Multitenant für Einsteiger

CarajanDB GmbH

… über mich

Oracle ACE Pro

- Oracle Spezialist seit 1992
  - 1992: Presales bei Oracle in Düsseldorf
  - 1999: Projektleiter bei Herrmann & Lenz Services GmbH
  - 2005: Technischer Direktor ADM Presales bei Quest Software GmbH
  - 2011: Geschäftsführer CarajanDB GmbH
- 2011 → Ernennung zum Oracle ACE
- Autor der Bücher:
  - Oracle9i für den DBA, Oracle10g für den DBA, Oracle 11g Release 2 für den DBA
- DOAG Themenverantwortlicher Datenbankadministration, Standard Edition
- Hobbies:
  - Drachen steigen lassen (Kiting) draußen wie drinnen (Indoorkiting)
  - Motorradfahren (nur draußen)
  - Bier Brauen
  - Singen (überall)

CarajanDB

- Experten mit über 30 Jahren Datenbank Erfahrung
- Spezialisten für
  - Datenbank Administration (Oracle und PostgreSQL)
  - Hochverfügbarkeit (RAC, Data Guard, Replication, etc.)
  - Migrationen (Unicode, PostgreSQL)
  - Performance Optimierung
  - Monitoring (OEM, Foglight, CheckMK, PEM)
- Fernwartung
- Schulung und Workshops
  - PostgreSQL
  - Oracle Multitenant
  - Toad

CARAJANDB

Multitenant

Definition

Multi-Tenancy = Mehrmandantenfähigkeit

„Als mandantenfähig (auch mandantentauglich) wird Informationstechnik bezeichnet, die auf demselben Server oder demselben Software-System mehrere Mandanten, also Kunden oder Auftraggeber, bedienen kann, ohne dass diese gegenseitigen Einblick in ihre **Daten**, **Benutzerverwaltung** und Ähnliches haben.

Ein IT-System, das dieser Eigenschaft genügt, bietet die Möglichkeit der disjunkten, mandantenorientierten Datenhaltung, Präsentation (GUI) und Konfiguration (Customizing). Jeder Kunde kann nur seine Daten sehen und ändern. Ein System wird nicht mandantenfähig, indem man für jeden Mandanten eine eigene Instanz (Kopie) des Systems erstellt.

Quelle: Wikipedia, 2023-05-22

Vergleich

Mandantentrennung	Vorteil	Nachteil	
Nummernkreise	Einfach zu implementieren	Sehr fehleranfällig Unsicher	☹ ☹ ☹
Zusätzliche Spalte	Bessere Trennung der Mandanten	Aufwändige nachträgliche Implementierung Unsicher	☹ ☹
Eigene Schemata	Einfach zu implementieren Gute Trennung der Mandanten	Keine Public-Objekte Keine getrennte Administration	☹
Eigene Datenbank	Einfach zu implementieren Sehr gute Trennung der Mandanten Getrennte Administration	Hoher administrativer Aufwand. Hoher Ressourcenverbrauch	☺
Pluggable Database	Einfach zu implementieren Sehr gute Trennung der Mandanten Getrennte Administration	Nicht in jeder Datenbank möglich Schulungsaufwand für Administratoren	☺ ☺ ☺

Historie

- 12.1.0.1 Eine neue Datenbankarchitektur hält Einzug: Multitenant
  Besser nicht einsetzen!
- 12.1.0.2 Long Term Support Release
  Multitenant brauchbar, erste Projekte
- 12.2.0.1 Flashback Pluggable Database möglich
  viele neue Optionen für PDBs (Proxy PDB, Application Container, …)
- 18c PDB Lockdown Profiles, Refreshable PDB Switchover
- 19c Drei Pluggable Databases in allen Editionen frei
- 21c Nur noch Multitenant möglich (Desupport NON-CDB)

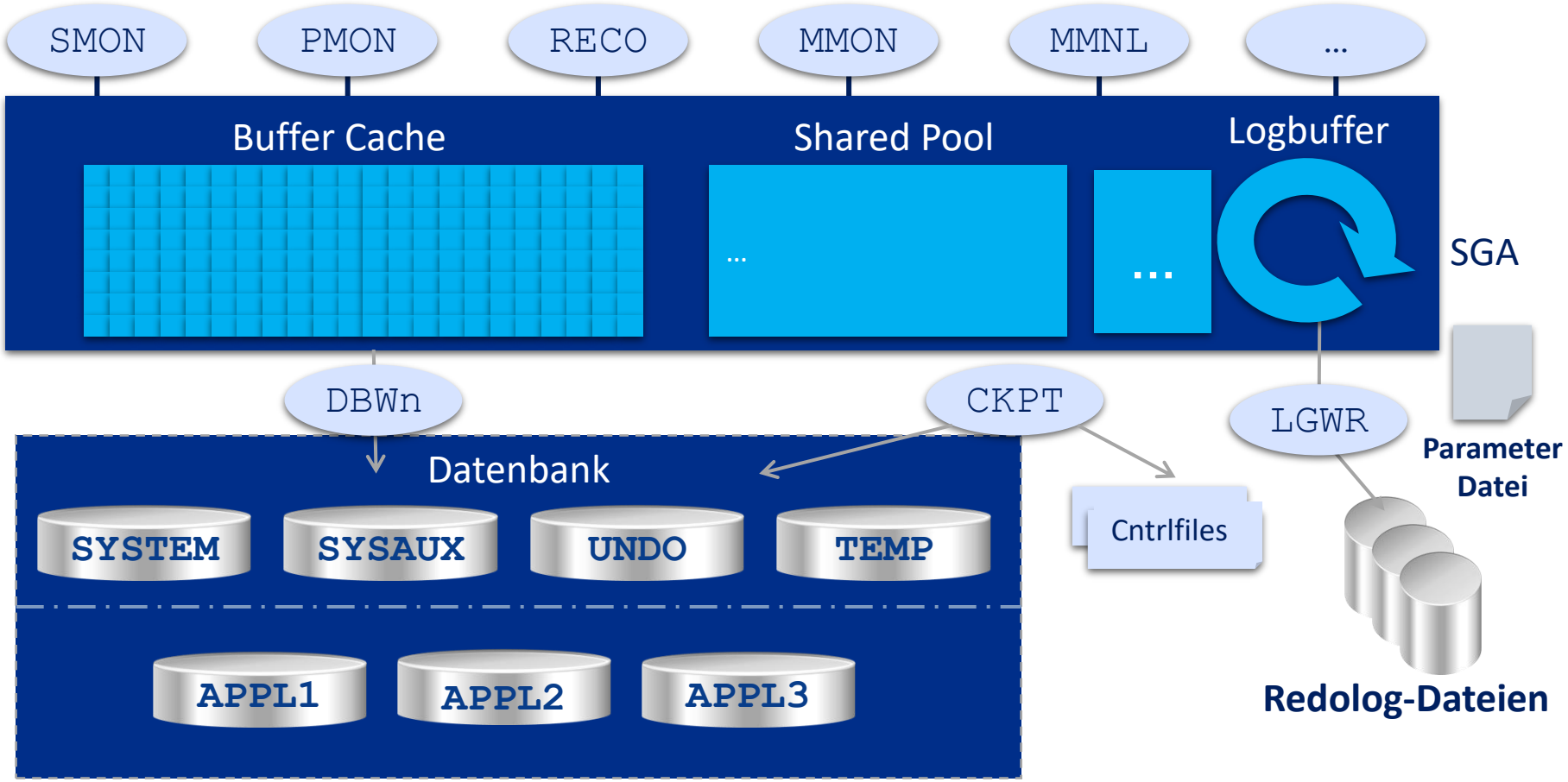
Überblick

- NON-CDB
  - Architektur bis Oracle 11.2
- Multitenant Architektur
  - Alternative Architektur ab Version 12.1.1 für alle Editionen
  - Einzig mögliche Architektur ab Version 21c
- Multitenant Option
  - Kostenpflichtig für die Enterprise Edition
  - Bis zu 252 PDBs pro CDB (Exadata ab 12.2 4096 PDBs)

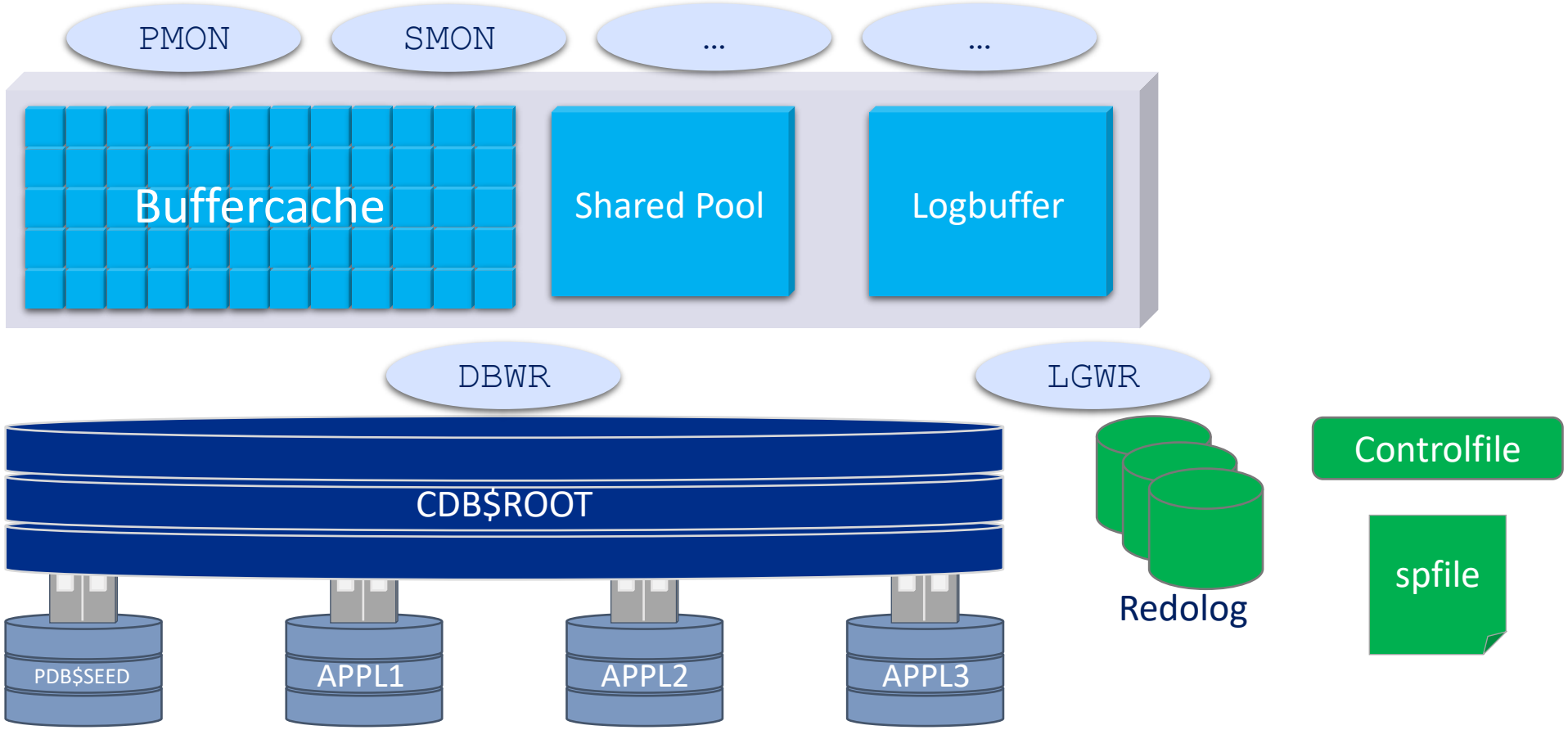
Lizenzierung

Release Availability	✗ 11.2 ✓ 12.1 ✓ 12.2 ✓ 18c ✓ 19c ✓ 21c
Licensed With	**Oracle Multitenant**
Available On	✓ Express Edition ✓ Enterprise Edition ✓ Oracle Database Appliance ✓ Exadata ✓ Exadata Cloud Service / Cloud@Customer ✓ Database Cloud Service Enterprise Edition - High Performance ✓ Database Cloud Service Enterprise Edition - Extreme Performance **Notes:** For all offerings using Oracle Database 19c or later, if you are not licensed for Oracle Multitenant, then you may have up to 3 user-created PDBs in a given container database at any time.

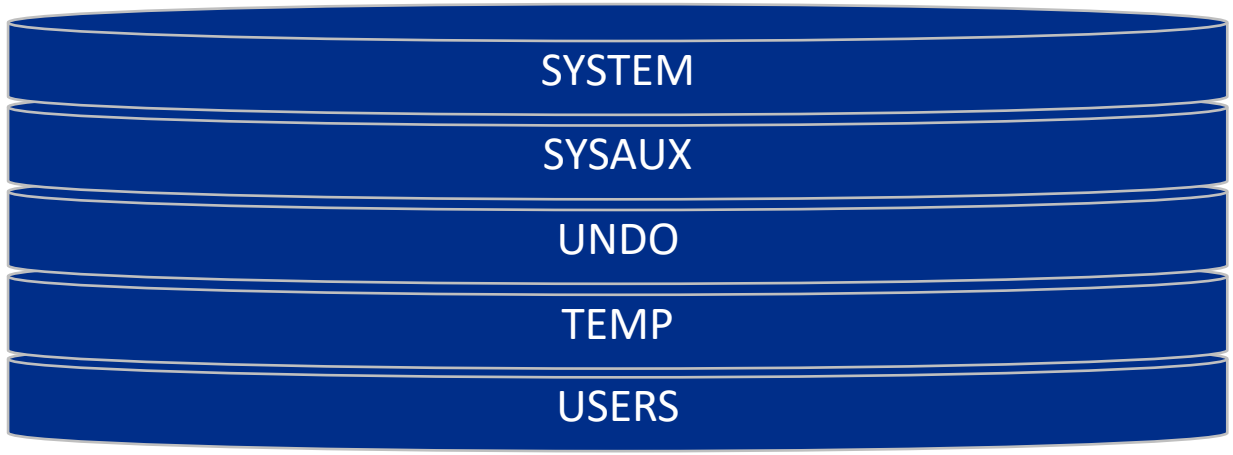
NON-CDB Database

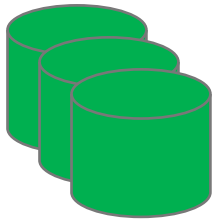


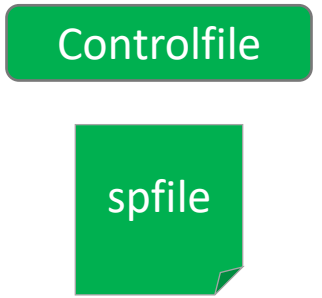
Multitenant Architektur



Root Container (CDB)





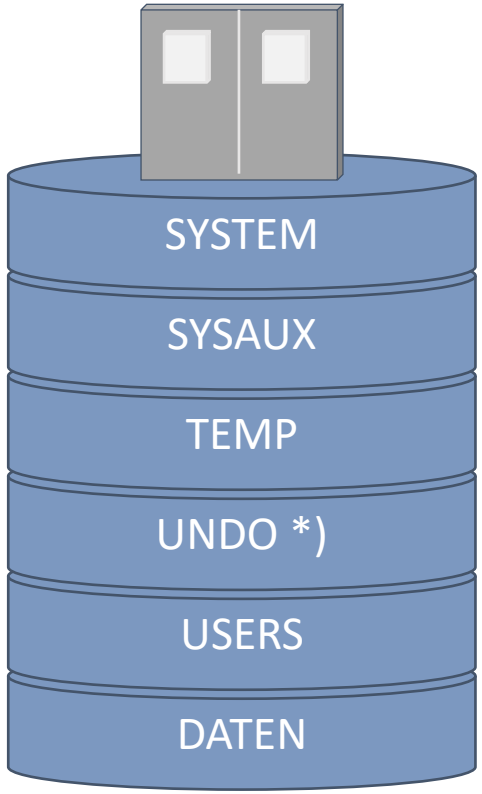


- Komponenten für den Aufbau der Instanz
  - Controlfiles
  - spfile

  → Connect an die Instanz nur über die CDB möglich
- Verwaltung der Redolog-Dateien

  → Vollständiges Recovery der Datenbank nur über die CDB möglich

Pluggable Database (PDB)



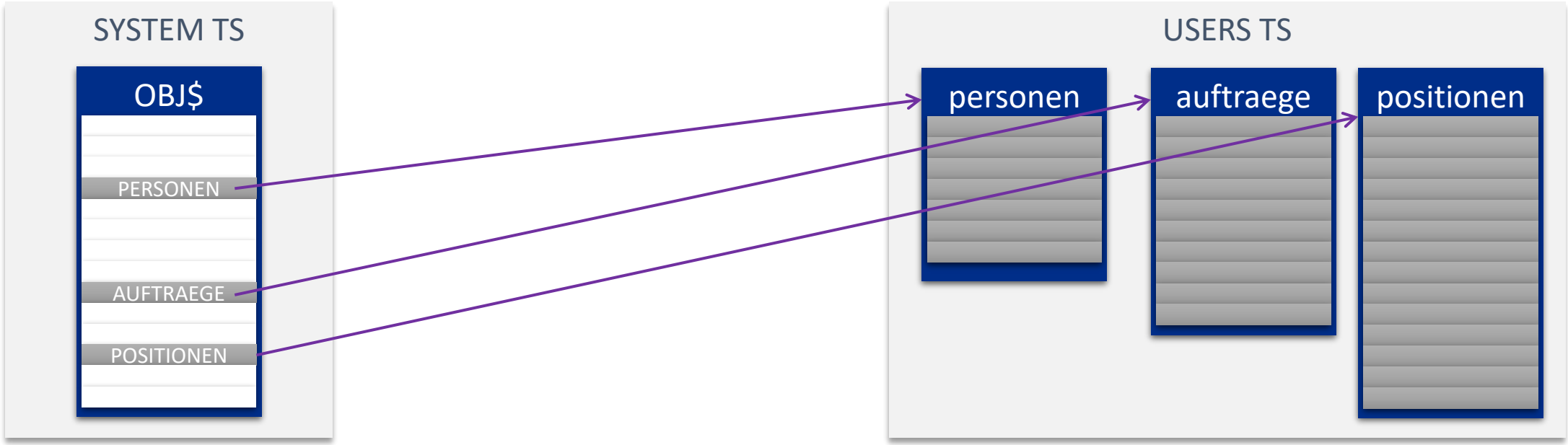
- In sich geschlossenes System
  - Data Dictionary
  - Tablespaces
  - Schemata
  - Objekte
  - Public Objekte
  - Workload Repository
  - Lokales Undo (~~nicht immer~~)
  - Parameter

Data Dictionary

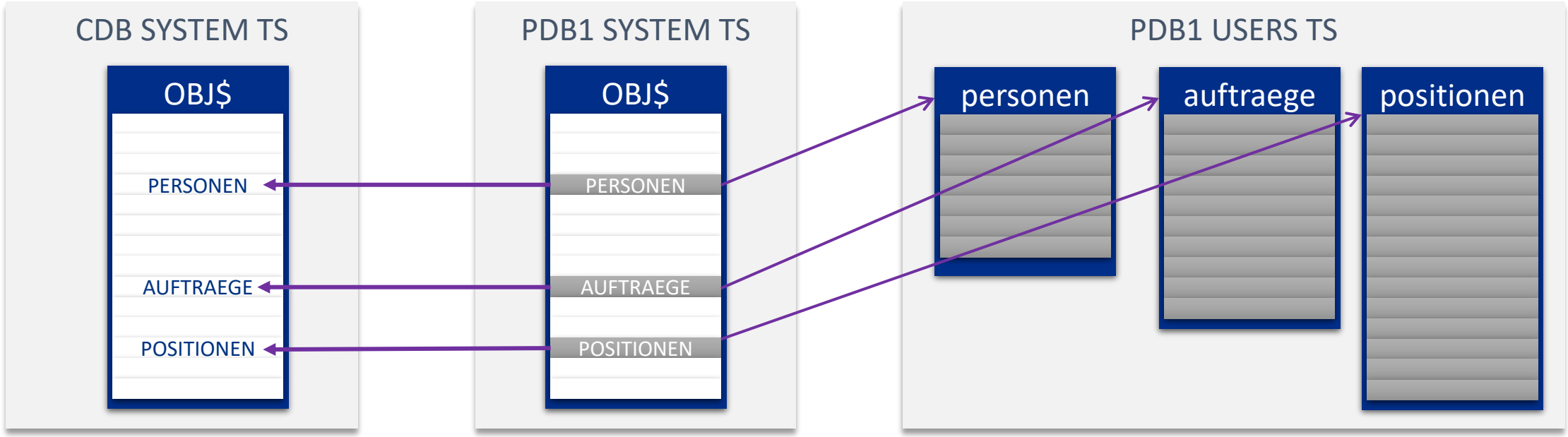
View Prefix

- `CDB_<view>` → Sicht auf alle Container-Daten aus der CDB genau wie DBA_<view> in der PDB CON_ID wird mit angezeigt
- `DBA_<view>` → Sicht auf die lokalen Objekte (egal ob CDB oder PDB) als DBA
- `ALL_<view>` → Sicht auf alle Objekte, auf die man zugreifen darf (nur lokal in der PDB bzw. CDB
- `USER_<view>` → Sicht des Benutzers auf seine eigenen Objekte

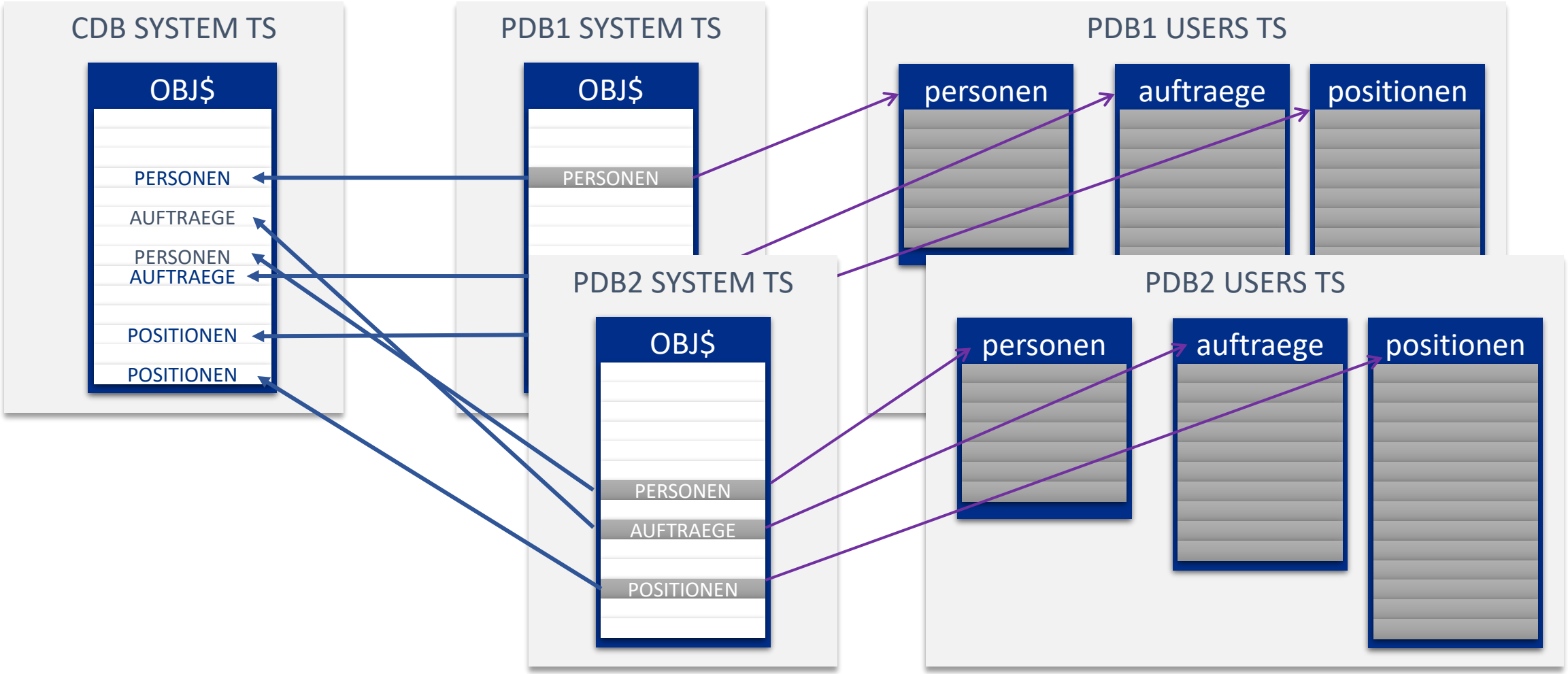
NON-CDB Data Dictionary



PDB Data Dictionary



PDB Data Dictionary



Data Dictionary und PDBs

- Data Dictionary Metadaten liegen in der CDB$ROOT
  - Z.B. `TAB$`
- PDBs haben einen Link in diese Root-Tabellen (Metadata Link)
- Data Dictionary Daten können sowohl im Root als auch in der PDB liegen
  - COMMON USER → Root
  - Local Users → PDB
  - PDB bekommt die Root-Daten vererbt (Read Only) und hat seine eigenen Daten

Internes Data Dictionary

- Abfrage 1

```
SELECT obj#, owner#, name FROM obj$ WHERE name = 'PERSONEN';
```

- CDB

```
no rows selected
```

- PDB1

```
      OBJ#     OWNER# NAME
---------- ---------- ----------
     19938         69 PERSONEN
```

- PDB2

```
      OBJ#     OWNER# NAME
---------- ---------- ----------
     19881         66 PERSONEN
```

Lokale Data Dictionary Views

- Abfrage 2

```
SELECT username, user_id, common FROM dba_users WHERE username IN ('DEMO','SYSTEM');
```

- CDB

```
USERNAME      USER_ID COMMON
---------- ---------- --------
SYSTEM              8 YES
```

- PDB1

```
USERNAME      USER_ID COMMON
---------- ---------- --------
SYSTEM              8 YES
DEMO               69 NO
```

- PDB2

```
USERNAME      USER_ID COMMON
---------- ---------- --------
SYSTEM              8 YES
DEMO               69 NO
```

Spezielle Funktionen

- Flashback PDB
- Restore PDB
- Clone / Copy PDB
- Snapshot Clone PDB
- Refreshable PDB
- Proxy PDB

Mythen und Wahrheiten

Benefits of the Multitenant Architecture for Manageability

The multitenant architecture improves manageability by storing the data *and* metadata specific to a PDB in the PDB itself.

By storing its own dictionary metadata, a PDB becomes easier to manage as a unit. This benefit occurs even when only one PDB resides in a CDB. Grouping PDBs into a separately managed application container increases manageability even further.

In a CDB, the data dictionary metadata is split between the CDB root and the PDBs. Benefits of data dictionary separation include the following:

- Easier upgrade of data and code

  For example, instead of upgrading a CDB from one database release to another, you can rapidly unplug a PDB from the existing CDB, and then plug it into a newly created CDB from a higher release.

ALTERNATIVE FAKTEN

https://docs.oracle.com/en/database/oracle/oracle-database/21/multi/multitenant-administrators-guide.pdf

Kontakt

- E-Mail: johannes.ahrends@carajandb.com
- Homepage: www.carajandb.com
- Adresse:
  - CarajanDB GmbH
    Siemensstraße 25
    50374 Erftstadt
- Telefon: +49 (1 70) 4 05 69 36
- Twitter: carajandb
- Facebook: johannes.ahrends
- Blogs:
  - blog.carajandb.com