SEMINARE

Workshop

Oracle 12*c* - Multitenant Architecture

MD CONSULTING & INFORMATIONSDIENSTE GMBH

MD
CONSULTING & INFORMATIONSDIENSTE GMBH

Gründung:
1991

Büros:
Erfurt und München

35 Mitarbeiter

Vertriebsgebiet:
Deutschsprachiger Raum

CONSULTING & INFORMATIONSDIENSTE GMBH

Datenbankprodukte
- MS SQL Server
- Gupta
- Oracle
- ...

MD
CONSULTING & INFORMATIONSDIENSTE GMBH

Datenbank-Technologien

Software-Entwicklung

Programmiersprachen

Entwicklungswerkzeuge

MD
CONSULTING &
INFORMATIONSDIENSTE
GMBH

Nationale und internationale Unternehmenslösungen

Consulting

Projektcoaching

Projektrealisierung

Kunden / Anwendungsentwicklung

Düsseldorf	DOW Living. Improved daily.		Immenstaad
München	BMW MINI	Boehringer Ingelheim	Ingelheim
München, Nürnberg, Erlangen	SIEMENS	OPEL	Eisenach
Ingolstadt	MediaMarkt	LIDL	Neckarsulm

Agenda

SEMINARE

1

Konzepte, Möglichkeiten und Vorteile der Pluggable Databases – Multitenant Architecture

MD CONSULTING & INFORMATIONSDIENSTE GMBH

Private Cloud Datenbank
Architekturen mit Oracle 11g und 12c

- Oracle Database 11g

Virtuelle Maschinen	Dedizierte Datenbanken	Schema Konsolidierung
Gemeinsame Server	Gemeinsamer Server und OS	Gemeinsamer Server, OS und Datenbank

Zunehmende Konsolidierung

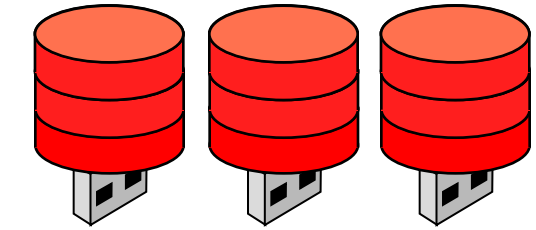
- Oracle Database 12c

Virtuelle Maschinen	Dedizierte Datenbanken	**Multitenant Datenbank**
Gemeinsame Server	Gemeinsamer Server und OS	Gemeinsamer Server, OS und Datenbank

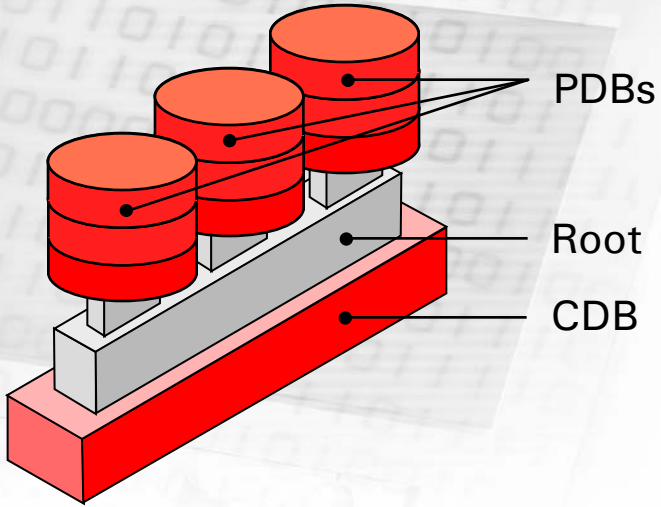
Zunehmende Konsolidierung

Multitenant Architektur – Komponenten

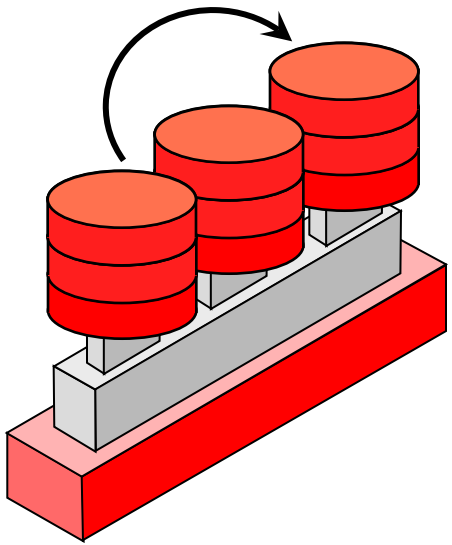
- Die neue Multitenant Architektur
  - Hauptspeicher und Prozesse werden nur noch auf Ebene des „multitenant containers” benötigt
  - Komponenten einer Multitenant Container Database (CDB)



Pluggable Databases (PDBs)



Multitenant Architektur – Merkmale

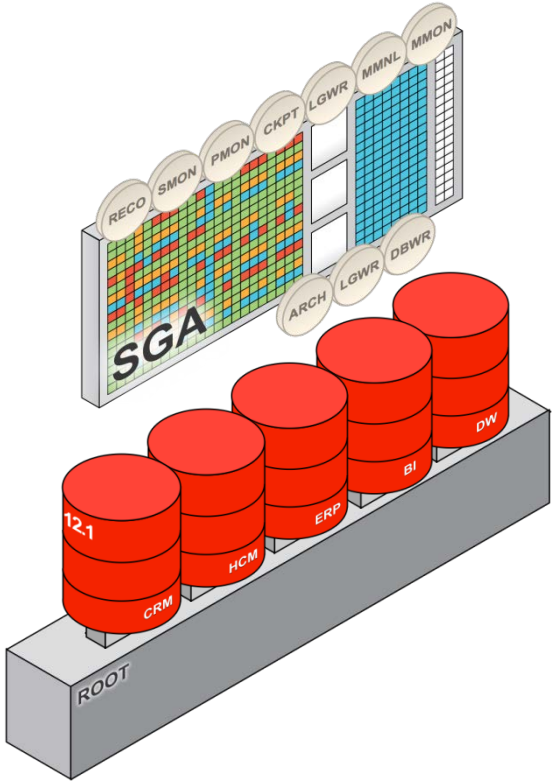


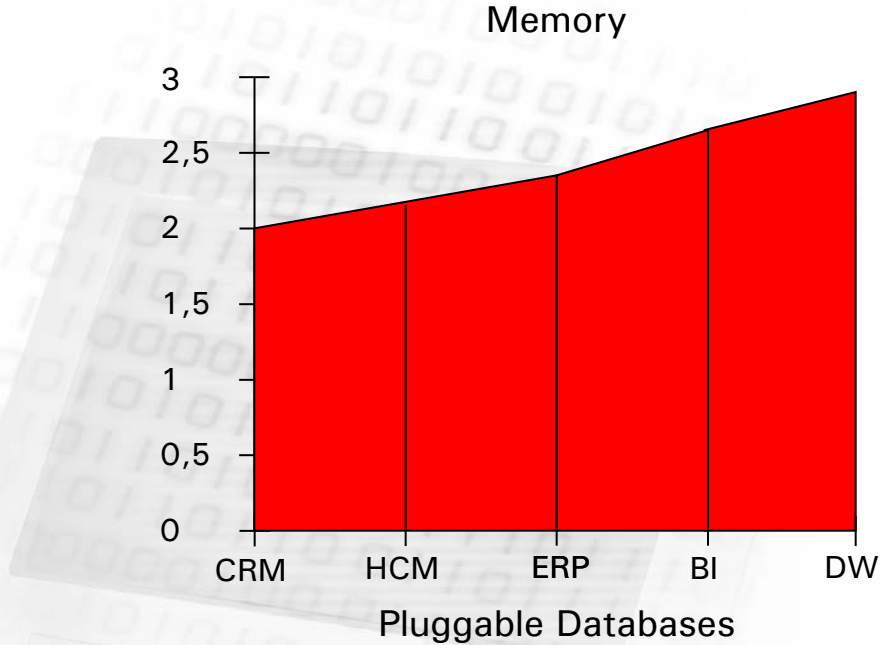
- Merkmale
  - die Multitenant Architektur unterstützt derzeit bis zu 252 PDBs
  - eine PDB verhält sich identisch zu einer non-CDB
  - ein DB-Client kann nicht erkennen, ob er an einer PDB oder einer non-CDB angemeldet ist.
  - EE, > 1 PDB kostenpflichtige Option

Multitenant Architektur – Dynamische Anteile

- Dynamische Anteile
  - PDBs „teilen sich” SGA und Hintergrundprozesse
  - optionale Resourcenbegrenzung und -verteilung
  - Vordergrundprozesse (DB Sessions) „sehen” nur die PDB an der sie angemeldet sind
- Multitenant Skalierbarkeit
  - jeweils nur kleiner Speicherzuwachs beim Hinzufügen weiterer PDBs

Multitenant Architektur – Skalierbarkeit





- jeweils nur kleiner Speicherzuwachs beim Hinzufügen weiterer PDBs

Multitenant Architektur – 3

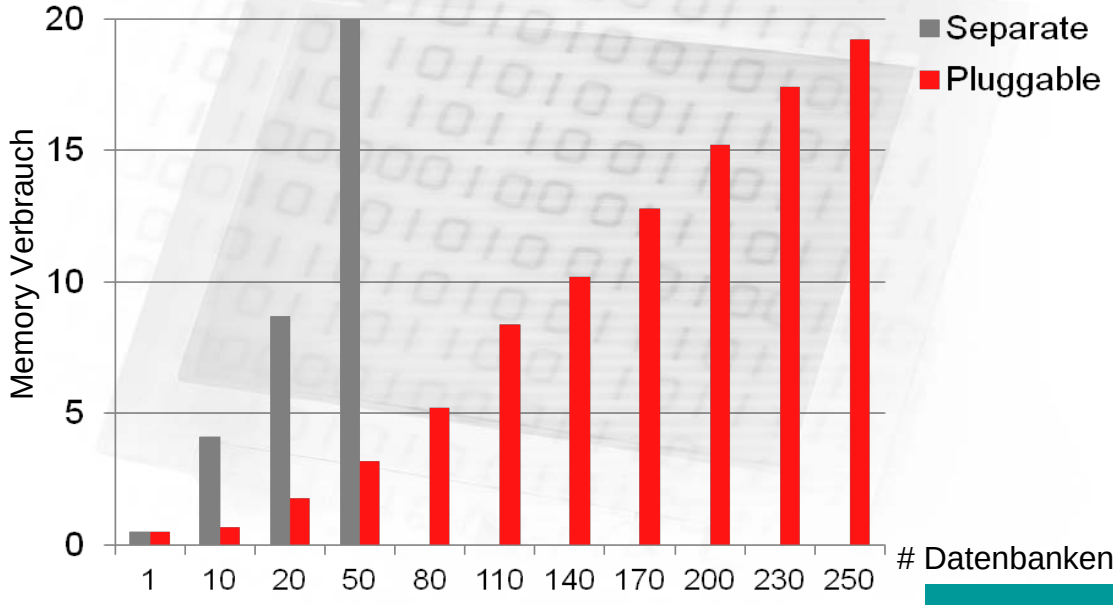
- Pluggable Databases vs. separate Datenbanken - höchst effizient:
  6x weniger H/W-Ressourcen,
  5x skalierbarer

OLTP Benchmark Vergleich

Nur **3GB** Memory **vs. 20GB** Memory für 50 Datenbanken

Pluggable Datenbanken skalieren bis **252 Instanzen**

Separate Datenbanken maximal **50 Instanzen**



Multitenant Architektur – 4

- Dateien in der Multitenant Architektur
  - jede PDB hat einen eigenen Satz Tablespaces, inkl. SYSTEM und SYSAUX
  - PDBs teilen sich UNDO, REDO und Control Files, (S)PFILE
  - als Default hat die CDB einen einzelnen Tablespace TEMP für alle, PDBs können aber ihren eigenen anlegen

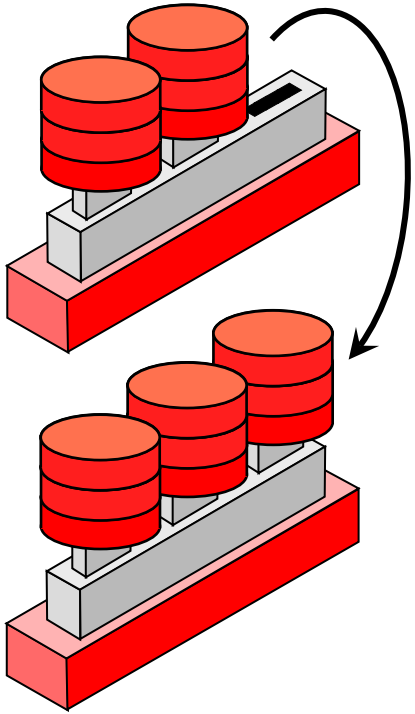
Datenbankbenutzer in der Multitenant Architecture

- *Lokale Nutzer* (*„local user"*)
  - entsprechen selbst erstellten Nutzern in einer non-CDB
  - … sind nur in einer PDB definiert
  - … können eine PDB nach Erhalt entsprechender Privilegien administrieren
- *Globaler Nutzer* (*„common user"*)
  - ist im „root" definiert und wird in jeder PDB repräsentiert
  - kann sich an jede PDB anmelden, in der er Privileg „Create Session" hat und kann diese ggf. auch administrieren
  - Oracle-eigene Systemobjekte gehören „common users"

Users und Privilegien

- Common Users und Privilegien
  - einem „common user" können Privilegien lokal für eine PDB (oder „root") zugewiesen werden und daher in jedem Container unterschiedlich sein
  - alternativ können einem „common user" System Privilegien auch allgemein („commonly") zugewiesen werden – die Zuweisung gilt dann im „root" und jeder PDB (auch in zukünftigen)
  - es können „common roles" erstellt werden
  - eine „common role" kann einem „common user" allgemein („commonly") zugewiesen werden
  - Autorisierung erfolgt ausschließlich mit den Rechten, die der User in dem Container hat, in dem er das SQL ausführen will

Unplug / Plug der PDBs – 1



- Einfach aus der alten CDB „ausklinken"…
- …und in die andere CDB „einklinken"…

Unplug / Plug der PDBs – 2

- Verschieben zwischen CDBs erfordert lediglich das Verschieben der Metadaten der PDB
- Upgrades und Patching werden dadurch einfacher
- eine „ausgeklinkte” PDB beinhaltet Infos zu lineage, opatch, encryption keys etc.

Neue Möglichkeiten – 1

- „Manage Many as One” mit Multitenant
  - gemeinsames DB-Backup – Recovery und ggf. Point-in-time Recovery auf Pluggable Database - Ebene
  - eine Standby Datenbank deckt alle Pluggable Databases ab
  - vereinfachtes Patching:
    - Änderungen nur einmal anwenden, alle Pluggable Databases sind aktualisiert
    - Upgrade in-place
  - Flexibilität beim Patchen und Upgraden von Datenbanken

Neue Möglichkeiten – 2

- verbesserte Agilität bei Änderungen der DB-Last
  Cluster-Erweiterung für flexible Konsolidierungsmodelle
- eine PDB kann SLAs „durchwandern”,
  je mehr „mission critical” sie wird
- Pluggable Datenbanken können schnell
  aus „Seed DB” erzeugt werden
- schnelles Cloning von PDBs
  - innerhalb der gleichen CDB
  - aus “remote CDBs”

Vorteile der Multitenant Architektur

- „Self-contained PDB" für jede Anwendung
  - Applikationen laufen unverändert
  - Schnelles Ausrollen (über Clones)
  - Portabilität (durch „Plug/Unplug")
- gemeinsame Nutzung von RAM und Prozessen
  - Mehr Applikationen pro Server
- gemeinsame Verwaltung auf CDB-Ebene
  - „Manage many as one" (Upgrade, High Ave, Backup)
  - Granulare Kontrolle wo angemessen

Anwendungsfälle Multitenant DBs

- Multitenant für Test und Entwicklung
  schnelle, flexible Kopien und Snapshots von Pluggable Databases
- Konsolidierung von vereinzelten Anwendungen
  „teilt" den Overhead von Speicher und Prozessen
- Self-Service Database as a Service (DBaaS)
  Auswählen von Größe und Service Level
- ideal für Software as a Service (SaaS)
  mandantenfähig durch die Datenbank, nicht die Anwendung
- ideal für Independent Software Vendors (ISVs)
  anstelle von langwierigen Setup-Skripten u.ä.:
  Verteilung von vorkonfigurierten Anwendungen als PDB

SEMINARE

2

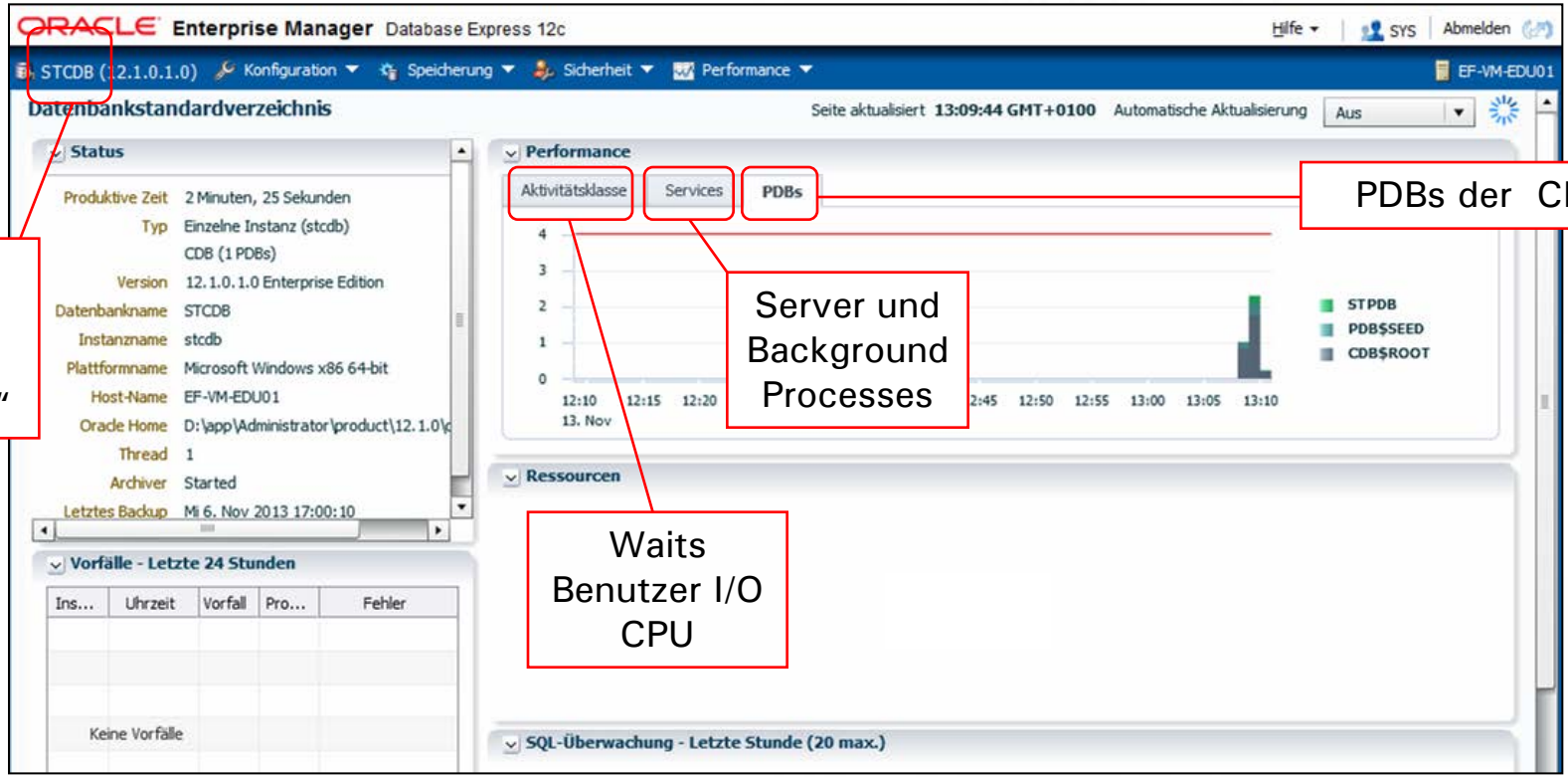
Utilities zum Management der Multitenant Architecture

Verwaltungs-Tools für Oracle12*c*

einschließlich Multitenant Architecture

- Oracle EM 12c Cloud Control
  - zentrale Konsole für alle Oracle Systeme
  - modularer Aufbau - rasche Unterstützung neuer Systeme
- Enterprise 12*c* Database Express (EM DB Express)
  - Monitoring und Management einer *einzelnen* Oracle DB12*c*
  - webbasiert unter Verwendung der XDB
  - automatische Installation
  - Konfiguration durch Angabe bei DBCA oder später mit PL/SQL
  - ersetzt EM Database Control / weniger Overhead
- OUI und DBCA
- SQL*Plus und SQL*Developer

Oracle EM 12*c* Database Express – CDB



im Beispiel CDB "STCDB"

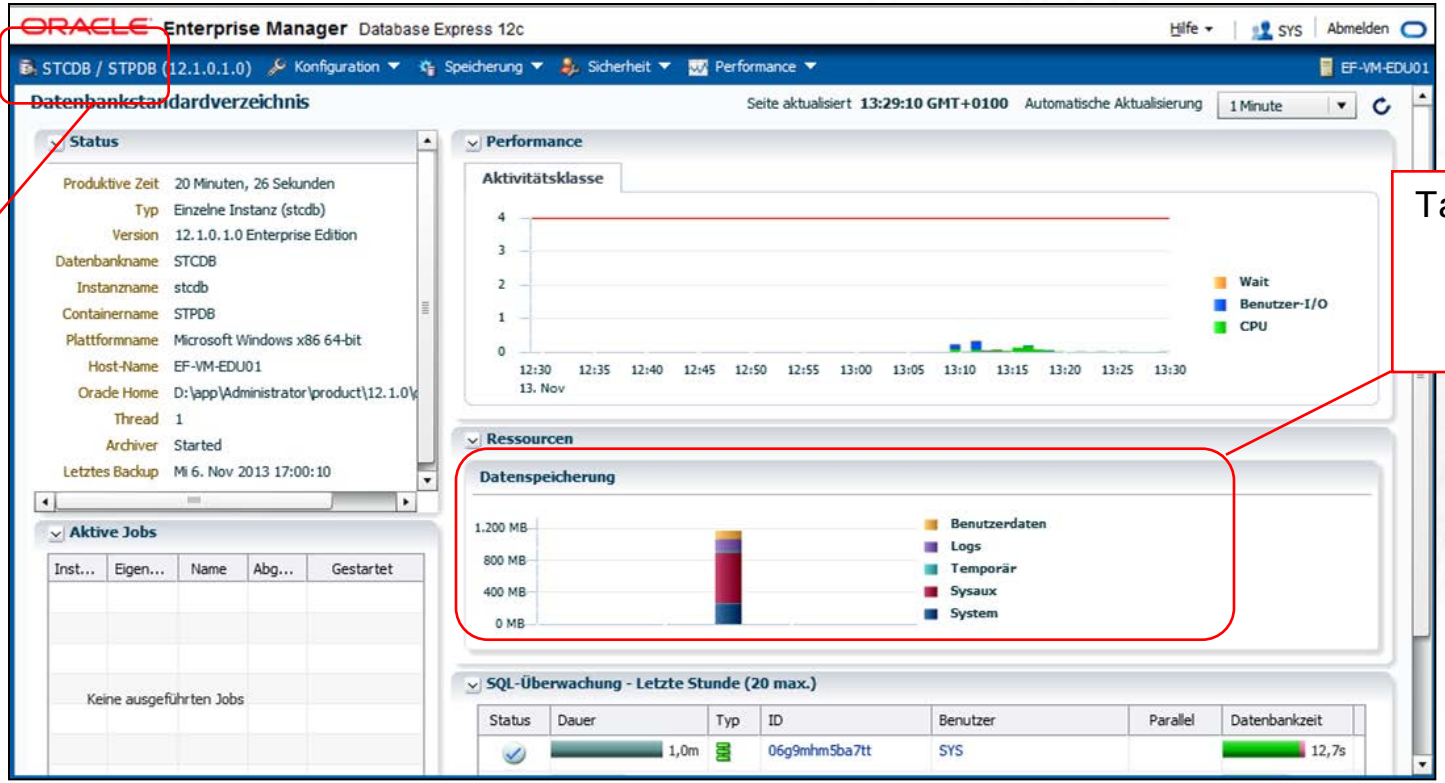
PDBs der CDB

Server und Background Processes

Waits
Benutzer I/O
CPU

Oracle EM 12*c* Database Express – PCB

im Beispiel PDB “STPDB“ in CDB “STCDB“



Tablespaces der PDB “STPDB“

Oracle EM 12*c* Database Express – unterschiedliche Menüpunkte CDB/PDB

Kategorie	CDB	PDB
Konfiguration	Initialisierungsparameter Speicher Verwendung von Datenbank-Features Aktuelle Datenbankeigenschaften	Initialisierungsparameter
Speicherung	Undo-Management Redo-Loggruppen Archive-Logs Kontrolldateien	Tablespaces
Sicherheit	Benutzer Rollen	Benutzer Rollen Profiles
Performance	Performance-Hub SQL Tuning Advisor	Performance-Hub SQL Tuning Advisor

kontext-bezogen

Oracle EM 12*c* Database Express – Konfiguration der Ports

- DBCA konfiguriert EM Express für die CDB, nicht für die PDBs
- Setzen des Ports für die jeweils kontaktierte Datenbank

  möglich für alle Container einschließlich CDB$ROOT
  - ungesichert

    ```
    execute DBMS_XDB_CONFIG.SETHTTPPORT(portnumber);
    ```

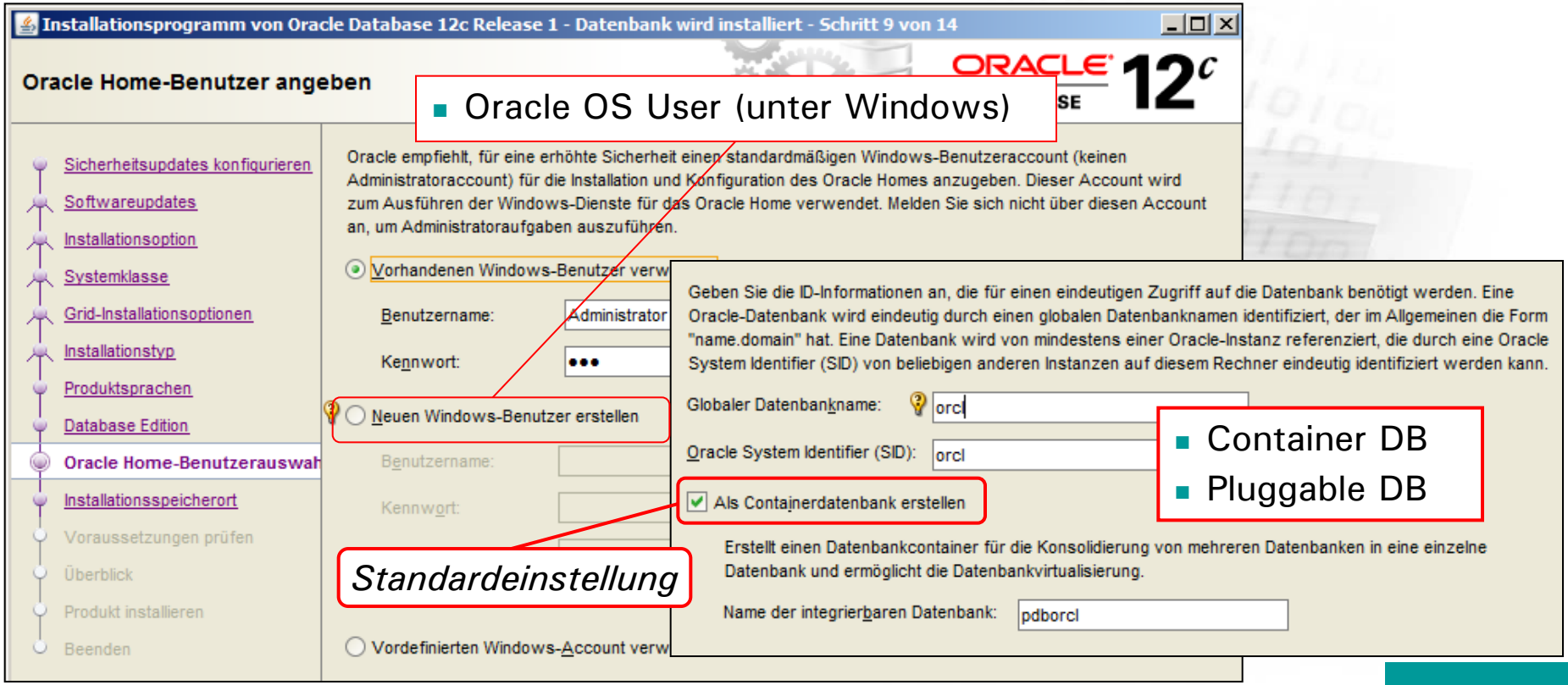
  - gesichert

    ```
    execute DBMS_XDB_CONFIG.SETHTTPSPORT(portnumber);
    ```

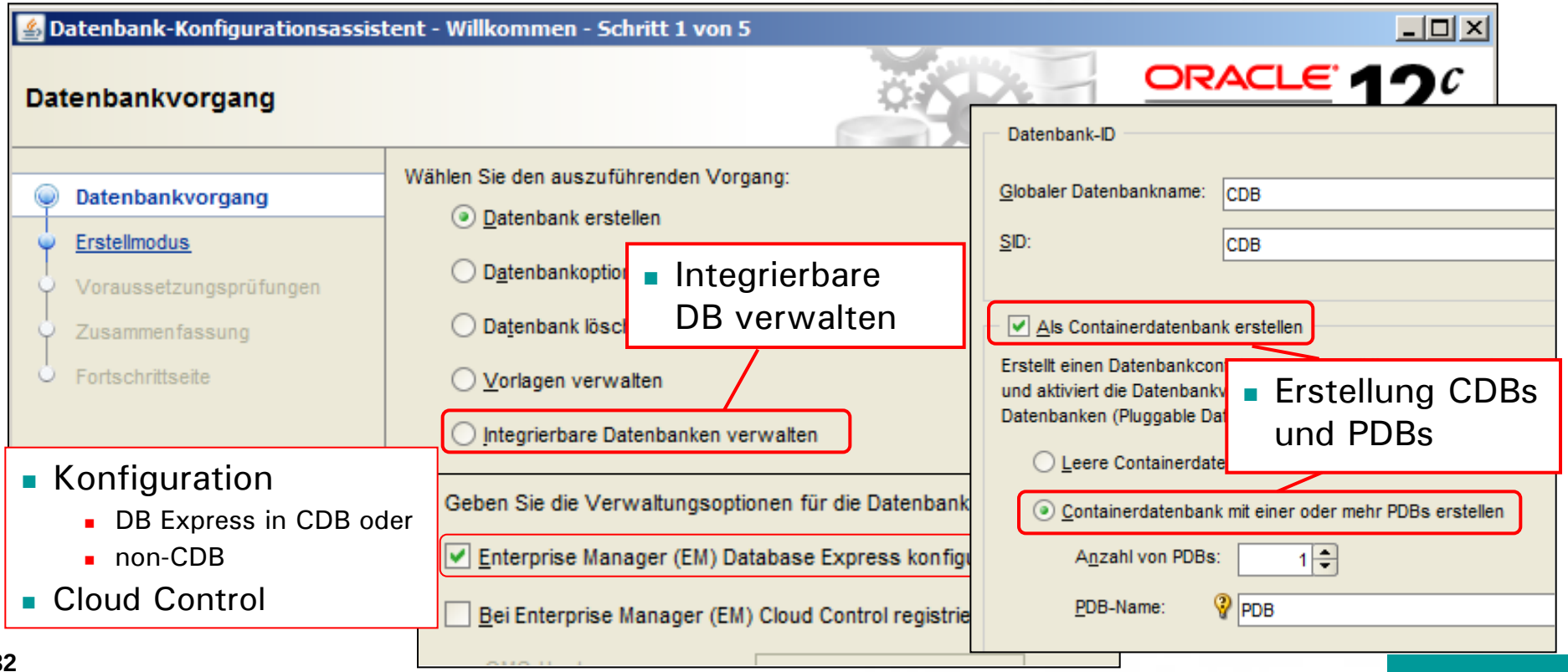
OUI und DBCA

- OUI
  - bei optionaler Erzeugung der DB
    Erstellung CDBs und PDBs möglich
- Datenbank-Konfigurationsassistent unterstützt Multitenant Architektur
  - Erstellung CDBs und PDBs
    - Standardkonfiguration: nur 1 PDB, nur OMF
    - Erweiterter Modus: mehrere PDBs,
      ASM, Dateisystem (OMF nicht zwingend)
  - Unplug / Plug der PDBs

Oracle Universal Installer



DBCA



- Konfiguration
  - DB Express in CDB oder
  - non-CDB
- Cloud Control

SEMINARE

3

Management der Multitenant Architecture

MD CONSULTING & INFORMATIONSDIENSTE GMBH

Management der Multitenant Architecture

- Schwerpunkte
  - Zugriff auf die PDBs
  - Öffnen und Schließen der PDBs
  - Unplug / Plug der PDBs – Konzept
  - Verwalten von geteilten Ressourcen
  - Änderungen und Neuerungen der dynamischen Performance Views und Data Dictionary Views
  - globale und lokale Systemparameter
  - Datenbankdateien in der Multitenant Architecture
  - Erstellung und Verwaltung der Benutzer

Zugriff auf die PDBs

- Zugriff auf Pluggable Databases
  für „Normal“-Benutzer
  nur über Service erfolgreich
  bei lokaler Benennungen deshalb nur möglich:
  - EASY CONNECT mit Angabe Service

```
CONNECT user/password@host[:port]/ service
```

  - TNSNAMES mit Angabe Service

```
net_alias_name =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = ...))
   (CONNECT_DATA = (SERVICE_NAME = service) ))
```

Öffnen der PDBs

- nach Startup der CDB-Root einschließlich Öffnen sind die PDBs noch geschlossen – genauer im Zustand MOUNT
- Öffnen der PDBs
  - bei Zugriff auf die CDB

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

```
ALTER PLUGGABLE DATABASE pdb1_name, pdb2_name OPEN;
```

  - bei Zugriff auf jeweilige PDB

```
ALTER DATABASE OPEN;
```

oder

```
STARTUP
```

Verwendbarkeit der bekannten Optionen

Schließen der PDBs

- Herunterfahren der CDB$Root schließt automatisch alle PDBs
- Schließen der PDBs
  - bei Zugriff auf die CDB

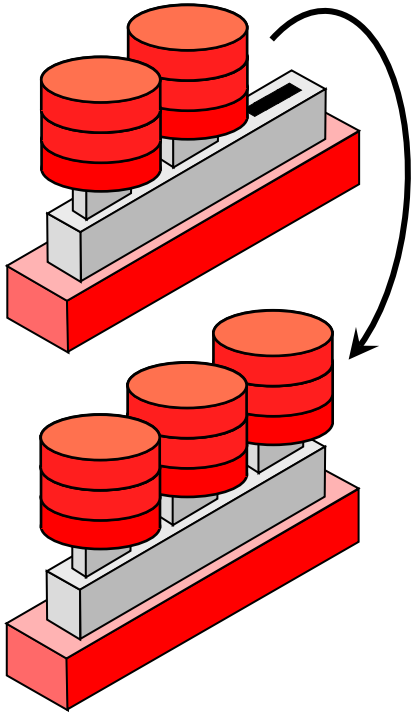
```
ALTER PLUGGABLE DATABASE ALL CLOSE;
```

```
ALTER PLUGGABLE DATABASE pdb1_name, pdb2_name CLOSE;
```

  - bei Zugriff auf jeweilige PDB
    NUR mit SQL*Plus-Kommando
    einschließlich bekannter Optionen:

```
SHUTDOWN
```

Unplug / Plug der PDBs – Konzept



- Verschieben zwischen CDBs erfordert lediglich das Verschieben der Metadaten der PDB
- Transport der Datafiles als „Gesamtdatenbank“ einschließlich Data Dictionary im Tablespace SYSTEM
- eine „ausgeklinkte” PDB beinhaltet Infos zu lineage, opatch, encryption keys etc.

Unplug / Plug der PDBs mit SQL

- SQL
  - Unplug nach Herunterfahren der PDB:

```
ALTER PLUGGABLE DATABASE pdb
UNPLUG INTO 'path/pdb.xml';
```

  - Plug nach Kopieren der Data Files auf Zielsystem und Test auf Kompatiblität (bei späteren Releases):

```
CREATE PLUGGABLE DATABASE pdb2
USING 'path/pdb.xml' NOCOPY TEMPFILE REUSE;
```

    eingeklinkte PDB ist im MOUNT-Zustand und muss noch geöffnet werden,
    Backup erforderlich, sonst nicht recoverbar!

Unplug / Plug der PDBs mit DBCA

- Varianten
  - integrierbares Datenbankarchiv generieren
    - entstehen Metadatendatei *.xml und komprimierte Archivdatei *.pdb
    - nicht unterstützt, wenn PDB ASM zur Speicherung verwendet
  - integrierbares Datenbankdateiset generieren
    - entstehen Metadatendatei *.xml und Backupdatei *.dfb
    - empfohlen, wenn Quell- und Ziel-CDB ASM verwenden
- Vorteile gegenüber SQL:
  - automatisches Shutdown und Startup der PDBs
  - geringere Größe der Archiv- bzw. Backupdateien
  - Test beim Plug

Erstellung einer Pluggable DB

- Varianten:
  - aus Seed Database (einschließlich **ADMIN USER** mit Rolle **PDB_DBA**)
  - durch Klonen aus vorhandener PDB
- Werkzeuge:
  - DBCA
  - SQL

```
CREATE PLUGGABLE DATABASE pdbname
ADMIN USER syspdb IDENTIFIED BY password
[ROLES = (role_list)]              -- zusätzl. Rolle(n)
FILE_NAME_CONVERT =                -- nicht
 ('source_path','target_path') -- bei OMF
[DEFAULT TABLESPACE data_ts DATAFILE 'path\filename'
SIZE size] [options];
```

Verwalten von geteilten Ressourcen

- PDBs konkurrieren um gemeinsame Ressourcen
- Resource Management in Multitenant Umgebung - Einstellung mit Resource Manager pro PDB:
  - CPU
  - Exadata I/O
  - Sessions
  - Anzahl der „Parallel Execution Servers”
- konfigurierte Policies regeln, wie Ressourcen benutzt werden:
  - …über eine Default Konfiguration, die auch bei Hinzufügen/Entfernen von PDBs funktioniert
  - …über harte Limits

Verteilen von Ressourcen zwischen PDBs

- Konzept
  - eine Anzahl „Shares” wird jeder PDB zugewiesen
    (Default: 1 Share pro PDB)
  - die Summe der Shares ist beliebig
  - die Summe der Shares entspricht der Gesamtleistung
  - zusätzlich:
    ein optionales „Cap”
    (d.h. ein maximales Nutzungslimit)
    kann jeder PDB zugewiesen werden
    (Default: Kein Cap, d.h. max. 100%)

Ressourcenzuteilung mit Limit

- Einfaches Beispiel für Shares mit optionales „Cap“:

Pluggable Database	Shares	Garantierte CPU-Leistung	Maximale CPU-Leistung
A	2	2/4 = 50%	80%
B	1	1/4 = 25%	70%
C	1	1/4 = 25%	30%

- Einstellung mit Package **DBMS_RESOURCE_MANAGER**

Änderungen der DD-Views

- Data Dictionary Views
  - Erweiterung um eine View-Ebene
    - in CDB Sicht über alle PDBs hinweg
    - Präfix **CDB_**
    - zusätzliche Spalte **CON_ID** (Container-ID)
  - Beispiel:

```
SELECT username, con_id
FROM   cdb_users
ORDER BY username, con_id;
```

  - *Hinweis*:
    Anzeige in PDB nur für jeweilige **CON_ID** (nur lokale Daten)

Views **V$PDBS** und **V$CONTAINERS**

- Spezielle dyn. Performance View für PDBs: **V$PDBS**
  - in der CDB (Root) zeigt diese View alle PDBs an, die gerade in einer CDB betrieben werden
  - in einer PDB zeigt diese View Informationen über die kontaktierte PDB an
  - Beispiel:

```
SELECT   con_id, name, open_mode
       , restricted, open_time, total_size
FROM   v$pdbs;
```

- View **V$CONTAINERS** zeigt darüber hinaus auch die Root an (nur bei Kontakt zur CDB)

Systemparameter

- CDB-weit geltende und PDB-bezogene Parameter
  - neben den global für die CDB einzustellenden Systemparametern gibt es eigene Systemparameter, die pro PDB eingestellt werden können.
  - existiert nur 1 Parameterdatei
  - 2 neuen Spalten in Views **V$SYSTEM_PARAMETER** und **V$PARAMETER** :
    - **ISPDB_MODIFIABLE**
      CBD: **true** oder **false**
      non-CDB: **null**
    - **CON_ID**

Globale Systemparameter

- globale (in CDB$ROOT und PDBs) geltende Systemparameter
  - Ermittlung mit:

```
SELECT name FROM v$system_parameter
WHERE  ispdb_modifiable = 'FALSE' ;
```

  - wesentliche Kategorien und Parameter:
    - Memory, Undo, Auditing,
      db_block_size, **processes**
      etc.
  - Modifikation nur in CDB möglich

PDB-Systemparameter

- in PDBs modifizierbare Systemparameter
  - Ermittlung mit:

```
SELECT name FROM v$system_parameter
WHERE  ispdb_modifiable = 'TRUE' ;
```

  - wesentliche Kategorien und Parameter:
    - NLS, Optimizer, Statistikeinstellungen, **`sql_trace`**, **`workarea_size_policy`** etc.
  - Modifikation in PDB möglich
  - falls Parameter nicht spezifisch für eine PDB gesetzt, gilt der Wert aus der CDB
  - bei „Unplug/Plug“ werden abweichende Werte mit Metadaten in eine andere CDB übernommen

Erstellung der Benutzer in Multitenant Architecture

- *Lokale Nutzer*
  - nur im Kontext der PDBs zu erstellen
  - Namen dürfen nicht mit “C##“ oder “c##“ beginnen
- *Globaler Nutzer*
  - nur im Kontext der „root“ erstellbar
  - Namen müssen mit “C##“ oder “c##“ beginnen
  - können in den verschiedenen PDBs unterschiedliche Privilegien bekommen
  - Passwort eines globalen Datenbankbenutzers ist aber sowohl in der CDB als auch in allen PDBs gleich.

Merkmale der neuen vordefinierten Benutzer in Multitenant Architecture

- neue in 12c vordefinierte Benutzer für wichtige Verwaltungstätigkeiten ohne Privileg **SYSDBA**
  - **SYSBACKUP**, **SYSDG**, **SYSKM**, **SYSAUD**
  - verhalten sich wie globale Benutzer (in jeder PDB separate Einstellungen möglich)
  - dürfen auch im Kontext CDB nicht gelöscht werden
- neue View: **CDB_USERS**
  mit Spalten **CON_ID** und **COMMON** (Wert YES)

SEMINARE

4

Backup und Recovery der Multitenant Architecture

MD CONSULTING & INFORMATIONSDIENSTE GMBH

RMAN und Multitenant Architecture

- Unterstützung von Multitenant:
  - Container Databases inklusive PDBs
  - Pluggable Databases
- Anzeigen und mögliche Aktivitäten in Abhängigkeit des jeweiligen Kontextes bei:
  - **REPORT**
  - **LIST**
  - **BACKUP**
  - **RECOVERY**
  - **DELETE**

Report

- Beispiel **REPORT SCHEMA**
  - Connect zur CDB-Root

kontext-bezogen

```
List of Permanent Datafiles
===========================
File Size(MB) Tablespace
---- -------- ------------
1    770      SYSTEM
2    250      PDB$SEED:SYSTEM
3    720      SYSAUX
4    610      PDB$SEED:SYSAUX
5    725      UNDOTBS1
6    5        USERS
7    260      STPDB:SYSTEM
8    630      STPDB:SYSAUX
9    5        STPDB:USERS
```

Beachte: Kennzeichnung

  - Connect zur PDB
    nur Sicht auf Files der PDB “STPDB“

```
List of Permanent Datafiles
===========================
File Size(MB) Tablespace
---- -------- ------------
7    260      SYSTEM
8    630      SYSAUX
9    5        USERS
```

Backup – Connect zur PDB – 1

- Beispiele Befehl **BACKUP** bei Connect zur PDB
  - Backup der gesamten pluggable Database

```
BACKUP AS BACKUPSET DATABASE;
```

  - Backup Tablespaces der pluggable Database

```
BACKUP AS COPY TABLESPACE users;
```

  - Backup des Control File

```
BACKUP AS BACKUPSET CURRENT CONTROLFILE;
```

wie bisher

Backup – Connect zur PDB – 2

- Backup von archivierten Redo Log Files

```
BACKUP AS BACKUPSET ARCHIVELOG ALL;
```

kein Abbruch, Meldung ohne Backupausführung:

```
specification does not match any archived log in the repository
backup cancelled because there are no files to backup
```

- Backup Database einschließlich archivierter Redo Log Files

```
BACKUP AS BACKUPSET DATABASE PLUS ARCHIVELOG;
```

Backup erfolgreich, aber nur für Datafiles
Meldung:

```
specification does not match any archived log in the repository
backup cancelled because there are no files to backup
```

Schlussfolgerung:
Online Backup auf PDB allein relativ sinnlos

Backup – Connect zur CDB-Root

- Beispiele Befehl **BACKUP** bei Connect zur CDB-Root
  - Backup der Root Database

```
BACKUP AS BACKUPSET DATABASE ROOT ;
```

  - Backup einer pluggable DB

```
BACKUP AS BACKUPSET PLUGGABLE DATABASE pdb1 ;
```

  - Backup Tablespaces von pluggable DBs

```
BACKUP AS COPY TABLESPACE pdb1:users, pdb2:users;
```

  - Backup Tablespaces der Root Database

```
BACKUP AS COPY TABLESPACE users;
```

Restore und Recovery – Connect zur PDB

- Beispiele Befehle **RESTORE**, **RECOVER** bei Connect zur PDB
  - Recovery der gesamten pluggable Database

```
RESTORE DATABASE;
RECOVER DATABASE;
ALTER DATABASE OPEN;
```

  - Recovery der einzelner Dateien der pluggable Database

```
ALTER DATAFILE 20,21 OFFLINE;
RESTORE DATAFILE 20,21;
# oder RESTORE TABLESPACE daten, indexes;
RECOVER DATAFILE 20,21;
# oder RECOVER TABLESPACE daten, indexes;
ALTER DATAFILE 20,21 ONLINE;
```

wie bisher

Restore und Recovery – Connect zur CDB-Root – 1

- Beispiele Befehle **RESTORE**, **RECOVER** bei Connect zur CDB-Root
  - Recovery einer gesamten pluggable Database

```
ALTER PLUGGABLE DATABASE stpdb CLOSE;
RESTORE PLUGGABLE DATABASE stpdb ;
RECOVER PLUGGABLE DATABASE stpdb ;
ALTER PLUGGABLE DATABASE stpdb OPEN;
```

  - Recovery der CDB-Root ohne PDBs

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
RESTORE DATABASE ROOT ;
RECOVER DATABASE ROOT ;
ALTER DATABASE OPEN;
ALTER PLUGGABLE DATABASE ALL OPEN;
```

Restore und Recovery – Connect zur CDB-Root – 2

- Recovery der gesamten CDB einschließlich PDBs

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
RESTORE DATABASE;
RECOVER DATABASE;
ALTER DATABASE OPEN;
ALTER PLUGGABLE DATABASE ALL OPEN;
```

Anmerkung:
kein Öffnen der Seed Database

SEMINARE

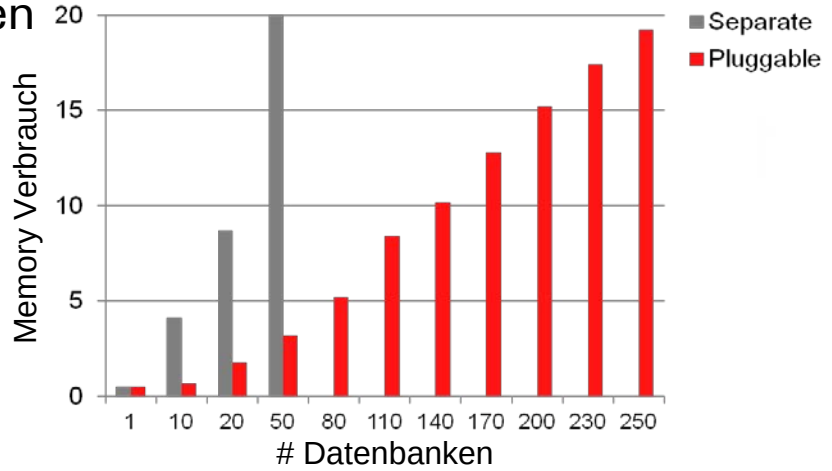
5

Demonstration Multitenant Architecture und Fazit

MD CONSULTING & INFORMATIONSDIENSTE GMBH

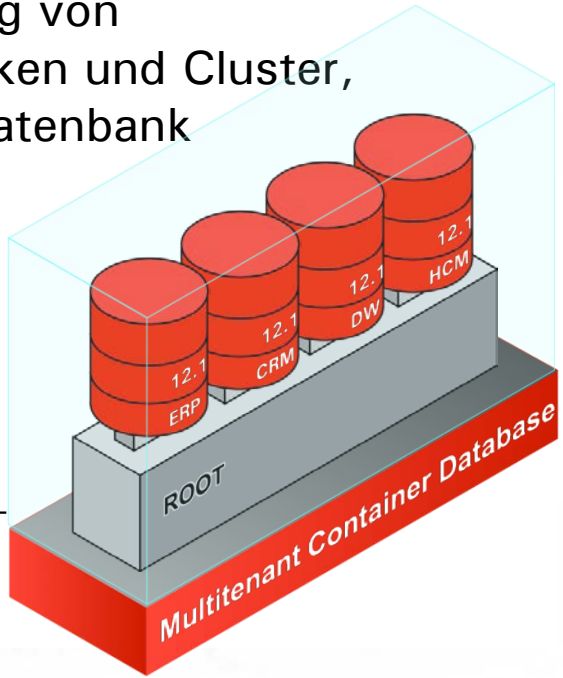
Fazit –
Gründe für die Oracle Multitenant Architecture

- **Resourcenschonung und Skalierbarkeit**
  Mehrere pluggable Databases benötigen im OLTP deutlich weniger Resourcen als separate Databases
- **Pluggable Databases**
  Einfaches Ausklinken (Unplug) und Einklinken (Plug) in andere Datenbankumgebungen
- **Keine Änderung von Applikationen**
  Völlige Transparenz für bestehende Applikationen – jede pluggable Databases verhält sich für die Anwendung wie eine separat betriebene Datenbank



Fazit – Gründe für die Oracle Multitenant Architecture

- „Manage Many as One" mit Multitenant
  Gemeinsames DB-Backup, gemeinsame Nutzung von Verfügbarkeitslösungen wie Standby-Datenbanken und Cluster, vereinfachtes Patching für gesamte Containerdatenbank
- Multitenant für Test und Entwicklung
  schnelle, flexible Kopien und Snapshots von Pluggable Databases
- Ideal für Independent Software Vendors (ISVs)
  Anstelle von langwierigen Setup-Skripten u. ä – Verteilung von vorkonfigurierten Anwendungen als PDB



Abschluss

- Abschlussbesprechung
- Urkundenausgabe
- Beurteilung des Workshop