

# Multitalent Multitenant

Susanne Jahr

DOAG Konferenz und Ausstellung  
Nürnberg, 20. September 2022

# Herrmann & Lenz Services

- Über 25 Jahre am Markt
- >20 Mitarbeiter
- Experten für Datenbanken (Oracle, SQL Server, ...) sowie Oracle Database Appliances
- Beratung, Rufbereitschaft, Betrieb
- Schwesterfirma Herrmann & Lenz Solutions GmbH
  - Software-Produkte: HL Monitoring Module, Taskzone, Timetracking

# Inhalt

- Multitenant – brauche ich das wirklich?
- Anlage / Client-Zugriff
- Migrationen / Upgrades/ Testumgebungen

# Multitenant – Brauch ich das wirklich?

# Multitenant – brauche ich das wirklich?

- Kurze Antwort: JA
  - existiert seit Oracle 12c
  - verpflichtend ab Oracle 21c
  - Transparent für die Anwendung
    - Achtung bei Verwendung von ORACLE\_SID!!!
  - Best Practice: neue Systeme immer schon mit der neuen Architektur anlegen

# Anlage, Client-Zugriff

# Create Pluggable Database

- from seed
  - `create pluggable database MYPDB admin user PDBADMIN identified by <pw> roles=(connect, ...);`
  - zusätzliche Parameter möglich:
    - `file_name_convert=NONE | '<src_path>','<dst_path>'`
    - `service_name_convert`
    - `storage(maxsize unlimited max_share_temp_size unlimited)`
- Unplug / Plug
  - `alter pluggable database MYPDB unplug into mypdb.xml | mypdb.pdb`
  - `create pluggable database NEWPDB using 'mypdb.xml' | 'mypdb.pdb';`
- per Remote Clone → bestens geeignet für Migrationen und Aufbau von Testsystemen

# Client-Zugriff

- Nach Umstellung auf Multitenant-Architektur:
- Hilfe, meine Clients können sich nicht mehr anmelden!

- TNS-Eintrag:

MYDB =

(DESCRIPTION=

(ADDRESS= (PROTOCOL=TCP) (HOST=192.168.149.1) (PORT=1521))

(CONNECT\_DATA= (SERVER=dedicated) (SID=MYDB))

)

- JDBC-Connect-String mit JDBC Thin Driver:

jdbc:oracle:thin:@192.168.149.1:1521:MYDB

**Don't.**

**Just**

**don't.**



# Client-Zugriff


- ORACLE\_SID:
  - Name für eine Oracle Datenbank**instanz** auf einem bestimmten Host
  - in Multitenant-Architektur: CDB\$ROOT! PDBs haben keine eigenen Instanzen
- Also: Verwendung von Service-Namen!
  - TNS-Eintrag:

```
MYDB =  
  (DESCRIPTION=  
    (ADDRESS= (PROTOCOL=TCP) (HOST=192.168.149.1) (PORT=1521) )  
    (CONNECT_DATA= (SERVER=dedicated) (SERVICE_NAME=MYDB.world) )  
  )
```

- JDBC-Connect-String mit JDBC Thin Driver:

```
jdbc:oracle:thin:@//192.168.149.1:1521/MYDB.world
```

# Client-Zugriff mit Service-Namen

- Immer da, sobald die PDB geöffnet ist: Default-Service-Name
  - PDB: <pdb\_name>.<db\_domain>
  - Non-CDB oder CDB: <db\_unique\_name>.<db\_domain>
- Kann man schon so machen, aber:
- Besser: Benutzerdefinierte Services!

# Client-Zugriff mit Service-Namen

- Bei vorhandener Grid Infrastructure:

```
srvctl add service -db <db_unique_name> -service <myfancyservice> -pdb  
<mypdb> [-role PRIMARY]
```

- Ohne Grid Infrastructure:

```
alter session set container=mypdb;  
exec dbms_service.add_service('myfancyservice','myfancyservice');  
exec dbms_service.start_service('myfancyservice');
```

- Autostart z.B. durch Startup-Trigger
- Abfrage cdb\_services liefert alle existierenden Service-Namen
  - innerhalb der ganzen CDB (Abfrage in der Root)
  - innerhalb der lokalen PDB (Abfrage in der PDB)
- Instanz-Parameter service\_names ist in 19c deprecated und soll nicht mehr manuell gefüllt werden!

# Client-Zugriff (aber aber aber...)

- Aber unsere Anwendung ist von 1976 und kann nur SIDs und die Firma ist schon seit 1994 pleite und wir können da gar nix dran machen...



- OK – na gut... es gibt (noch) einen Weg, die SID zu retten:
- Parameter `USE_SID_AS_SERVICE_LISTENER=ON` in der `listener.ora` bewirkt, daß die SID bei eingehenden Requests als Service-Name interpretiert wird und somit funktioniert
- Im Net Services Guide Version 21c noch enthalten...
- **Trotzdem: definierte Services sind besser!!**

# Gedanken im Vorfeld...

# Service- und andere Namen...

- Ziel: möglichst wenig an den Clients ändern
- Also: bisherige Service-Namen möglichst weiterhin verwenden
- Wenn bisher Default-Service-Namen: Neuer db\_name
- Wenn bisher schon benutzerdefinierte Service-Namen: alles ok – aber der Host...?
- Praktisch: DNS-Alias für den DB-Host

# Namenskonventionen

- Denkbare Namenskonvention:
  - C-Präfix vor den bisherigen DB-Namen (Achtung: 8-Zeichen-Regel gilt weiterhin für den db\_name!)
  - P-Präfix für den/die PDB-Namen
- Vorher (Non-CDB):
  - db\_name: PROD
  - service\_name: PROD
  - Hostname: myhost
  - DNS-Alias: dbhost
- Nachher (CDB):
  - db\_name: CPROD
  - pdb\_name: PPROD
  - service\_name für PPROD: PROD
  - Hostname: mynewhost
  - DNS-Alias dbhost: wird nach der Migration von myhost auf mynewhost geschwenkt

# Migrationen / Upgrades



# Klassiker bei Migrationen: Datapump

- Anlage neue DB (der selbe Server oder ein neuer)
- Datapump Export aus der alten DB
- Datapump Import in die neue DB
- Nacharbeiten (SYS-Grants, PUBLIC Objekte, ...)

# Datapump Export/Import vs. Remote Clone

- Datapump:
  - traditionelle Methode
  - Vorteile
    - Reorganisation
    - weitere Aktionen möglich (Unicode-Migration, Remapping, Übernahme Subset der Original-Daten, ...)
    - Über Plattformen hinweg möglich
  - Nachteile:
    - ggfs. Platzbedarf für Dump File(s)
    - Achtung bei Importen im Archivelog-Modus!
    - ggfs. lange Dauer, insbesondere bei LOBs
    - → bei großen Systemen ggfs. lange Downtime erforderlich

# Datapump Export/Import vs. Remote Clone

- Remote Clone:
  - PDB in der Zielumgebung wird aus Quell-Datenbank aufgebaut
  - Quelle kann PDB oder Non-CDB sein (min. 12.2)
  - Vorteile:
    - insb. bei großen Systemen und / oder vielen LOBs deutlich schneller als Datapump Export / Import
    - keine riesigen Archivelog-Mengen
    - Umwandlung der alten in die neue Architektur "in einem Aufwasch"
    - gleichzeitiger Upgrade möglich – manuell oder integriert in Autoupgrade – sogar refreshable
    - im Fehlerfall schnell reproduzierbar
  - Nachteile:
    - Quell-DB kann nur 1:1 übernommen werden, keine Änderungen innerhalb der DB möglich, keine Reorganisation
    - nicht möglich für Quell-Datenbanken mit alten Versionen
    - nur innerhalb der selben Plattform

# Remote Clone - Voraussetzungen

- Quell-DB:
  - PDB oder Non-CDB (min. 12.2)
  - Kommunikation mit Ziel-DB über Oracle Net möglich
  - User mit dem CREATE PLUGGABLE DATABASE Privileg (Common User oder normaler User in Non-CDBs)
  - installierte Komponenten Quell-(P)DB  $\leq$  installierte Komponenten Ziel-CDB
- Ziel-DB:
  - CDB ab 12c; am besten 19c oder 21c
  - Database Link zur Quell-DB unter Verwendung des o.a. Users

# Remote Clone: Durchführung

- Auf der Ziel-CDB
  - Quelle ist **PDB**:
    - `create pluggable database <dst_pdb> from <src_pdb>@<db_link>;`
    - ggfs. noch:
      - `$ORACLE_HOME/OPatch/datapatch -verbose -pdba <dst_pdb> -skip_upgrade_check`

# Remote Clone: Durchführung

- Auf der Ziel-CDB
  - Quelle ist **Non-CDB**:
    - `create pluggable database <dst_pdb> from NON$CDB@<db_link>;`
    - `alter session set container=<dst_pdb>;`
    - `@?/rdbms/admin/noncdb_to_pdb.sql`
    - ggfs. noch:
      - `$ORACLE_HOME/OPatch/datapatch -verbose -pdba <dst_pdb> -skip_upgrade_check`

# Remote Clone: Durchführung

- Am besten: OMF verwenden!
  - Ansonsten abweichende Pfade durch Anwendung von `file_name_convert` definieren
- Kollidierende Service-Namen können durch Anwendung von `service_name_convert` verhindert werden
- Achtung: während des Remote-Clonings auf der Quellseite keine Löschung von Archivelogs!

# Sonderfall: Unterschiedliche Versionen

- Zusätzlicher Schritt: Upgrade der neuen PDB!
- Upgrade erfolgt nach dem Cloning auf der Zielseite
- Quell-DB bleibt intakt → Fallback
- Auf der Quell-DB:
  - Preupgrade-Checks
    - preupgrade.jar oder
    - autoupgrade.jar –mode analyze (einzige Methode bei Upgrade nach 21c)



# Sonderfall: Unterschiedliche Versionen

- Auf der Ziel-CDB
  - Quelle ist **PDB**:
    - `create pluggable database <dst_pdb> from <src_pdb>@<db_link>;`
    - `alter pluggable database <dst_pdb> open upgrade;`
    - `$ORACLE_HOME/bin/dbupgrade -c <dst_pdb>`
  - Quelle ist **Non-CDB**:
    - `create pluggable database <dst_pdb> from NON$CDB@<db_link>;`
    - `alter pluggable database <dst_pdb> open upgrade;`
    - `$ORACLE_HOME/bin/dbupgrade -c <dst_pdb>`
    - `alter session set container=<dst_pdb>;`
    - `@?/rdbms/admin/noncdb_to_pdb.sql`

# Remote Clone mit Autoupgrade

- Preupgrade-Checks ab 21c nicht mehr mit `preupgrade.jar`, sondern mit `autoupgrade.jar -mode analyze`
- Download aktuelles `autoupgrade.jar` auf MOS Doc ID **2485457.1**
- Upgrade und Patching sowohl In-Place als auch innerhalb des Remote Cloning in Autoupgrade integriert
- Quell- und Ziel-DB auf dem selben oder auf unterschiedlichen Servern möglich
- Voraussetzungen identisch zum manuellen Remote Cloning

# Remote Clone mit Autoupgrade

- **Achtung:** der PDB-Creator-User auf der Quell-DB braucht zusätzlich das READ-Recht auf die Tabelle **sys.enc\$** - auch wenn keine Encryption verwendet wird!!!

- Im Oracle Upgrade Guide 21c steht:

*"...The PDB created from the non-CDB must continue to use the source non-CDB name. You cannot change the name of the database. ..."*

→ falsch! Abweichende PDB-Namen sind möglich; sie müssen nur in der Config-Datei angegeben werden

# Remote Clone mit Autoupgrade

- Autoupgrade-Config-Datei

```
global.autoupg_log_dir=/u01/app/oracle/admin/DAT18/script/autoupgrade/log
dat18upg.source_home=/u01/app/oracle/product/18.0.0.0/dbhome_1
dat18upg.target_home=/u01/app/oracle/product/19.0.0.0/dbhome_1
dat18upg.sid=DAT18
dat18upg.source_dblink.DAT18=CLONEDB
dat18upg.target_cdb=CDB19
dat18upg.target_pdb_name.DAT18=PDB19
dat18upg.target_version=19.16
dat18upg.target_pdb_copy_option.DAT18=file_name_convert=NONE
```

- Wenn keine OMF genutzt werden: file\_name\_convert verwenden!

# Remote Clone mit Autoupgrade

- Preupgrade-Check mit:

```
$ORACLE_HOME/jdk/bin/java -jar autoupgrade.jar -mode analyze
```

- Ggfs. erforderliche Fixups mit:

```
$ORACLE_HOME/jdk/bin/java -jar autoupgrade.jar -mode fixups
```

- Remote Clone und Upgrade mit:

```
$ORACLE_HOME/jdk/bin/java -jar autoupgrade.jar -mode upgrade
```

- Oder alles mit:

```
$ORACLE_HOME/jdk/bin/java -jar autoupgrade.jar -mode deploy
```

# Remote Clone mit Autoupgrade

AutoUpgrade 22.4.220712 launched with default internal options

Processing config file ...

+-----+

| Starting AutoUpgrade execution |

+-----+

1 PDB(s) will be processed

upg> lsj

+---+-----+-----+-----+-----+-----+-----+-----+

|Job#|DB\_NAME| STAGE|OPERATION| STATUS|START\_TIME|UPDATED|MESSAGE|

+---+-----+-----+-----+-----+-----+-----+-----+

| 101| DAT18|**CLONEPDB**|EXECUTING|RUNNING| 19:01:18| 3s ago| |

+---+-----+-----+-----+-----+-----+-----+-----+

Total jobs 1

...

upg> Copying remote database 'dat18' as 'PDB19' for job 101

**Remote database 'dat18' created as PDB 'PDB19'** for job 101

...

# Remote Clone mit Autoupgrade

```
+---+-----+-----+-----+-----+-----+-----+-----+
|Job#|DB_NAME|  STAGE|OPERATION| STATUS|START_TIME| UPDATED|      MESSAGE|
+---+-----+-----+-----+-----+-----+-----+-----+
| 101| DAT18|DBUPGRADE|EXECUTING|RUNNING| 19:01:18|102s ago|39%Upgraded PDB19|
+---+-----+-----+-----+-----+-----+-----+-----+
...
+---+-----+-----+-----+-----+-----+-----+-----+
|Job#|DB_NAME|  STAGE|OPERATION| STATUS|START_TIME|UPDATED|      MESSAGE|
+---+-----+-----+-----+-----+-----+-----+-----+
| 101| DAT18|NONCDBTOPDB|EXECUTING|RUNNING| 19:01:18|37s ago|noncdb_to_pdb - 70%|
+---+-----+-----+-----+-----+-----+-----+-----+
...
+---+-----+-----+-----+-----+-----+-----+-----+
|Job#|DB_NAME|  STAGE|OPERATION| STATUS|START_TIME|UPDATED|      MESSAGE|
+---+-----+-----+-----+-----+-----+-----+-----+
| 101| DAT18|NONCDBTOPDB|EXECUTING|RUNNING| 19:01:18|25s ago|80% utlrp 11|
+---+-----+-----+-----+-----+-----+-----+-----+
...
```



# Remote Clone mit Autoupgrade

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|Job#|DB_NAME|   STAGE|OPERATION| STATUS|START_TIME|UPDATED|MESSAGE|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101|  DAT18|POSTFIXUPS|EXECUTING|RUNNING| 19:01:18|25s ago|      |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

upg> Job 101 completed

----- Final Summary -----

Number of databases [ 1 ]

Jobs finished [1]

Jobs failed [0]

Jobs restored [0]

Jobs pending [0]

Please check the summary report at:

/u01/app/oracle/admin/DAT18/script/autoupgrade/log4/cfgtoollogs/upgrade/auto/status/status.html

/u01/app/oracle/admin/DAT18/script/autoupgrade/log4/cfgtoollogs/upgrade/auto/status/status.log



# Testumgebungen

# Testumgebungen

- Analog zur Migration von produktiven Umgebungen lassen sich auch Testumgebungen per Remote Clone schnell und einfach neu aufbauen
- Bis zu 3 PDBs pro Container in der SE2/EE ohne Multitenant Option in 19c möglich (vorher eine)
- Kein Shutdown des Root-Containers oder der anderen PDBs erforderlich
- Einfach per Skript und z.B. cronjob zu automatisieren
- Pre- und Post-Aktionen einplanbar, z.B.
  - vor Beginn Datapump Export der vorhandenen Metadaten
  - nach Abschluss Erstellung / Entfernung nicht mehr benötigter Services

# Beispiel

```
export ORACLE_SID=CTEST
export DST_PDB=PMCR2
export SRC_PDB=MCR2
export DBLINK=CLONEMCR2
export NLS_DATE_FORMAT='dd.mm.yyyy hh24:mi:ss'

srvctl add service -d CTEST -s MCR2 -pdb PMCR2
sqlplus / as sysdba @drop_pdb $DST_PDB
sqlplus / as sysdba @cr_pdb.sql $DST_PDB $SRC_PDB $DBLINK
$ORACLE_HOME/OPatch/datapatch -verbose -pdb $DST_PDB -skip_upgrade_check
sqlplus / as sysdba @noncdb2pdb $DST_PDB
sqlplus / as sysdba @save_state.sql $DST_PDB
srvctl start service -d CTEST -s MCR2
```



- Mail:
- [susanne.jahr@hl-services.de](mailto:susanne.jahr@hl-services.de)
- Web: [www.hl-services.de](http://www.hl-services.de)
- Blog: [blog.hl-services.de](http://blog.hl-services.de)
- Unsere weiteren Vorträge:
  - Ist Ihre Datenbank 23c-Ready?  
(Dierk Lenz, Do 11:00 Tokio)
  - Mehr als 4 Jahre ODA Betrieb: Gute und weniger gute Erfahrungen  
(Rastislav Ciganek / Festo AG & Co. KG & Dierk Lenz, Do 14:00 Shanghai)