Alkmene Verlags- und
Mediengesellschaft mbh
Wittelsbacherallee 107
60385 Frankfurt am Main

IT-Tage 2015

Schwerpunkt: Datenbanken
14.12. - 18.12.2015
Frankfurt am Main

Marek Adar:

Oracle 12c Multitenant


Luxaviation
Germany GmbH

Multitenant

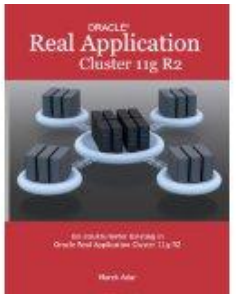
luxaviation

- Wer bin ich?
  - Marek Adar/ Bj. 1970 / 4 Kinder 2, 5, 15, 20
  - Luxaviation Group / IT-Leitung Luxaviation Germany
  - Gruppenweit zuständig für Oracle, Monitoring, Entwicklung
  - Arbeite mit Oracle seit 2000
  - SQL, PL/SQL, Administration, B&R, RAC, Data Guard, Tuning

Multitenant

luxaviation

- Publikationen
  - Ein strukturierter Einstieg in die Or
    Datenbankadministration.
  - Ein strukturierter Einstieg in Oracle
    Application Cluster 11g R2.
  - Ein strukturierter Einstieg in die Or
    und
    PL/SQL-Entwicklung.



- Publikationen
  - Das große Oracle Datenbank-Einsteiger

  - Der Oracle DBA: Handbuch für die Administration der Oracle Database 11g R2.

  - Der Oracle DBA: Handbuch für die Administration der Oracle Datenbank 12c

Multitenant

luxaviation

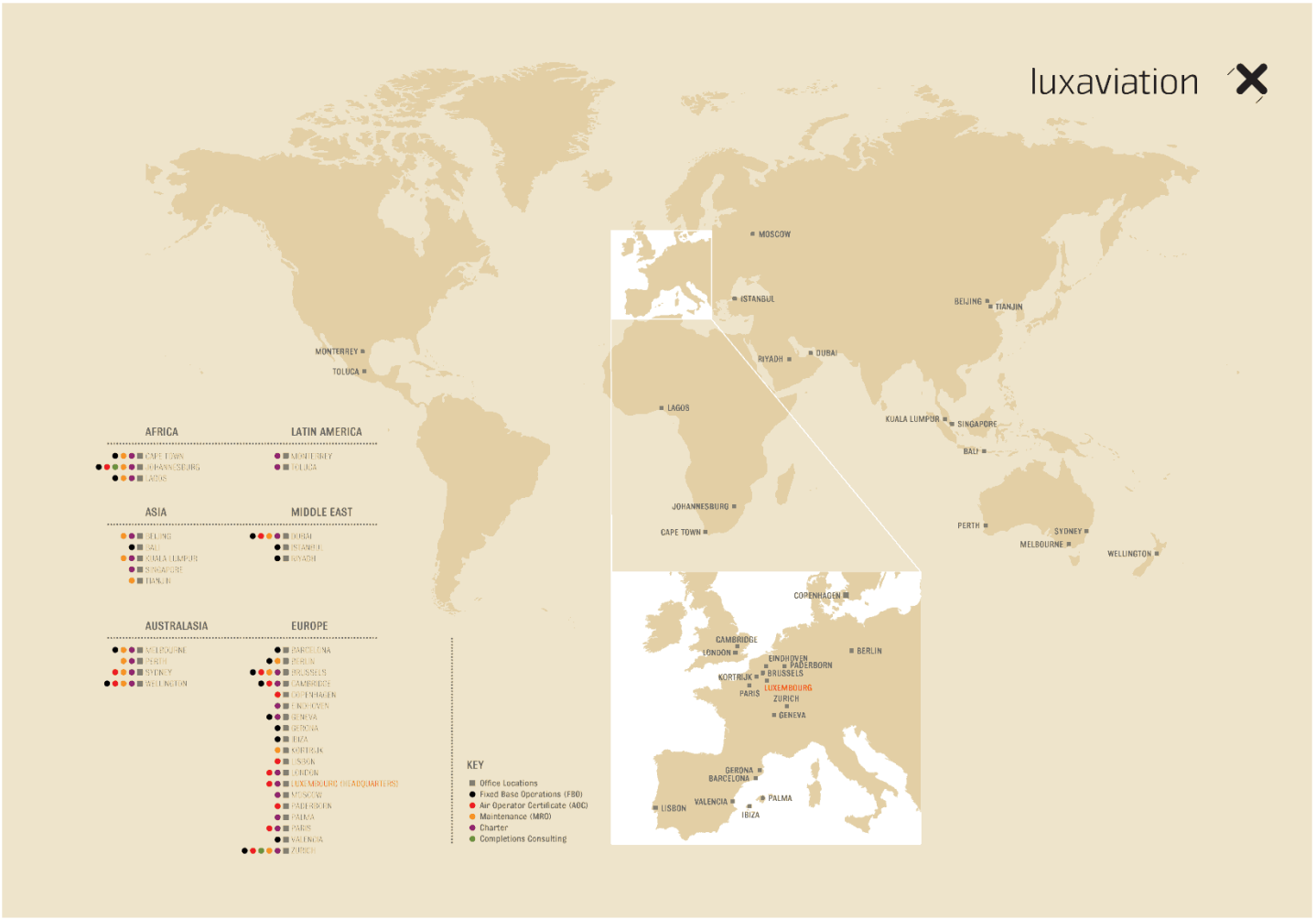
- Luxaviation Group
  - Luxaviation Luxemburg / Germany / Asia
  - LEA
  - UNIJET
  - Abelag
  - ExecuJet
  - MasterJet

Multitenant

luxaviation

- Luxaviation Group
  - Charter-Gesellschaft seit 2005
  - Ca. 1000 Mitarbeiter
  - Ca. 100 Jets
  - Aircraft Management
  - Aircraft Charter
  - Aircraft reselling
  - Operation-Handling

Luxaviation Group



Luxaviation Group

Cessna Citation CJ1

Beechcraft Premier I

Cessna Citation CJ2

Cessna Citation CJ3

Cessna Citation XLS

Hawker 800XP

Bombardier Challenger 300

Dassault Falcon 50

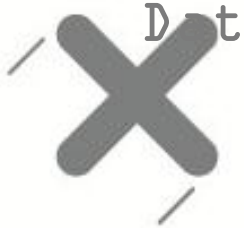
Dassault Falcon 900

Multitenant

luxaviation

Grundlagen

- Multitenant gibt die Möglichkeit eine Oracle Datenbank als eine Container-Datenbank agieren zu lassen.
- Diese Container-Datenbank kann so eine oder mehrere voneinander isolierte Datenbanken (PDB) verwalten.
- Jede PDB besitzt ihre eigenen Schemata sowie deren Objekte und Benutzern.
- Jede PDB sieht für die verbundene Anwendung wie eine eigen Datenbank aus.



Grundlagen

- Es ist effizienter mehrere PDBs in einer Multitenant-Umgebung zu betreiben, als einzelne Datenbanken auf der gleichen Maschine.
- Eine Multitenant-Umgebung hat nur einen Einfluss auf die administrativen Tätigkeiten des Administrators.
- Aus Entwicklersicht bleibt diese Umgebung transparent und hat auf ihn keinen Einfluss.

Grundlagen

- Multitenant muss separat lizenziert werden und ist nur in der Enterpriseversion verfügbar.
- Ohne Lizenz darf nur eine PDB aufgesetzt werden.
- Bei mehreren PDBs muss eine Lizenz für Multitenant erworben werden.

Grundlagen

Oracle Multitenant

Oracle Multitenant is an option to the Oracle Database 12c Enterprise Edition. Its new, modern architecture simplifies the process of consolidating databases onto the cloud without any required changes to existing applications. It provides efficient database provisioning, patching and upgrading, and the ability to manage many databases as one, all while increasing server utilization and scalability. It complements other Oracle Database Options, such as Oracle Real Application Clusters and Active Data Guard.

> Learn More
> Datasheet
> White Paper

€15,194.00 / Processor

Metrik: Processor ▼ Laufzeit: Unbefristete ▼ Menge: 1

In den Einkaufskorb

✔ Software Update License and Support (ersten Jahr) €3,342.57

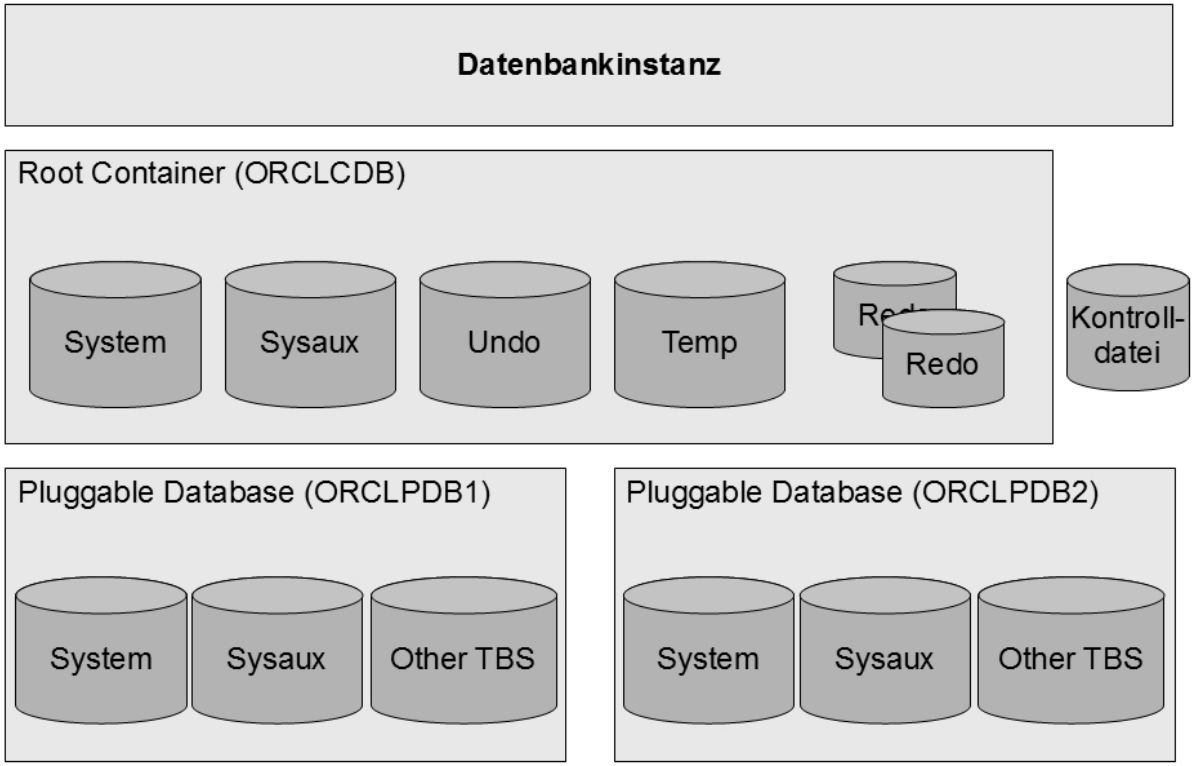
Architektur

- Die Containerdatenbank ist ähnlich einer traditionellen Datenbank aufgebaut.
- Sie besitzt alle für die Datenbank wichtigen Tablespaces sowie Redolog- und Kontrolldateien.
- Die Pluggable Datenbanken sind an die Containerdatenbank angeschlossen und alle teilen sich die gleiche Datenbankinstanz sowie die Redo-Log-Dateien und die Kontrolldateien.

Architektur

- Da alle PDBs die gleichen Redo-Log-Dateien verwenden ist dafür Sorge zu tragen, dass die Redo-Log-Dateien ausreichend groß sind, um die zu protokollierenden Daten jeder PDB aufnehmen zu können und eine adäquate Checkpoint-Rate vorhanden ist.
- Standardmäßig besitzt die CDB einen temporären Tablespace, welcher für die CDB aber auch für alle PDBs verwendet wird. (Allerdings kann für jede PDB auch ein eigener Standard temporärer Tablespace erstellt werden).

Architektur



- Container in einer Containerdatenbank
  - Ein Container ist eine Ansammlung von Schemata, Objekten und Strukturen in einer CDB, die logisch zu einer Anwendung als eine separate Datenbank angezeigt wird.
  - Die Root und alle Pluggable Datenbanken gelten als ein Container.
  - Der Stammcontainer, auch die Root genannt, ist eine Sammlung von Schemata, Schemaobjekten und Nichtschemaobjekten, die zu allen PDBs gehören.

- Container in einer Containerdatenbank
  - Jeder CDB hat nur einen Root-Container, der die Systemmetadaten für die Verwaltung von PDBs speichert.
  - Alle PDBs gehören zum Stamm.
  - Der Name des Stammes ist CDB$ROOT. Der Root-Container speichert so zum Beispiel die von Oracle bereitgestellten Packages oder Metadaten von allgemeinen Benutzern, die in allen PDBs verwendet werden können.
  - Objekte oder Benutzer, die nur einer PDB selber zugehörig sind, werden im Datadictionary der PDB selber gespeichert.

Erstellung einer MT-Datenbank

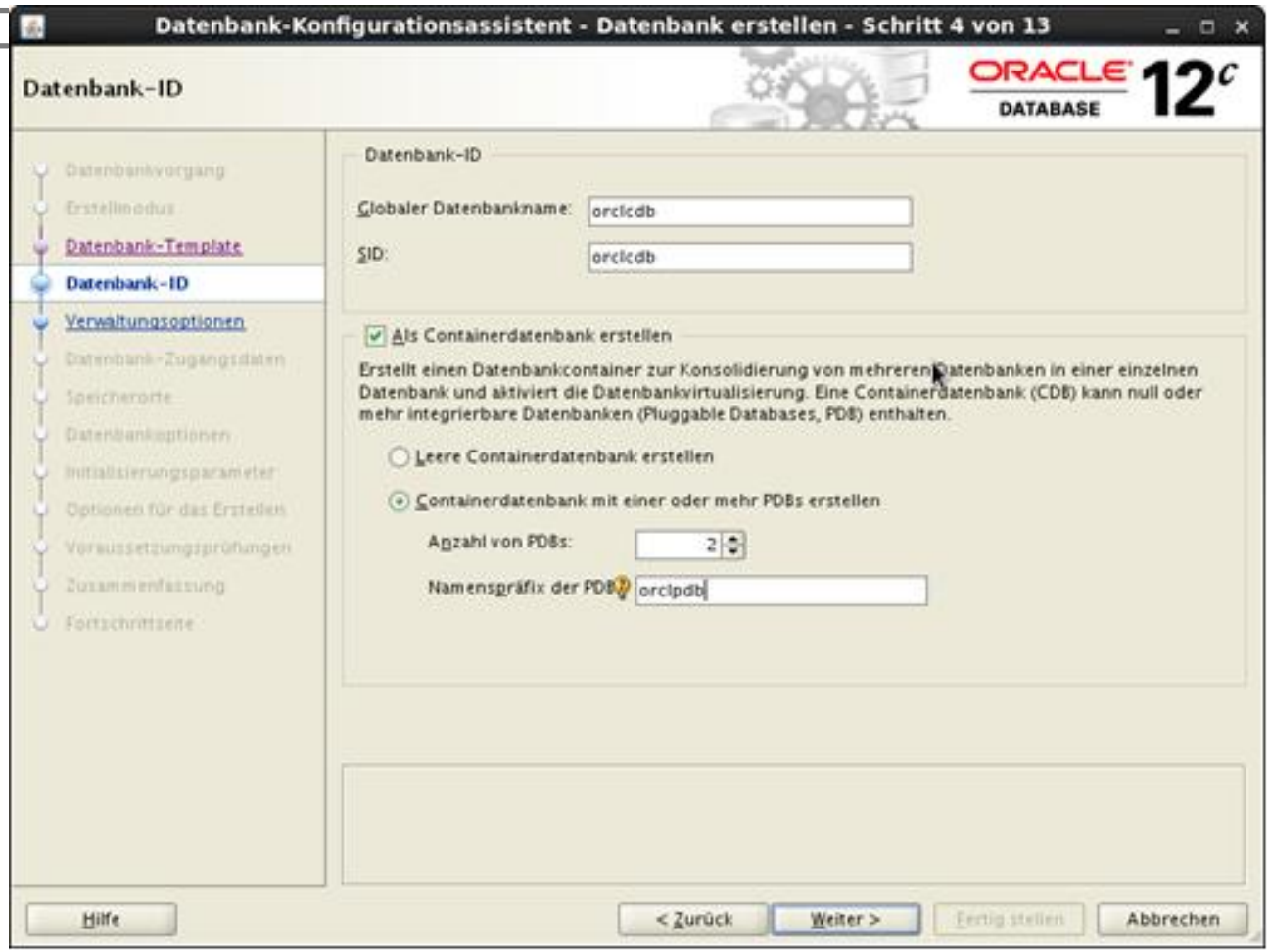
- Erstellung einer Multitenant Datenbank mit dem DBCA
  - Für die Erstellung einer Multitenant Datenbank besitzt der DBCA einen speziellen Dialog, der zum einen nach dem globalen Datenbanknamen und der SID der Containerdatenbank fragt, sofern der Haken „Als Containerdatenbank erstellen“ ausgewählt wurde.
  - Des Weiteren wird nach der Erstellung von PDBs gefragt, oder ob eine leere Containerdatenbank, also erst einmal ohne PDBs, erzeugt werden soll.

Erstellung einer MT-Datenbank

- Erstellung einer Multitenant Datenbank mit dem DBCA
  - Wird die Option „Containerdatenbank mit einer oder mehr PDBs erstellen“ ausgewählt, kann die Anzahl der zu erstellenden PDBs, sowie deren Name angegeben werden.
  - Die einzelnen PDBs werden dann mit dem angegebenen Namen als Präfix durchnummeriert.

Erstellung einer MT-Datenbank mit dem DBCA

- Erstellung einer Multitenant Datenbank mit dem DBCA



Anmelden

luxaviation

- Anmelden an einer CDB und PDB
  - Das Anmelden an einer CDB erfolgt auf gleiche Art und Weise wie an einer traditionellen Datenbank.
  - Für die Anmeldung wird entweder bei lokaler Anmeldung am Betriebssystem die Umgebungsvariable ORACLE_SID auf die entsprechende Datenbank gesetzt oder bei einer Remoteanmeldung die TNS-Architektur verwendet.
  - Für die direkte Anmeldung an einer PDB muss TNS verwendet werden.
  - Mit Hilfe des SQLPlus-Befehls SHOW CON_NAME kann der aktuelle Container ermittelt werden.

Anmelden an einer CDB und PDB

Anmelden an der Container Datenbank:

```
[oracle@ORASRV ~]$ export ORACLE_SID=orclcdb
[oracle@ORASRV ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Tue Apr 28 15:53:26 2015

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


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

SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

Anmelden an einer PDB:

```
[oracle@ORASRV ~]$ sqlplus sys/oracle@orasrv:1521/orclpdb1 as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Tue Apr 28 15:49:04 2015

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


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

SQL> show con_name

CON_NAME
------------------------------
ORCLPDB1
SQL>
```

- Anmelden an einer CDB und PDB
  - Eine weitere Möglichkeit der Anmeldung an einer PDB ist die Anmeldung an der CDB und im zweiten Schritt wird mit Hilfe des Befehls
    - **ALTER SESSION SET CONTAINER=[Name]**
  - der Container gewechselt.

Anmelden an einer CDB und PDB

```
[oracle@ORASRV ~]$ export ORACLE_SID=orclcdb
[oracle@ORASRV ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Tue Apr 28 15:53:26 2015

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

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

SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL> ALTER SESSION SET CONTAINER=orclpdb1;

Session wurde geandert.

SQL> show con_name

CON_NAME
------------------------------
ORCLPDB1
```

Listener und Services

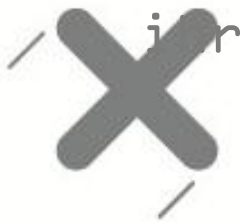
- Im Listener werden bei Erstellung einer PDB deren Servicenamen automatisch registriert, um eine Anmel

```
LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 12.1.0.1.0 - Production
Start Date                22-APR-2015 13:29:38
Uptime                    6 days 2 hr. 9 min. 30 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/product/12.1.0.2/db_1/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/ORASRV/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ORASRV)(PORT=1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=ORASRV)(PORT=5501))(Security=(my_wallet_directory=/
u01/app/oracle/admin/orcl/xdb_wallet))(Presentation=HTTP)(Session=RAW))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=ORASRV)(PORT=5500))(Security=(my_wallet_directory=/
u01/app/oracle/admin/orclcdb/xdb_wallet))(Presentation=HTTP)(Session=RAW))
Services Summary...
Service "orcl" has 2 instance(s).
  Instance "orcl", status UNKNOWN, has 1 handler(s) for this service...
  Instance "orcl", status READY, has 1 handler(s) for this service...
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this service...
Service "orclcdb" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "orclcdbXDB" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "orclpdb1" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "orclpdb2" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Datadictionary

luxaviation

- Informationen über Dictionary-Views abfragen
  - Oracle hat für die Verwaltung und das Auslesen von Informationen für Multitenant zusätzliche Dictionary-View hinzugefügt und hat vorhandene Views mit weiteren Spalten versehen.
  - So wurde neben den Dictionary-Views mit der Präfix DBA_, ALL_ und USER_, Views mit der Präfix CDB_ hinzugefügt, die alle Objekte in der CDB aber auch allen PDBs anzeigen.
  - In dem folgenden Beispiel wird die Struktur der View CDB_PROCEDURES wiedergegeben, die die zusätzliche Spalte CON_ID beinhaltet und mit ihr die Containerzugehörigkeit zeigt.



Informationen über Dictionary-Views abfragen

```
SQL> DESCRIBE CDB_PROCEDURES
 Name                                      Null?    Typ
 ----------------------------------------- -------- ----------------
 OWNER                                              VARCHAR2(128)
 OBJECT_NAME                                        VARCHAR2(128)
 PROCEDURE_NAME                                     VARCHAR2(128)
 OBJECT_ID                                          NUMBER
 SUBPROGRAM_ID                                      NUMBER
 OVERLOAD                                           VARCHAR2(40)
 OBJECT_TYPE                                        VARCHAR2(13)
 AGGREGATE                                          VARCHAR2(3)
 PIPELINED                                          VARCHAR2(3)
 IMPLTYPEOWNER                                      VARCHAR2(128)
 IMPLTYPENAME                                       VARCHAR2(128)
 PARALLEL                                           VARCHAR2(3)
 INTERFACE                                          VARCHAR2(3)
 DETERMINISTIC                                      VARCHAR2(3)
 AUTHID                                             VARCHAR2(12)
 ORIGIN_CON_ID                                      NUMBER
 CON_ID                                             NUMBER
```

- Informationen über Dictionary-Views abfragen
  - Über die View V$CONTAINERS kann entsprechend der Spalte CON_ID der dazu-gehörige Container sowie weitere Information ermittelt werden.

```
SQL> DESCRIBE V$CONTAINERS;
 Name                                      Null?    Typ
 ----------------------------------------- -------- ----------------
 CON_ID                                             NUMBER
 DBID                                               NUMBER
 CON_UID                                            NUMBER
 GUID                                               RAW(16)
 NAME                                               VARCHAR2(30)
 OPEN_MODE                                          VARCHAR2(10)
 RESTRICTED                                         VARCHAR2(3)
 OPEN_TIME                                          TIMESTAMP(3)
 CREATE_SCN                                         NUMBER
 TOTAL_SIZE                                         NUMBER
```

```
SQL> SELECT CON_ID, NAME FROM V$CONTAINERS;

    CON_ID NAME
---------- ------------------------------
         1 CDB$ROOT
         2 PDB$SEED
         3 ORCLPDB1
         4 ORCLPDB2
```

```
SQL> SELECT COUNT(*) AS ANZAHL, CON_ID
  2  FROM CDB_PROCEDURES
  3  GROUP BY CON_ID;

    ANZAHL     CON_ID
---------- ----------
     28143          1
     28151          4
     28151          3
     28151          2
```

Datadictionary

luxaviation

- Informationen über Dictionary-Views abfragen
  - Werden die Dictionary-Views mit den Präfixen DBA_, ALL_ oder USER_ ausgeführt, so liefern diese nur Objektinformationen der PDB zurück, zu der zurzeit eine Sitzung besteht.
  - Über die View V$PDBS können Informationen über die PDBs selber ausgelesen werden, also ohne Container

```
SQL> DESCRIBE V$PDBS
Name                                      Null?    Typ
----------------------------------------- -------- ------------
CON_ID                                             NUMBER
DBID                                               NUMBER
CON_UID                                            NUMBER
GUID                                               RAW(16)
NAME                                               VARCHAR2(30)
OPEN_MODE                                          VARCHAR2(10)
RESTRICTED                                         VARCHAR2(3)
OPEN_TIME                                          TIMESTAMP(3)
CREATE_SCN                                         NUMBER
TOTAL_SIZE                                         NUMBER
```

- Informationen über Dictionary-Views abfragen
  - Die Views CDB_PDBS und DBA_PDBS liefern ebenfalls Informationen über die vorhandenen PDBs zurück. Speziell ist hier die Spalte STATUS interessant, die an-gibt, ob eine PDB

```
SQL> describe CDB_PDBS
 Name                                      Null?    Typ
 ----------------------------------------- -------- ----------------------
 PDB_ID                                    NOT NULL NUMBER
 PDB_NAME                                  NOT NULL VARCHAR2(128)
 DBID                                      NOT NULL NUMBER
 CON_UID                                   NOT NULL NUMBER
 GUID                                               RAW(16)
 STATUS                                             VARCHAR2(13)
 CREATION_SCN                                       NUMBER
 CON_ID                                             NUMBER
```

```
SQL> select pdb_name, status from cdb_pdbs;

PDB_NAME   STATUS
---------- -------------
ORCLPDB1   NORMAL
PDB$SEED   NORMAL
ORCLPDB2   NORMAL
ORCLPDB3   UNPLUGGED
ORCLPDB4   NEW
```

Starten und Stoppen

luxaviation

- Starten und Stoppen von Pluggable Datenbank
  - Einzelne PDBs können gestoppt werden, bzw. beim Stoppen wird die PDB in die Mount-Fase gebracht, sodass eine normale Anmeldung über TNS nicht mehr möglich ist.
  - Zum Beenden wechselt der Administrator in den Container und führt den standard-mäßigen SHUTDOWN-Befehl aus.
  - Danach wechselt der Status dieses Containers in die Mount-Fase.

Starten und Stoppen von Pluggable Datenbank

```
SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> shutdown immediate
Integrierbare Datenbank geschlossen.
SQL> select CON_ID, NAME, OPEN_MODE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE
---------- ------------------------------ ----------
         3 ORCLPDB1                       MOUNTED

SQL> alter session set container=CDB$ROOT;

Session wurde geandert.

SQL> select CON_ID, NAME, OPEN_MODE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE
---------- ------------------------------ ----------
         2 PDB$SEED                       READ ONLY
         3 ORCLPDB1                       MOUNTED
         4 ORCLPDB2                       READ WRITE
```

Starten und Stoppen von Pluggable Datenbank

- Für das Starten einer PDB wird der gleiche Vorgang mit dem STARTUP Befehl durchgeführt.

```
SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> startup
Integrierbare Datenbank geoffnet.
SQL> select CON_ID, NAME, OPEN_MODE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE
---------- ------------------------------ ----------
         3 ORCLPDB1                       READ WRITE
```

Starten und Stoppen

luxaviation

- Starten und Stoppen von Pluggable Datenbank
  - Wird die Containerdatenbank heruntergefahren und erneut gestartet, so bleiben die PDBs in der Mount-Fase. Um nun alle PDBs zu öffnen, können diese mit dem Befehl
    - alter pluggable database all open;
  - wieder ge einzeln starten z

```
Datenbank mounted.
Datenbank geoffnet.
SQL> select CON_ID, NAME, OPEN_MODE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE
---------- ------------------------------ ----------
         2 PDB$SEED                       READ ONLY
         3 ORCLPDB1                       MOUNTED
         4 ORCLPDB2                       MOUNTED

SQL> alter pluggable database all open;

Integrierbare Datenbank geandert.

SQL> select CON_ID, NAME, OPEN_MODE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE
---------- ------------------------------ ----------
         2 PDB$SEED                       READ ONLY
         3 ORCLPDB1                       READ WRITE
         4 ORCLPDB2                       READ WRITE
```

Standard Tablespace

luxaviation

- Standard permanent und temporären Tablespace ändern
  - alter pluggable database default tablespace [Name];
  - verwendet.
  - Für das Setzen des Standard temporären Tablespace kann der folgende Befehl angewendet werden:
    - alter pluggable database default temporary tablespace [Name];

21.9. Benutzerverwaltung in Multitenant

- Die Benutzerverwaltung ist ein wenig in einer Multitenant-Umgebung zu einer traditionellen installierten Datenbank.
- Hier wird zwischen lokalen und allgemeinen Benutzern unterschieden (Local Users und Common Users).
  - Allgemeine Benutzer sind verfügbar in allen Containern, während lokale Benutzer nur in einem Container präsent sein können.
  - Es können zwar lokale Benutzer mit gleichem Namen in unterschiedlichen PDBs vorhanden sein, es handelt sich aber dennoch um unterschiedliche Benutzer.

21.9. Benutzerverwaltung in Multitenant

- Analog zu den zwei unterschiedlichen Benutzertypen existieren zwei Rollentypen.
  - Allgemeine Rollen (Common Roles) sind in allen Containern verfügbar, während lokale Rollen (Local Roles) jeweils nur in ihrem Container existieren.
  - Alle von Oracle mitgelieferten Rollen sind allgemeine Rollen.

- Erstellung von allgemeinen Benutzern (Common Users)
  - Für die Erstellung von allgemeinen Benutzern müssen die folgenden Voraussetzungen getroffen sein.
    - Der aktuelle Container muss der Root-Container sein.
    - Es müssen CREATE USER-Berechtigungen zugeordnet sein.
    - Der neue Benutzername muss mit der Präfix c## beginnen.
    - Standard permanent und temporärer Tablespace muss in allen Containern existieren.
    - Der Benutzername muss über alle Container eindeutig sein.

Erstellung von allgemeinen Benutzern (Common U

```
SQL> create user c##marek identified by geheim123 default tablespace users temporary table-
space temp;

Benutzer wurde erstellt.

SQL> grant create session to c##marek;

Benutzerzugriff (Grant) wurde erteilt.

SQL> conn c##marek/geheim123@orasrv:1521/ORCLPDB1
Connect durchgefuhrt.

SQL> show con_name

CON_NAME
------------------------------
ORCLPDB1

SQL> conn c##marek/geheim123@orasrv:1521/ORCLPDB2
Connect durchgefuhrt.

SQL> show con_name

CON_NAME
------------------------------
ORCLPDB2
```

- Erstellung von lokalen Benutzern (Local Users)
  - Für die Erstellung von lokalen Benutzern müssen folgende Punkte erfüllt sein:
  - Der aktuelle Container muss die PDB sein, in der der lokale Benutzer erstellt werde soll.
  - Der Benutzer darf nicht mit der Präfix c## beginnen.
  - Der Benutzer muss eindeutig in der PDB sein.

Erstellung von lokalen Benutzern (Local

```
SQL> conn / as sysdba
Connect durchgefuhrt.
SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> create user marek identified by geheim123 default tablespace users temporary tablespace
temp;

Benutzer wurde erstellt.

SQL> grant create session to marek;

Benutzerzugriff (Grant) wurde erteilt.

SQL> conn marek/geheim123@orasrv:1521/ORCLPDB1
Connect durchgefuhrt.

SQL> conn marek/geheim123@orasrv:1521/ORCLPDB2
ERROR:
ORA-01017: Benutzername/Kennwort ungultig; Anmeldung abgelehnt

Achtung: Sie sind nicht mehr mit ORACLE verbunden.
```

- Erstellung von allgemeinen Rollen (Common Roles)
  - Die charakteristischen Eigenschaften einer allgemeinen Rolle sind analog zu einem allgemeinen Benutzer, denn diese Rolle ist verfügbar in allen Containern.
    - Der aktuelle Container muss der Root-Container sein.
    - Es müssen CREATE ROLE-Berechtigungen zu
    - muss mit der Präfix c## be
    - über alle Container ei

```
SQL> conn / as sysdba
Connect durchgefuhrt.
SQL> create role c##tollerolle;

Rolle wurde erstellt.

SQL> grant c##tollerolle to c##marek;

Benutzerzugriff (Grant) wurde erteilt.

SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> grant c##tollerolle to marek;

Benutzerzugriff (Grant) wurde erteilt.
```

- Erstellung von lokalen Rollen (Local Roles)
  - Lokale Rollen sind in ihrer Gültigkeit auf den Container beschränkt, in dem sie er-stellt wurden.
    - Der aktuelle Container muss die PDB sein, in der die lokale Rolle erstellt wer-den soll.
    - Die Rolle darf nicht mit der Präfix c## beginne-
    - Die Rol ~ PDB sein.

```
SQL> conn / as sysdba
Connect durchgefuhrt.
SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> create role tollerolle;

Rolle wurde erstellt.

SQL> grant tollerolle to c##marek;

Benutzerzugriff (Grant) wurde erteilt.

SQL> grant tollerolle to marek;

Benutzerzugriff (Grant) wurde erteilt.
```

Erstellen von PDBs

luxaviation

- Erstellen von Pluggable Datenbanken
  - Für die Erstellung einer Pluggable Database muss die Containerdatenbank geöffnet sein und darf sich weder im Read Only noch im Upgrade-Modus befinden.
  - Damit eine PDB erstellt werden kann muss der Ersteller die Systemberechtigungen CREATE PLUGGABLE DATABASE besitzen und muss mit dem Root-Container verbunden sein.
  - Bei der Erstellung kann eine Pluggable Datenbank aus der Vorlagendatenbank PDBSEED oder aber von einer vorhandenen PDB erstellt werden.

Erstellen von PDBs

luxaviation

- Soll eine PDB aus der Vorlagendatenbank Seed erstellt werden, so wird der folgende Befehl verwer

```
CREATE PLUGGABLE DATABASE [Name] ADMIN USER [Name] IDENTIFIED BY [Kennwort]
FILE_NAME_CONVERT=('[Seed-Pfad]', '[PDB-Pfad]'

SQL> conn / as sysdba
Connect durchgefuhrt.
SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL> create pluggable database ORCLPDB3 admin user ADMPDB3 identified by gehe
FILE_NAME_CONVERT=('orclcdb/pdbseed','orclcdb/orclpdb3');

Integrierbare Datenbank erstellt.

SQL> alter session set container=orclpdb3;

Session wurde geandert.

SQL> select name, open_mode from v$pdbs;

NAME                           OPEN_MODE
------------------------------ ----------
ORCLPDB3                       MOUNTED

SQL> alter pluggable database open;

Integrierbare Datenbank geandert.
```

Erstellen von PDBs

luxaviation

- Erstellen einer Pluggable Database aus einer vorhandenen Pluggable Database
  - Um eine PDB aus einer vorhandenen zu erstellen, muss die Quell-PDB im Vorfeld in den Read-Only Modus gebracht werden.
  - Danach kann die Erstellung mit dem folgenden Befehl durchgeführt werden:

Erstellen von PDBs

luxaviation

- Erstellen einer Pluggable Database aus einer vorhandenen Pluggable Database

```
CREATE PLUGGABLE DATABASE [Name] FROM [Quell-PDB] FILE_NAME_CONVERT=('[PDB-Pfad]','[PDB-Pfad]
';

SQL> conn / as sysdba
Connect durchgefuhrt.
SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL> alter session set container=ORCLPDB1;

Session wurde geandert.

SQL> shutdown immediate
Integrierbare Datenbank geschlossen.

SQL> startup open read only
Integrierbare Datenbank geoffnet.
SQL> alter session set container=CDB$ROOT;

Session wurde geandert.

SQL> create pluggable database ORCLPDB4 from orclpdb1
FILE_NAME_CONVERT=('orclcdb/orclpdb1','orclcdb/orclpdb4');

Integrierbare Datenbank erstellt.
```

An- und Abhängen von PDBs

- An- und Abhängen von Pluggable Datenbanken
  - PDBs können abgehängt und in die gleiche oder einer anderen Multitenant-Umgebung eingehängt werden.
  - Beim Abhängen wird eine XML-Datei für die PDB erzeugt, die für das spätere Einhängen alle wichtigen Metadaten besitzt.

Abhängen einer Pluggable Datenbank

- Damit eine Pluggable Datenbank abgehängt werden kann, muss sie zuerst geschlossen werden. Danach kann sie aus dem Root-Container

```
ALTER PLUGGABLE DATABASE [Name] UNPLUG INTO 'Speicherpfad der XML-Datei';

SQL> alter session set container=orclpdb3;

Session wurde geandert.

SQL> shutdown immediate
Integrierbare Datenbank geschlossen.

SQL> alter session set container=cdb$root;

Session wurde geandert.

SQL> alter pluggable database orclpdb3 unplug into '/u01/app/oracle/orclpdb3.xml';

Integrierbare Datenbank geandert.
```

An- und Abhängen von PDBs

luxaviation

- Einhängen einer Pluggable Database
  - Für das Einhängen einer PDB wird die XML-Datei mit den zu der PDB gehörenden Metadaten benötigt.
  - Ist diese nicht mehr vorhanden, so kann diese aus dem Packa-ge und der Prozedur DBMS_PDB.RECOVER neu erstellt werden.
  - Sollte zusätzlich die PDB in der View CDB_PDBS den Status UNPLUGGED haben, so muss diese PDB zuerst gelöscht werden, da sonst ein Einhängen nicht möglich ist.

Einhängen einer Pluggable Database

```
CREATE PLUGGABLE DATABASE [Name] USING 'Speicherpfad der XML-Datei' NOCOPY;

SQL> drop pluggable database ORCLPDB3;

Integrierbare Datenbank geloscht.

SQL> create pluggable database ORCLPDB3 using '/u01/app/oracle/orclpdb3.xml' nocopy;

Integrierbare Datenbank erstellt.
```

- Die Option NOCOPY signalisiert dem System, dass die PDB mit ihren Dateien schon an ihrem original Ort liegen.
- Sollen die Dateien an einen neuen Ort abgelegt werden, so ist die Option COPY in Verbindung mit FILE_NAME_CONVERT zu verwenden.

Patchen einer PDB

luxaviation

- Patchen einer Pluggable Datenbank
  - Abhängen der PDB und einhängen in die höhere Version
  - Öffnen der PDB im Upgrade-Mode
    - SQL> alter pluggable database PDB1 open upgrade;
  - Ausführen von catctl.pl
    - $ORACLE_HOME/perl/bin/perl catctl.pl -c "PDB1" -l /home/oracle/upgrade catupgrd.sql
  - Ausführen von utlrpt.sql
    - SQL> alter session set container=PDB1;
    - SQL> startup
    - @?/rdbms/admin/utlrp.sql