------ ---
PDB$SEED                       NO
PDB01                          NO
PDB3                           YES
```

PDB를 rename한다. PDB를 rename하기 위해 해당 PDB에 접속한다.

```
cdb01$SYS> alter pluggable database pdb3 rename global_name to pdb3_bis;
alter pluggable database pdb3 rename global_name to pdb3_bis
                                                    *
ERROR at line 1:
ORA-65046: operation not allowed from outside a pluggable database


Elapsed: 00:00:00.00
cdb01$SYS> connect sys/welcome1@localhost:1521/pdb3 as sysdba
Connected.
localhost:1521/pdb3$SYS> alter pluggable database pdb3 rename global_name to pdb3_bis;


Pluggable database altered.

Elapsed: 00:00:01.83
```

PDB를 close/open하면 PDB는 rename되어 있다.

```
localhost:1521/pdb3$SYS> localhost:1521/pdb3$SYS> alter pluggable database close immediate;


Pluggable database altered.

Elapsed: 00:00:00.22
localhost:1521/pdb3$SYS> localhost:1521/pdb3$SYS> alter pluggable database open;

Pluggable database altered.

Elapsed: 00:00:00.94
localhost:1521/pdb3$SYS> select name, open_mode from v$pdbs;

NAME                           OPEN_MODE
------------------------------ ----------
PDB3_BIS                       READ WRITE
```

Manage the storage in a CDB and its PDBs

CDB는 각 container마다 datafile과 그에 해당하는 data를 가지고 있다. (tempfile은 CDB 및 PDB 모두 구성이 가능하다.)

- Root의 tablespace, datafile, tempfile을 list-up

```
cdb01$SYS> select file_name, con_id from cdb_data_files where con_id=1;
```

```
FILE_NAME                          CON_ID
------------------------------ ----------
/u01/app/oracle/oradata/cdb01/              1
system01.dbf

/u01/app/oracle/oradata/cdb01/              1
sysaux01.dbf

/u01/app/oracle/oradata/cdb01/              1
undotbs01.dbf

/u01/app/oracle/oradata/cdb01/              1
users01.dbf

/u01/app/oracle/product/12.1.0              1
/db_1/dbs/com_tbs.dbf


Elapsed: 00:00:00.01
cdb01$SYS> select tablespace_name, con_id from cdb_tablespaces where con_id=1;


TABLESPACE_NAME                 CON_ID
------------------------------ ----------
SYSTEM                                1
SYSAUX                                1
UNDOTBS1                              1
TEMP                                  1
USERS                                 1
COM_TBS                               1

6 rows selected.

Elapsed: 00:00:00.01
cdb01$SYS> cdb01$SYS> select file_name, con_id from cdb_data_files where con_id=1;


FILE_NAME                          CON_ID
------------------------------ ----------
/u01/app/oracle/oradata/cdb01/              1
system01.dbf

/u01/app/oracle/oradata/cdb01/              1
sysaux01.dbf

/u01/app/oracle/oradata/cdb01/              1
undotbs01.dbf

/u01/app/oracle/oradata/cdb01/              1
users01.dbf

/u01/app/oracle/product/12.1.0              1
/db_1/dbs/com_tbs.dbf


Elapsed: 00:00:00.00
cdb01$SYS> cdb01$SYS> select file_name, con_id from cdb_temp_files where con_id=1;



FILE_NAME                          CON_ID
------------------------------ ----------
/u01/app/oracle/oradata/cdb01/              1
temp01.dbf


Elapsed: 00:00:00.01
```

- ROOT에 permanent tablespace를 생성

```
cdb01$SYS> create tablespace cdata datafile '/u01/app/oracle/oradata/cdb1/cdata01.dbf' SIZE 10M;

Tablespace created.

Elapsed: 00:00:00.15
```

```
cdb01$SYS> select tablespace_name, con_id from cdb_tablespaces order by con_id;


TABLESPACE_NAME                  CON_ID
------------------------------ ----------
SYSTEM                                1
CDATA                                 1
COM_TBS                               1
SYSAUX                                1
TEMP                                  1
UNDOTBS1                              1
USERS                                 1
SYSTEM                                2
TEMP                                  2
SYSAUX                                2
SYSTEM                                3
EXAMPLE                               3
USERS                                 3
TEMP                                  3
SYSAUX                                3
SYSTEM                                4
SYSAUX                                4
TEMP                                  4

18 rows selected.

Elapsed: 00:00:00.04
cdb01$SYS> select file_name, con_id from cdb_data_files order by con_id;

FILE_NAME                                                                  CON_ID
-------------------------------------------------------------------------- ----------
/u01/app/oracle/oradata/cdb01/system01.dbf                                      1
/u01/app/oracle/oradata/cdb1/cdata01.dbf                                        1
/u01/app/oracle/oradata/cdb01/sysaux01.dbf                                      1
/u01/app/oracle/oradata/cdb01/users01.dbf                                       1
/u01/app/oracle/oradata/cdb01/undotbs01.dbf                                     1
/u01/app/oracle/product/12.1.0/db_1/dbs/com_tbs.dbf                             1
/u01/app/oracle/oradata/cdb01/pdbseed/sysaux01.dbf                              2
/u01/app/oracle/oradata/cdb01/pdbseed/system01.dbf                              2
/u01/app/oracle/oradata/cdb01/pdb01/system01.dbf                                3
/u01/app/oracle/oradata/cdb01/pdb01/example01.dbf                               3
/u01/app/oracle/oradata/cdb01/pdb01/SAMPLE_SCHEMA_users01.dbf                   3
/u01/app/oracle/oradata/cdb01/pdb01/sysaux01.dbf                                3
/u01/app/oracle/oradata/cdb1/pdb3/sysaux01.dbf                                  4
/u01/app/oracle/oradata/cdb1/pdb3/system01.dbf                                  4

14 rows selected.

Elapsed: 00:00:00.01
```

- Root에 temporary tablespace 생성

```
cdb01$SYS> create temporary tablespace temp_root tempfile '/u01/app/oracle/oradata/cdb1/temproot01.dbf'
SIZE 10M;



Tablespace created.

Elapsed: 00:00:00.05
cdb01$SYS> cdb01$SYS> cdb01$SYS> select tablespace_name, con_id from cdb_tablespaces where
contents='TEMPORARY' and con_id=1;


TABLESPACE_NAME                  CON_ID
------------------------------ ----------
TEMP                                  1
TEMP_ROOT                             1

Elapsed: 00:00:00.02
cdb01$SYS> cdb01$SYS> select file_name, con_id from cdb_temp_files where con_id=1;

FILE_NAME                                                                  CON_ID
-------------------------------------------------------------------------- ----------
/u01/app/oracle/oradata/cdb01/temp01.dbf                                        1
```

```
/u01/app/oracle/oradata/cdb1/temproot01.dbf                                1

Elapsed: 00:00:00.01
```

- PDB에 tablespace 생성

```
cdb01$SYS> connect system/welcome1@localhost:1521/pdb3_bis
Connected.
localhost:1521/pdb3_bis$SYSTEM> create tablespace ldata datafile
'/u01/app/oracle/oradata/cdb1/pdb3/ldata01.dbf' SIZE 10M;


Tablespace created.

Elapsed: 00:00:00.36
localhost:1521/pdb3_bis$SYSTEM> select tablespace_name, con_id from cdb_tablespaces order by con_id;


TABLESPACE_NAME                 CON_ID
------------------------------ ----------
SYSTEM                              4
SYSAUX                              4
TEMP                                4
LDATA                               4

Elapsed: 00:00:00.01
localhost:1521/pdb3_bis$SYSTEM> select file_name, con_id from cdb_data_files order by con_id;


FILE_NAME                                                        CON_ID
---------------------------------------------------------------- ----------
/u01/app/oracle/oradata/cdb1/pdb3/system01.dbf                            4
/u01/app/oracle/oradata/cdb1/pdb3/sysaux01.dbf                            4
/u01/app/oracle/oradata/cdb1/pdb3/ldata01.dbf                             4

Elapsed: 00:00:00.05
localhost:1521/pdb3_bis$SYSTEM> select file_name from dba_data_files;

FILE_NAME
--------------------------------------------------------------------------
/u01/app/oracle/oradata/cdb1/pdb3/system01.dbf
/u01/app/oracle/oradata/cdb1/pdb3/sysaux01.dbf
/u01/app/oracle/oradata/cdb1/pdb3/ldata01.dbf

Elapsed: 00:00:00.03
```

PDB에 접속해 CDB_xxx나 DBA_xxx를 조회하면 같은 정보가 조회된다.

- PDB내에 temporary tablespace 생성

```
localhost:1521/pdb3_bis$SYSTEM> create temporary tablespace temp_pdb3 tempfile
'/u01/app/oracle/oradata/cdb1/pdb3/temppdb301.dbf' SIZE 10M;


Tablespace created.

Elapsed: 00:00:00.05
localhost:1521/pdb3_bis$SYSTEM> select tablespace_name, con_id from cdb_tablespaces where
contents='TEMPORARY';


TABLESPACE_NAME                 CON_ID
------------------------------ ----------
TEMP                                4
TEMP_PDB3                           4

Elapsed: 00:00:00.01
localhost:1521/pdb3_bis$SYSTEM> select file_name from dba_temp_files;



FILE_NAME
--------------------------------------------------------------------------
/u01/app/oracle/oradata/cdb1/pdb3/pdbseed temp01.dbf
```

```
/u01/app/oracle/oradata/cdb1/pdb3/temppdb301.dbf

Elapsed: 00:00:00.01
```

Manage security in PDBs

Manage the Common and local users

CDB내에 위치한 각각의 Container는 common user와 local user를 가진다. common이든 local이든 user가 연결되어 있는 특정 container내에서 권한 부여를 할 수 있다.

Common user는 root에서 생성되고 자동적으로 각각의 PDB로 replicate된다.(seed PDB 제외)

Common user는 어떠한 PDB와도 특정 권한이 있으면 연결가능하고 database name은 항상 c##을 붙여 common user 임을 명시한다.

Local user는 현재 사용되고 있는 Container의 PDB 내에서 생성되고 local user가 생성된 PDB에만 연결 가능하다. 따라서 다른 스키마를 가진 PDB는 연결할 수 없다.

- Root container로 접속 후 common user 생성

```
cdb01$SYS> select username, common, con_id from cdb_users where username like 'C##%';

USERNAME             COM     CON_ID
-------------------- --- ----------
C##1                 YES          1
C##COMM_USER         YES          1
C##1                 YES          4
C##COMM_USER         YES          4
C##1                 YES          3
C##COMM_USER         YES          3

6 rows selected.

Elapsed: 00:00:00.01
```

Local user는 Seed PDB에 생성되지 않는다.

- PDB내의 common user로 로그인

```
cdb01$SYS> connect c##1/oracle@localhost:1521/pdb01
ERROR:
ORA-01045: user C##1 lacks CREATE SESSION privilege; logon denied


Warning: You are no longer connected to ORACLE.
cdb01$SYS> connect c##1/oracle@localhost:1521/pdb3_bis

ERROR:
ORA-01045: user C##1 lacks CREATE SESSION privilege; logon denied
```

User는 각각의 PDB를 인식할 수는 있지만 CREATE SESSION권한이 아직 주어지지 않았기 때문에 연결할 수가 없다.

- Local user를 생성하기 위해 PDB에 DBA권한으로 접속

```
cdb01$SYS> connect system/welcome1@localhost:1521/pdb3_bis
Connected.
localhost:1521/pdb3_bis$SYSTEM> create user hr identified by oracle;



User created.

Elapsed: 00:00:00.12
localhost:1521/pdb3_bis$SYSTEM> select username, common, con_id  from cdb_users where username ='HR';


USERNAME             COM     CON_ID
-------------------- --- ----------
HR                   NO           4

Elapsed: 00:00:00.03
```

각각의 PDB내의 local user인 HR로 접속한다.

```
cdb01$SYS> connect hr/oracle@localhost:1521/pdb3_bis;
ERROR:
ORA-01045: user HR lacks CREATE SESSION privilege; logon denied


cdb01$SYS>
```

Manage the common and local roles

CDB의 Container에는 common role과 local role이 존재한다.

root에서 common role가 생성되자마자 Seed PDB를 제외한 모든 PDB들에 복제가 된다. Common role의 이름은 반드시 c##으로 시작한다.

Local role은 PDB내에서만 생성될 수 있기 때문에 local role이 만들어진 PDB내에서만 role이 주어질 수 있다. 즉, CDB내의 다른 PDB와는 독립적이다.

Create a common role

```
cdb01$SYS> create role c##r1 container=all;


Role created.

Elapsed: 00:00:00.09
cdb01$SYS> cdb01$SYS> select role, common, con_id from cdb_roles where role='C##R1';


ROLE                 COM     CON_ID
-------------------- --- ----------
C##R1                YES          1
C##R1                YES          4
C##R1                YES          3

Elapsed: 00:00:00.01
```

Seed PDB를 제외한 모든 PDB에 role을 생성한다.

PDB내에 local role를 생성 후 PDB에 로그인하여 common role을 생성하려고 시도하지만 실패한다.

```
cdb01$SYS> connect system/welcome1@localhost:1521/pdb3_bis;
Connected.
localhost:1521/pdb3_bis$SYSTEM> create role hr_manager;

Role created.

Elapsed: 00:00:00.01
localhost:1521/pdb3_bis$SYSTEM> select role, common, con_id from cdb_roles where role='HR_MANAGER';


ROLE                 COM     CON_ID
-------------------- --- ----------
HR_MANAGER           NO           4

Elapsed: 00:00:00.02
localhost:1521/pdb3_bis$SYSTEM> create role c##r2 container=all;
create role c##r2 container=all
*
ERROR at line 1:
ORA-65050: Common DDLs only allowed in CDB$ROOT


Elapsed: 00:00:00.01
```

Manage the common and local privileges

Common 혹은 local privilege는 user와 role에 권한을 부여한다. 단지 CONTAINER=ALL(CURRENT) 옵션을 추가함으로써 common privilege나 local privilege 권한을 부여할 수 있다.

- Common privilege는 common user혹은 role을 부여하게되면 자동적으로 PDB에 생성된다.(SeedPDB제외)
- Local privilege는 특정 PDB에서 user이나 role을 부여함으로써 생성된다.

Grant the CREATE SESSION as a common privilege to a common user

```
cdb01$SYS> cdb01$SYS> grant create session to c##1 container=all;


Grant succeeded.

Elapsed: 00:00:00.05
cdb01$SYS> select grantee, privilege, common, con_id from cdb_sys_privs
where privilege='CREATE SESSION' and grantee='C##1';

  2
GRANTEE             PRIVILEGE                                COM     CON_ID
------------------- ---------------------------------------- --- ----------
C##1                CREATE SESSION                           YES         1
C##1                CREATE SESSION                           YES         3
C##1                CREATE SESSION                           YES         4

Elapsed: 00:00:00.01
cdb01$SYS> connect c##1/oracle@localhost:1521/pdb01
Connected.
localhost:1521/pdb01$C##1> select * from session_privs;


PRIVILEGE
----------------------------------------
CREATE SESSION

Elapsed: 00:00:00.02
localhost:1521/pdb01$C##1> connect c##1/oracle@localhost:1521/pdb3_bis

Connected.
localhost:1521/pdb3_bis$C##1> select * from session_privs;


PRIVILEGE
----------------------------------------
CREATE SESSION

Elapsed: 00:00:00.01
```

local user에서 Common privilege로 privilege를 부여하는 것은 허용되지 않는다. 하지만 local에서 local user로 privilege를 줄 수 있다.

- 해당 PDB의 local user는 Create Session이라는 local privilege를 부여한다.

```
cdb01$SYS> connect system/welcome1@localhost:1521/pdb3_bis;
Connected.
localhost:1521/pdb3_bis$SYSTEM> grant create session to hr container=all;
grant create session to hr container=all
*
ERROR at line 1:
ORA-65030: one may not grant a Common Privilege to a Local User or Role


Elapsed: 00:00:00.00
localhost:1521/pdb3_bis$SYSTEM> grant create session to hr;


Grant succeeded.

Elapsed: 00:00:00.04
localhost:1521/pdb3_bis$SYSTEM> select grantee, privilege, common, con_id from cdb_sys_privs
where privilege='CREATE SESSION' and grantee='HR';

  2
GRANTEE             PRIVILEGE                                COM     CON_ID
------------------- ---------------------------------------- --- ----------
```

```
HR                    CREATE SESSION                          NO          4

Elapsed: 00:00:00.02
```

```
localhost:1521/pdb3_bis$SYSTEM> connect hr/oracle@localhost:1521/pdb3_bis

Connected.
localhost:1521/pdb3_bis$HR> localhost:1521/pdb3_bis$HR> select * from session_privs;


PRIVILEGE
----------------------------------------
CREATE SESSION

Elapsed: 00:00:00.00
```

Drop PDBs

PDB를 drop하는 방법에는 datafile의 삭제유무에 따라서 2가지로 나뉜다. PDB를 plug-in/out할 때 datafile을 reuse하거나 PDB를 unplug하여 다른 CDB로 plug-in할 때 datafile을 삭제하지 않고 유지한다.

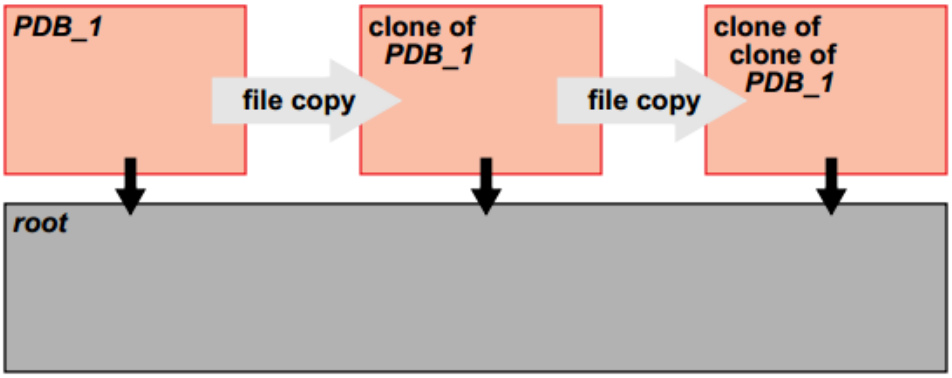
Close and drop a PDB

```
localhost:1521/pdb3_bis$HR> localhost:1521/pdb3_bis$HR> connect / as sysdba

Connected.
cdb01$SYS> alter pluggable database all close immediate;


Pluggable database altered.

Elapsed: 00:00:01.32
cdb01$SYS> select name, open_mode from v$pdbs;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB01                          MOUNTED
PDB3_BIS                       MOUNTED

Elapsed: 00:00:00.00
cdb01$SYS> drop pluggable database pdb3_bis including datafiles;


Pluggable database dropped.

Elapsed: 00:00:00.24
cdb01$SYS> cdb01$SYS> cdb01$SYS> select name from v$pdbs;

NAME
------------------------------
PDB$SEED
PDB01
```

III. Clone Pluggable Database

Clone Pluggable Database Overview

보통 시스템 운영중에 복제가 필요하다면 database를 unplug한 후 OS copy를 이용해 모든 files을 카피하고 원본 file을 다시 plug-in해 복제된 file을 rename하는 방법으로 clone PDB를 생성했다. 하지만 12c의 경우 source PDB를 unplug하지 않고 foreground process가 file을 복사하고 복사된 파일을 plug-in하여 복제한다.



<그림1-6 Clone Pluggable Database Diagram>

실제 foreground procoess는 복제된 PDB에 새로운 global unique ID를 할당하고 CDB와 sync하는 방법으로 진행된다.

Multitenant Architecture는 CDB와 PDB사이에 Oracle Net 연결이 있는한 다른 CDB(Remote Cloning) 혹은 같은 CDB내(Local cloning)에 다수의 PDB들을 복제할 수 있다.

Cloning Types

Create New PDB from PDB$SEED

PDB$SEED를 통해 거의 동시에 new PDB를 복제할 수 있는 장점이 있다. PDB$SEED에서 New PDB를 만드는 방법에는 OMF사용여부에 따라 2가지 방법으로 나뉜다. PDB생성 후 Open하게되면 Seed$PDB와 NewPDB는 자동적으로 Syn를 맞추게되고 error발생 시 메시지는 alert.log에 기록되고 PDB_PLUG_IN_VIOLATIONS view로 조회하여 확인 할 수 있다.

① With OMF(Oracle Managed File)

SYS유저로 ROOT Container에 접속
Target directory의 init.ora에 instance parameter지정
DB_CREATE_FILE_DEST='xxx/xxx/xxx/새로운 PDB이름'

```
SQL> CREATE PLUGGABLE DATABASE pdb1
2 ADMIN USER pdb1_admin IDENTIFIED BY p1 3 ROLES=(CONNECT);
```

② Without OMF

SYS유저로 ROOT Container에 접속
SEED$PDB와New PDB모두의 init.ora에 instance parameter지정
PDB_FILE_NAME_CONVERT='seed$pdb_dir','new_pdb_dir')

```
SQL> CREATE PLUGGABLE DATABASE pdb1
2 ADMIN USER pdb1_admin IDENTIFIED BY p1 3 ROLES=(CONNECT);
```

Plug a Non-CDB into CDB Using DBMS_PDB

Non-CDB를 CDB내의 PDB로 Plun-In하는 방법에는 크게 3가지가 있다.

① TTS, TDB, full export/import

User가 생성한 Tablespace 중 맨 마지막의 Object를 이용해 새로운 PDB를 생성한다.

② XML FILE DEFINITION

Non-CDB가 12c database일 경우 DBMS_PDB 프로시저를 이용하여 non-CDB의 XML file을 만들고 이를 이용하여 New PDB를 생성한다.

- Steps
  - Non-CDB가 transactionally consistent 상태인지 확인 후 Read-Only mode로 전환
  - Non-CDB내에서 DBMS_PDB.DESCRIBE 프로시저를 실행하여 NEW PDB를 위한 XML FILE을 생성한다.
  - Non-CDB가 Target CDB에 Plug할 수 있는지 check한다 (DBMS_PDB.CHECK_PLUG_COMPATIBILITY procedure 실행)
  - Non-CDB를 Plug-In하기 위해 Target CDB로 접속한다.
  - CREATE PLUGGABLE… USING 'XMLfile'구문을 이용해 PDB를 구성한다. 이때 새롭게 생성될 DATAFILE 이름과 위치정보를 XML FILE에서 읽어온다.
  - ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql 스크립트를 돌려준다. 스크립트를 돌려줌으로써 불필요한 메타데이타를 삭제한다.(반드시 NEW PDB를 OPEN하기 전에 돌려준다)
  - NEW PDB OPEN
  -

③ REPLICATE

- ORACLE PREFERENCE

3 Methods	Easier Method	비고
DBMS_PDB	Easiest Method	
DBMS_PDB 사용불가	Full export/import	Downtime 발생
Full export / import vs Replication(OGG)	Full export/import	
Whole Non-CDB into a CDB	TDB	
Part of Non-CDB into a CDB	TTS	

Cloning PDBs

- Steps
  - init.ora내에 Instance Parameter를 추가한다. (DB_CREATE_FILE_DEST= 'PDB3dir' (OMF) or PDB_FILE_NAME_CONVERT= 'PDB1dir', 'PDB3dir' (non OMF))
  - CREATE PLUGGABLE DATABASE 권한을 가진 common user로 Root Container에 접속한다.
  - Source PDB를 Read-Only mode로 전환
  - CREATE PLUGGABLE DATABASE구문으로 NEW PDB를 생성한다.
  - NEW PDB OPEN

Plug Unplugged PDB in to CDB

PDB를 CDB1에서 Unplugging하여 CDB2에 New PDB를 plugging하거나 PDB가 더 이상 필요가 없을 경우 방법이다.

- Steps
  - CDB1의 root container로 접속하여 V$PDBs view를 조회하여 PDB가 READ ONLY인지 확인한다.
  - ALTER PLUGGABLE DATABASE … UNPLUG 구문을 이용해 DATABASE 및 XML FILE을 UNPLUG한다.

```
SQL> ALTER PLUGGABLE DATABASE
2 pdb1UNPLUGINTO'xmlfile1.xml';
```

  - 만약 다른 CDB로 복제를 한다면, DATAFILE을 해당 CDB로 복사하고 PLUGING한다. 하지만 더 이상 사용하지 않을 경우 PDB를 반드시 DROP해야 한다.
  - 다른 CDB로 PDB를 PLUGGING하기 위해서는 ROOT CONTAINER에 COMMON USER로 접속하여 ALTER PLUGGABLE DATABASE pdb1 USING ‘xmlfile1’구문을 입력한다.

```
SQL> CREATE PLUGGABLE DATABASE 2 pdb1 USING 'xmlfile1.xml' 3
NOCOPY;
```

Prepare the Source PDB to Clone

Set the source PDB in READ ONLY mode

복제를 위해 PDB를 close 한 후 READ ONLY mode로 open한다.

```
[oracle@xteam ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Wed Jul 10 20:02:52 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

cdb01$SYS> alter pluggable database pdb01 close immediate;

Pluggable database altered.

Elapsed: 00:00:00.49
cdb01$SYS> alter pluggable database pdb01 open read only;

Pluggable database altered.

Elapsed: 00:00:00.60
```

Create a directory for the new clone PDB

복제된 pdb의 datafile이 저장될 공간을 찾아 디렉토리를 생성한다.

```
[oracle@xteam ~]$ su -
Password:
 [root@xteam ~]# mkdir -p /stage
[root@xteam ~]# chown oracle:oinstall /stage
[root@xteam ~]# su - oracle
```

Configure OMF to the directory of the clone PDB

Sysdba로 접속하여 다음의 파라미터값을 셋팅한다.

```
[oracle@xteam stage]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Wed Jul 10 20:13:23 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

cdb01$SYS> alter system set db_create_file_dest='/stage/pdb1_clone';

System altered.

Elapsed: 00:00:00.01
```

Clone the PDB Within the CDB

```
cdb01$SYS> create pluggable database pdb1_clone from pdb01;

Pluggable database created.

Elapsed: 00:00:43.83
cdb01$SYS> alter pluggable database pdb1_clone open;

Pluggable database altered.

Elapsed: 00:00:06.91
cdb01$SYS> connect system/welcome1@localhost:1521/pdb1_clone;
Connected.
localhost:1521/pdb1_clone$SYSTEM> show con_name;

CON_NAME
------------------------------
PDB1_CLONE
```

Set the Source PDB Back to Open Mode

```
cdb01$SYS> alter session set container=cdb$root;


Session altered.

Elapsed: 00:00:00.00
cdb01$SYS> alter pluggable database pdb01 close immediate;

Pluggable database altered.

Elapsed: 00:00:00.50
cdb01$SYS> alter pluggable database pdb01 open;

Pluggable database altered.

Elapsed: 00:00:00.87
```

Clean Up the Environment

```
cdb01$SYS> alter pluggable database pdb1_clone close immediate;
```

```
Pluggable database altered.

Elapsed: 00:00:00.65
cdb01$SYS> cdb01$SYS>
cdb01$SYS> drop pluggable database pdb1_clone including datafiles;


Pluggable database dropped.

Elapsed: 00:00:00.42
```