Oracle Database 12c Release 2 - Application Container Mandantenfähigkeit für Applikationen

Markus Flechtner
Trivadis GmbH
Düsseldorf

Schlüsselworte

Oracle 12.2, Multitenant, Application Container, SaaS

Einleitung

Mit den „Application Containern“ in der Version 12.2 der Oracle-Datenbank ermöglicht es Oracle den Kunden, die Prinzipien der mit Version 12.1 eingeführten „Container-Datenbanken“ auch für eigene Applikationen zu nutzen. Anwendungen werden dadurch „mandantenfähig“. Dies bringt neue Möglichkeiten für „Software as a Service“ (SaaS), wie z.B. vereinfachte Software-Installation und –Updates.

Container Datenbanken in Oracle Database 12c

Um die Prinzipien zu verstehen, schauen wir erst einmal kurz zurück auf die mit Oracle 12c Release 1 eingeführte neue Architektur der „Container-Datenbanken“:

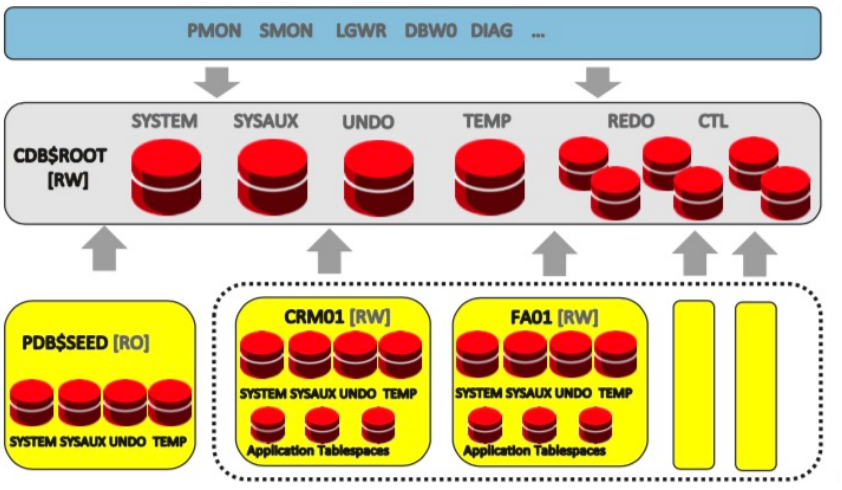

Abb. 1: Container-Datenbank-Architektur im Überblick

Es gibt weiterhin – den Real Application Cluster einmal außer acht gelassen – eine Instanz mit einer Menge von Prozessen und einer SGA. In einer Container-Datenbank (CDB) liegen die Definitionen

der Data Dictionary-Objekte, Oracle-PL/SQL-Packages wie z.B. DBMS_SQL etc. im Root-Container CDB$ROOT. Dort liegen aber keinerlei Anwendungsdaten oder Informationen zu den Strukturen der Anwendungsobjekte. Anwendungsdaten liegen in den Pluggable Datenbanken (PDB). Neben den Anwendungsdaten liegen dort auch die Definitionen der Anwendungstabellen. Die Data-Dictionary-Objekte in den PDBs übernehmen die Data-Dictionary-Definitionen aus der CDB$ROOT und speichern die Definitionen lokal. Dieses Prinzip nennt Oracle „Sharing“.

```
SQL> SELECT sharing, count(*) FROM dba_object GROUP BY sharing;
SHARING                  COUNT(*)
------------------       ----------
METADATA LINK                 66457
DATA LINK                       214
EXTENDED DATA LINK               56
NONE                           5053
```

Application-Container ermöglichen es in Oracle 12.2, dieses Sharing-Prinzip für eigene Anwendungen zu nutzen.

Application-Container im Überblick

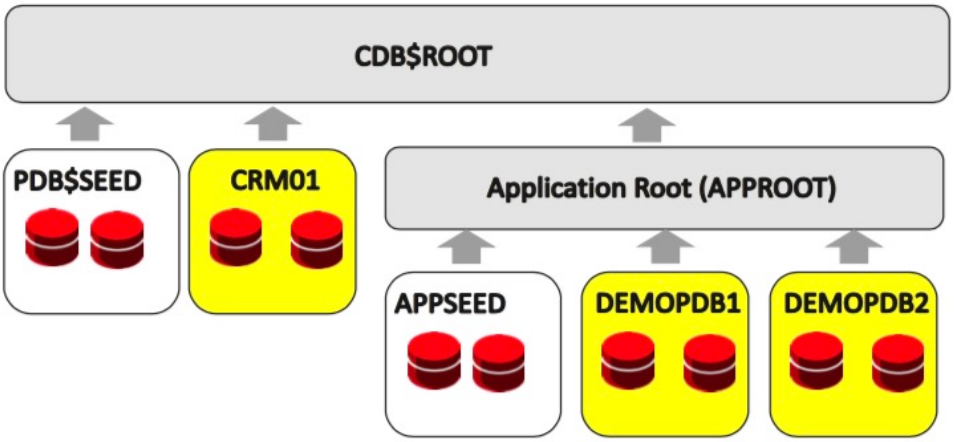

Abb. 2: Application Container im Überblick

Application Container sind technisch Pluggable Datenbanken, die aber eine eigene Unterstruktur innerhalb einer CDB bilden. Sie bestehen aus

- Application Root
- Application Seed (optional, vergleichbar mit der PDB$SEED)
- Einer oder mehrerer Applikations-Datenbanken

Vom Application Root aus wird der Application Container administriert. Dort liegen auch die Definitionen von gemeinsam genutzten Datenstrukturen (Tabellen etc.) und auch gemeinsam genutzte Daten (z.B. zentrale Stammdaten).

In einer CDB können mehrere Application Container liegen; die Applikationen können dabei in mehreren Versionen vorliegen; auch können in einem Application Container mehrere Applikationen abgelegt sein. Dies kann ggf. sinnvoll sein, wenn eine Applikation modular aufgebaut ist.

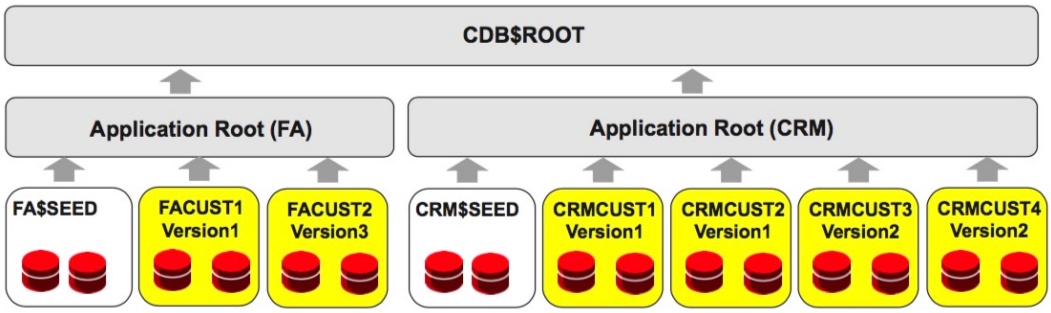

Abb. 3: mehrere Application Container in einer CDB

Application Common Objects

Gemeinsam genutzte Objekte nennt Oracle “Application Common Objects”. Dafür gibt es folgende Varianten:

Sharing-Attribut	Definition des Objektes	Speicherort der Daten	Anmerkung
METADATA	Application Root	Application PDB	
DATA	Application Root	Application Root	Daten sind für die Applikations-PDBS “read-only”
EXTENDED DATA	Application Root	Application Root und Application PDB	Daten die im Application Root abgelegt sind, können in allen Applikations-PDBs gelesen werden. Zusätzlich können in den Applikations-PDBS “lokale” Daten abgelegt werden
NONE	Application Root	Application Root und Application PDB	Die Daten sind jeweils “lokal”.

Welche Art des “Sharings” genutzt wird, wird im CREATE-Befehl angegeben:

```
create table PLZ_DATA sharing=EXTENDED DATA ..
```

Lt. Oracle-Dokumentation (Database Administrators Guide / Chapter 44) können folgende Typen "Application Common Objects" sein:

- Analytic views
- Attribute dimensions
- Directories
- External procedure libraries
- Hierarchies
- Java classes, Java resources, Java sources
- Packages
- Object tables
- Object types
- Object views
- Sequences
- Stored functions
- Stored procedures
- Synonyms
- Tables (including temporary tables)
- Triggers
- Views

"METADATA" und "NONE" sind für alle Typen erlaubt; "DATA LINK" bei Sequenzen erlaubt übergreifende Sequenzen z.B. für eine zentrale Vergabe von Kundennummern. Bei Tabellen und Views sind "DATA LINK" und "EXTENDED DATA LINK" erlaubt, so dass eine gemeinsame Nutzung von Daten möglich ist.

Application Container Objects erlauben auch Fremdschlüssel zwischen "Data Linked"- und "Metadata"-Shared objects, z.B.

```
create table DEPT SHARING=DATA
(
DEPTNO number constraint PK_DEPT primary key,
DNAME  varchar2(30)
);

create table EMP SHARING=METADATA
(EMPNO number constraint PK_EMP primary key,
 ENAME  varchar2(30) not null,
 DEPTNO constraint FK_DEPT references DEPT);
```

Achtung: damit dieses Konstrukt funktioniert, muss der Patch zu Bug 21955394 (CDB:ORA-02291 WHEN FOREIGN KEY REFERS TO THE PRIMARY KEY IN DATA LINK) installiert sein. Dieser Patch ist für die Basis-Version 12.2.0.1 und für das Release-Update 12.2.0.1.170718 verfügbar (Stand: Ende September 2017).

Beim DML auf "Application Common Tables" sind einige Dinge zu beachten:

1. Der Versuch, eine Data Linked-Tabelle von einer Applikations-PDB heraus zu ändern führt zu einer Fehlermeldung: ORA-65097: DML into a data link table is outside an application action.

2. Aber der Versuch, zentral (im Application Root) gespeicherte Daten einer Tabelle mit "Extended Data"-Sharing von einer Applikations-PDB heraus zu ändern, führt nicht zu einer Fehlermeldung. Ein Update ist erfolgreich – allerdings mit der Meldung "0 rows updated".

Applikation installieren, upgraden und Patchen

Im ersten Schritt muss ein Applikations-Container angelegt werden:

```
create pluggable database DEMO AS APPLICATION CONTAINER
admin user admin identified by manager;
```

Damit ist der "Application-Root-Container" angelegt. Sämtliche weitere Verwaltung des Applikations-Containers erfolgt aus diesem Root-Container heraus.

Wichtig: für Applikations-Container ist die Verwendung von Oracle Managed Files (OMF) erforderlich! Dies ist leider nicht dokumentiert.

Danach wird die Applikation im Application-Root angelegt:

```
alter pluggable database application DEMOAPP begin install '1.0';
```

Mit diesem "Begin"-Befehl startet Oracle eine interne Aufzeichnung aller folgenden Befehle, die zum Anlegen der Applikation erforderlich sind. Beim späteren Anlegen einer Applikations-PDB werden diese Befehle – unter Beachtung des Sharing-Attributes – in der Applikations-PDB wiederholt.

Danach können Tablespaces, Benutzer und Objekte für die Applikation angelegt werden.

Die Installation wird mit dem Befehl

```
alter pluggable database application DEMOAPP end install '1.0';
```

abgeschlossen. Die Aufzeichnung der Befehle wird gestoppt.

Wie kommt man vom Application Root zur Application PDB?

Dazu gibt es zwei Möglichkeiten:

1. Es wird eine "Application-Seed"-PDB angelegt, die als Vorlage für die Applikations-PDBS dient.
2. Die Applikations-PDB wird direkt aus der Application-Root heraus angelegt.

Application PDB anlegen via Application Seed

Die Seed-Datenbank wird mit folgendem Befehl angelegt:

```
CREATE PLUGGABLE DATABASE AS SEED ADMIN USER app_admin IDENTIFIED BY
manager;
```

Der Name der Seed-Datenbank wird automatisch vergeben (Name des Applikations-Containers+ "$SEED", d.h. in obigem Beispiel "DEMO$SEED".

Danach wird die Applikation in der neuen Seed-Datenbank angelegt. Dabei sorgt der “SYNC”-BEfehl dafür, dass die Applikationsobjekte angelegt werden:

```
ALTER PLUGGABLE DATABASE DEMO$SEED OPEN;
ALTER SESSION SET CONTAINER=DEMO$SEED;
ALTER PLUGGABLE DATABASE APPLICATION DEMO SYNC;
ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE OPEN READ ONLY;
```

Abschliessend wird die neue Applikations-PDB angelegt:

```
create pluggable database DEMOPDB1 from DEMO$SEED;
alter pluggable database DEMOPDB1 open;
```

Applikations-PDB aus dem Application-Root heraus anlegen

Dieser Fall verläuft ähnlich wie das Anlegen der Application-Seed-Datenbank:

```
ALTER SESSION SET CONTAINER=DEMO;
create pluggable database DEMOPDB1 admin user admin identified by admin;
alter pluggable database DEMOPDB1 open;
ALTER SESSION SET CONTAINER=DEMOPDB1;
Alter pluggable database application DEMOAPP sync;
```

Aktualisieren und Patchen von Applikationen

Applikations-Upgrades und –Patches verlaufen ähnlich wie die Installation:

1. Die Aktion wird gestartet
2. Die erforderlichen Befehle werden im Root ausgeführt
3. Die Aktion wird beendet. Damit wird die Aufzeichnung der Befehle beendet.
4. Wenn eine Applikations-PDB aktualisiert werden soll, werden die Objekte mit dem SYNC-Befehl aktualisiert.

Aktualisierung der Applikation im Root:

```
alter pluggable database application DEMOAPP  begin upgrade from '1.0'  to
'2.0';
[.. Objekte aktualisieren .. ]
alter pluggable database application DEMOAPP end upgrade to '2.0';
```

Damit die alte Version auch nach dem Upgrade des Application-Root für die Applikations-PDBs zur Verfügung steht, wird beim Upgrade automatisch eine Kopie des Root-Containers mit der alten Version erzeugt.

Upgrade der Applikations-PDB (in der Applikations-PDB):

```
Alter pluggable database application DEMOAPP sync;
```

Beim Patchen lauten die Befehle:

```
alter pluggable database application DEMOAPP  begin patch <Patch#>;
```

bzw.

```
alter pluggable database application DEMOAPP end patch <Patch#>;
```

Im Unterschied zum Upgrade stehen beim Patchen nicht alle Befehle zur Verfügung. So ist z.B. ein „ALTER TABLE“ nicht erlaubt. Auch sind sämtliche „DROP“-Befehle verboten.

Auch die De-Installation einer Applikation erfolgt nach dem obigen Prinzip: Befehle im Root ausführen und aufzeichnen („BEGIN UNINSTALL“, „END UNINSTALL“) und dann in einer Applikations-PDB wiederholen lassen.

Administration von Application Containern

Wie bereits erwähnt, wird ein Application Container vom Application Root aus administriert. Von dort aus werden Applikations-PDBs angelegt, geöffnet, geschlossen und gelöscht. Solange es noch eine Applikations-PDB gibt kann ein Application-Root nicht gelöscht werden.

Die zentrale Bedeutung des Application-Root zeigt sich z.B. beim Backup & Recovery:

Wenn man am Application-Root angemeldet ist, dann gibt es im RMAN folgende Backup-Möglichkeiten:

```
RMAN> BACKUP DATABASE ROOT;                # backup application root
RMAN> BACKUP PLUGGABLE DATABASE DEMOPDB1; # backup application PDB
RMAN> BACKUP DATABASE;                     # backup application root + PDBs
```

Beim Backup ist ein Application-Container somit eine Zusammenfassung von PDBs die gemeinsam gesichert werden kann. Die gleichen Prinzipien gelten auch für das Recovery.

Natürlich werden Applikationscontainer auch gesichert, wenn mittels „BACKUP DATABASE;“ von der CDB$ROOT aus eine Komplettsicherung der CDB durchgeführt wird.

Ausführungspläne in Application Containern

Bei Abfragen, die auf Daten in einer Applikations-PDB und im Application-Root zugreifen stellt sich natürlich die Frage, wie dies in den Ausführungsplänen dargestellt wird. Dazu beispielhaft hier zwei Ausführungspläne:

1. Zugriff von einer Application-PDB auf eine Data-Linked-Table im Application Root (DEPT) und eine lokale Tabelle (EMP):

```
select  e.empno, e.ename, d.deptno, d.dname
  2  from scott.emp e  -- local in application PDB
  3       scott.dept d -- from application root (data linked)
  4  where d.deptno=e.deptno;

-------------------------------------------------------------------
 | Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  1    |    35 |      3  (0)|
|*  1 |  HASH JOIN         |      |  1    |    35 |      3  (0)|
|   2 |   DATA LINK FULL   | DEPT |  1    |    22 |            |
|   3 |   TABLE ACCESS FULL| EMP  |    14 |   182 |  3  (0)    |
-------------------------------------------------------------------
```

Die Operation “DATA LINK FULL” zeigt den Zugriff auf das Objekt im Application Root an, in diesem Fall ein Full Table Scan.

2. SELECT auf eine EXTENDED DATA-Table

```
select * from scott.zip_codes;
-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name     | Pstart| Pstop |    TQ |IN-OUT|PQ Distrib
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |          |       |       |       |      |
|   1 |  PX COORDINATOR              |          |       |       |       |      |
|   2 |   PX SEND QC (RANDOM)        | :TQ10000 |       |       | Q1,00| P->S  |QC(RAND)
|   3 |    PX PARTITION LIST ALL     |          |     1 |     2 | Q1,00| PCWC  |
|   4 |     EXTENDED DATA LINK FULL  | ZIP_CODES|       |       | Q1,00| PCWP  |
-----------------------------------------------------------------------------------------------
```

Der Ausführungsplan zeigt, dass Oracle derartige Tabellen anscheinend intern wie partitionierte Tabellen (1 Partition im Application Root, 1 Partition in der jeweiligen Applikations-PDB) behandelt. Sie werden allerdings nicht im Data Dictionary als partitionierte Tabellen angezeigt.

Ausblick

Dies ist nur eine Einführung in dieses neue Feature. Application Container bieten darüberhinaus weitere Möglichkeiten, wie

- Konvertierung einer „normalen“ PDB zu einer Application-Root oder eine Application-PDB
- Container-Maps, die eine „Weiterleitung“ von Befehlen vom Application-Root in eine Applikations-PDB erlauben
- Application Common Roles, Users und Profile – vergleichbar ihren Äquivalenten auf CDB-Ebene

Fazit

Application Container sind ein sehr interessantes Feature für Software-as-a-Service-Angebote in der Cloud. Installation, Upgrade und Patching von Applikationen sind einfach und können zentral gesteuert werden. Als “Version 1” eines Features hat diese Funktionalität viel Potential, das sich in Zukunft sicher entfalten wird.

Weitere Informationen

- Oracle 12.2 Concepts (Kapitel 19)
- Oracle 12.2 Administrators Guide (Kapitel 40 und 44)

Kontaktadresse:

Markus Flechtner

Trivadis GmbH

Werdener Straße 4

D-40227 Düsseldorf

Telefon: +49 (0) 211 – 5866 6470

Fax: +49 (0) 211 – 5866 6471

E-Mail markus.flechtner@trivadis.com

Internet: www.trivadis.com