～ みなさまの投稿をお待ちしております ～

Twitter

#OracleTechNight

Oracle **Database** Technology Night
～集え！オラクルの力（チカラ）～

DB 12cから実装された
マルチテナント・アーキテクチャで
DBがより使いやすくなる

日本オラクル株式会社
クラウド・テクノロジー事業統括
Database & Exadataプロダクトマネジメント本部
伊藤 勝一

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

本日のセッションの構成

第1部 17:15～18:15

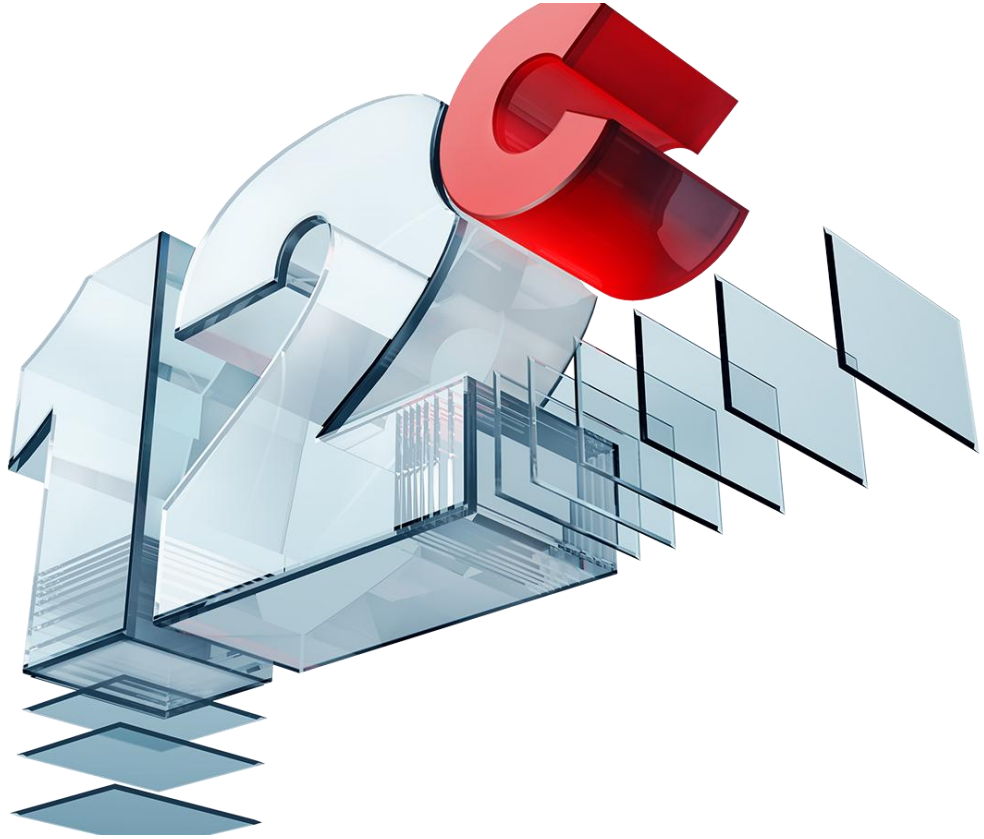
マルチテナント・アーキテクチャ 基礎編

第2部 18:45～19:45

DB12c R2新機能で広がるユースケース

“クラウド・ファースト” : Oracle Database 12c Release 2

- 提供開始済み
  - Exadata Express Cloud Service
  - Database Cloud Service
  - Exadata Cloud Service



Oracle Multitenant: 12.2で実装された新機能

NEW IN 12.2

プロビジョニングの容易さとテナントの移動しやすさ

- PDB再配置
- リフレッシュ・クローン
- ホット・クローン

規模の経済性と独立性の確保

- 1CDBあたり最大4,096PDB
- メモリー、I/Oのリソース制御
- ロックダウン・プロファイル

アプリケーション・テナントの中央集中管理

- アプリケーション・コンテナ
- プロキシPDB
- コンテナ・マップ

ORACLE

第1部
マルチテナント・アーキテクチャ 基礎編

第1部:マルチテナント・アーキテクチャ基礎編
Agenda

1. データベースの統合手法
2. マルチテナント・アーキテクチャ
3. プラガブル・データベースの管理・運用
4. プラガブル・データベースのプロビジョニング

第1部

1.データベースの統合手法

データベース統合におけるチャレンジ

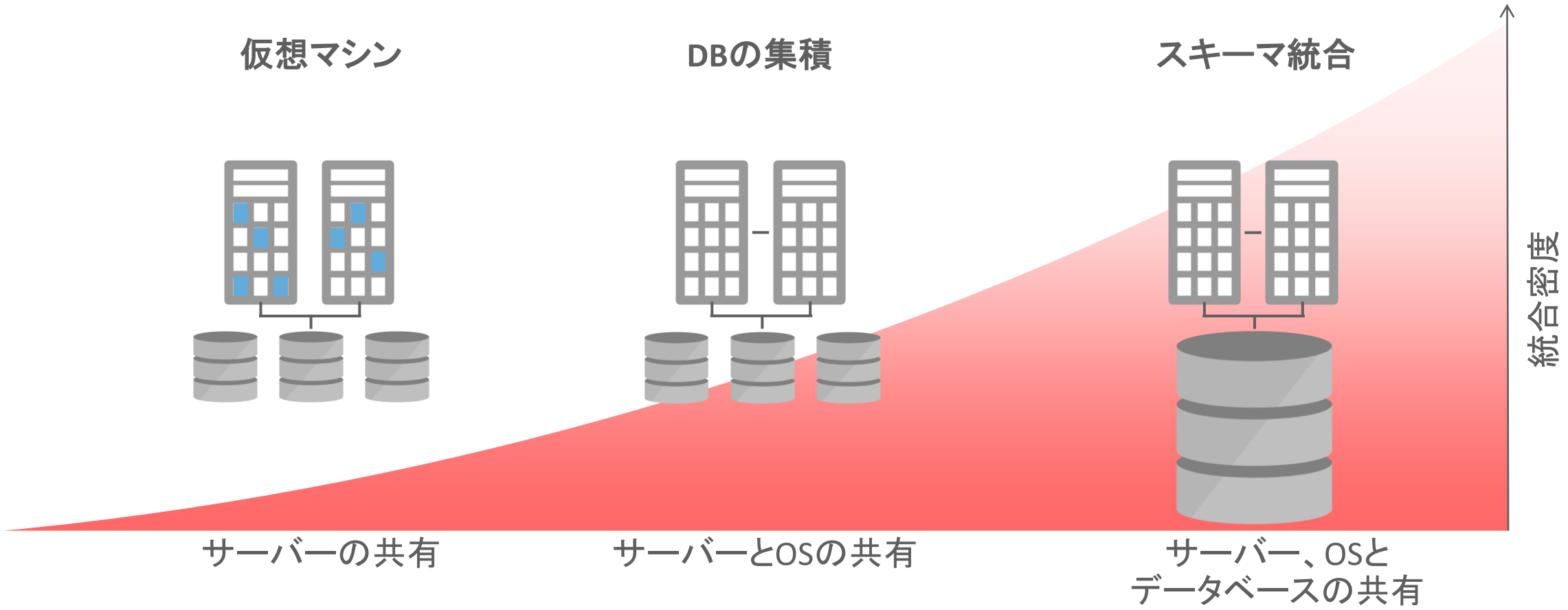
サーバー統合による
ITコストの削減

データベース数の
削減

アプリケーション
の独立性は維持、
変更は不要

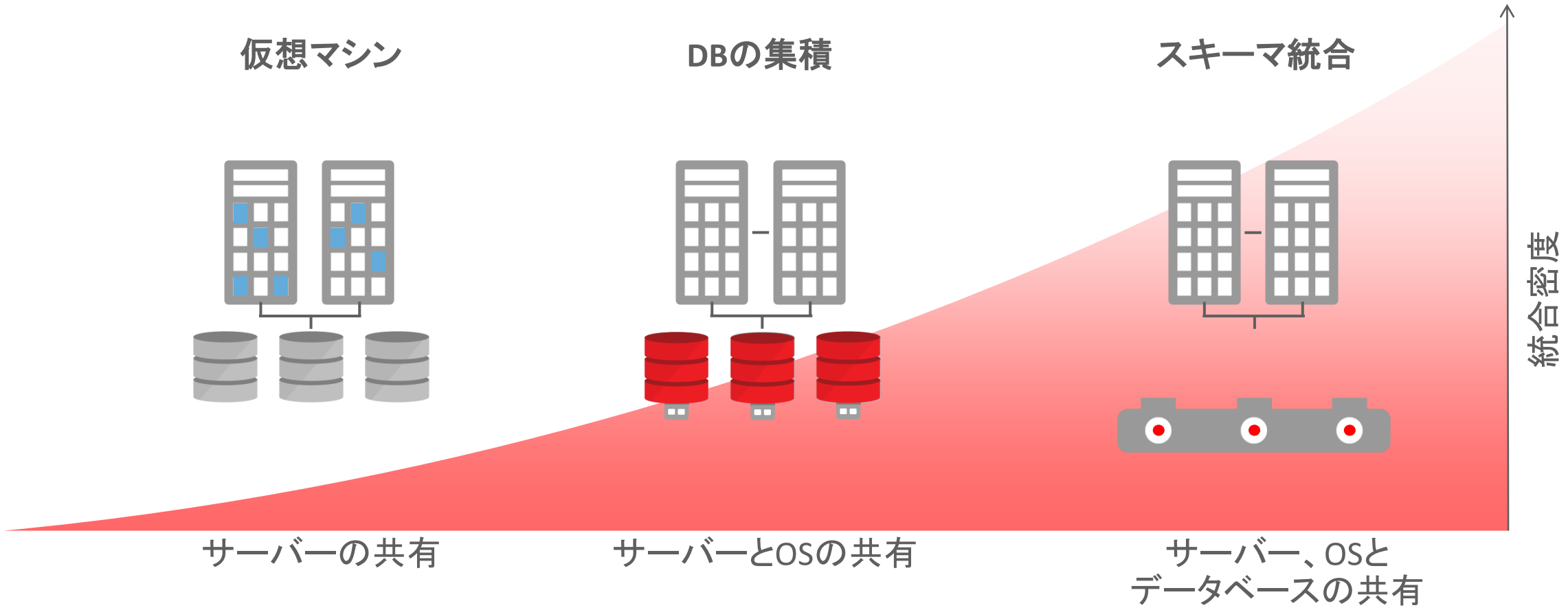
データベース統合

従来の統合手法



Oracle Multitenant

容易で簡潔な統合手法の提供、Database as a Serviceを実現



投資コスト
CapEx

合理化
標準化
自動化

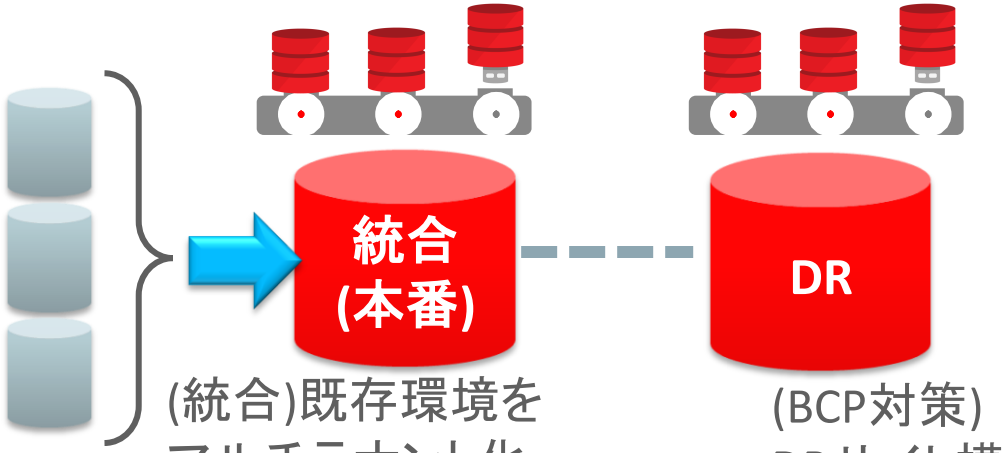
運用コスト
OpEx

Key Benefits

ベネフィット	有効な機能
CapExの最小化	• 1サーバー当たりのアプリケーション数
OpExの最小化	• Manage many as one (パッチ適用作業の削減) • 手順とサービス・レベルの標準化 • セルフ・サービスによるプロビジョニング
アジリティの最大化	• 開発・テスト環境用にスナップショット・クローニング • “プラガビリティ”による可搬性 • RACによるスケーラビリティ
容易	• 適用: アプリケーションの変更不要 • 利用: インターフェースはSQL

マルチテナントの代表的なユースケース

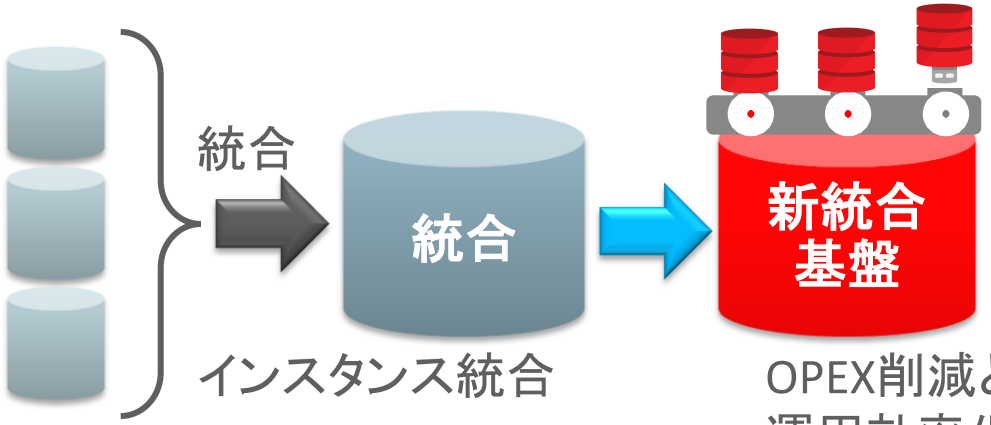
①既存システムの統合基盤(CAPEXとOPEXの削減)



(統合)既存環境を
マルチテナント化

(BCP対策)
DRサイト構築

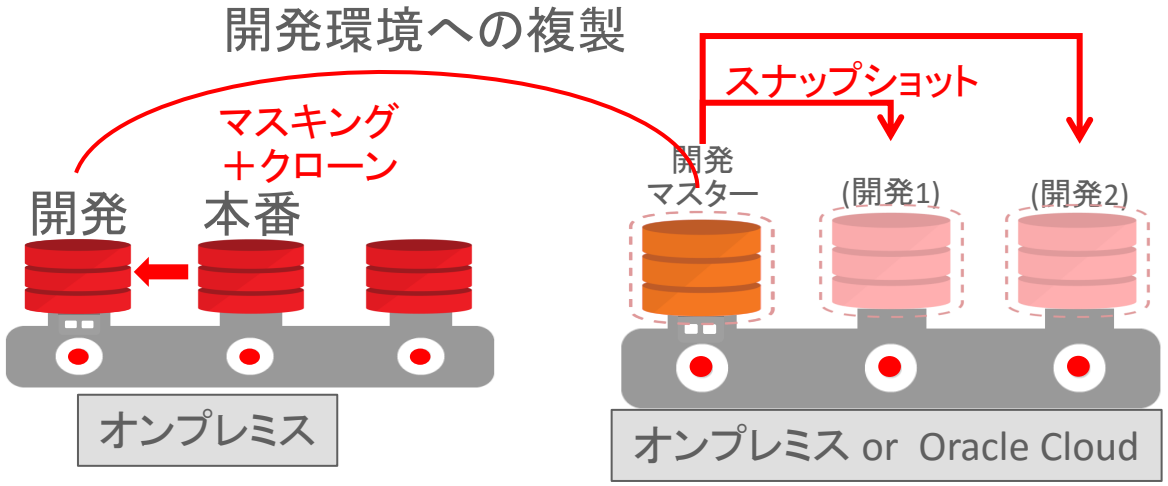
②統合システムの更改(12cにアップグレード)



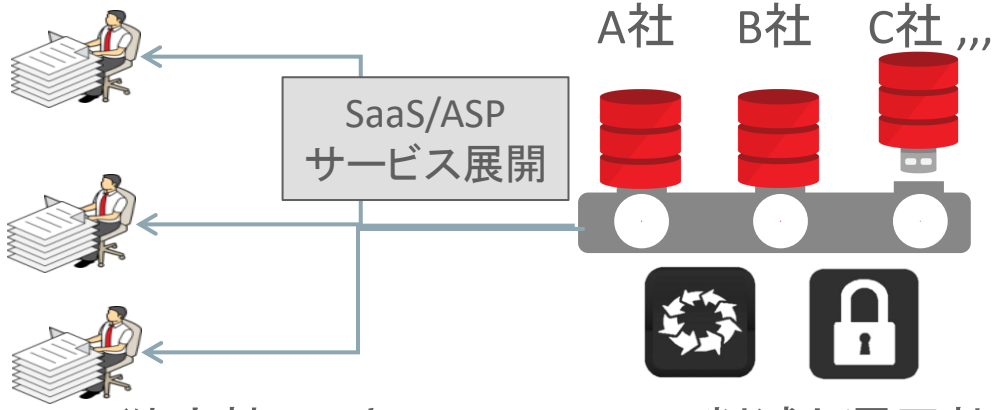
インスタンス統合

OPEX削減と
運用効率化

③開発・検証環境のアジリティ向上



④ SaaS/ASPサービスの基盤としての活用



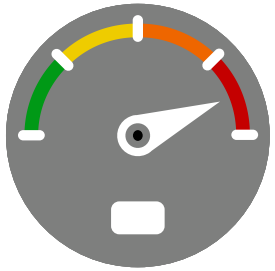
独立性、セキュリティ、OPEX削減と運用効率化

第1部

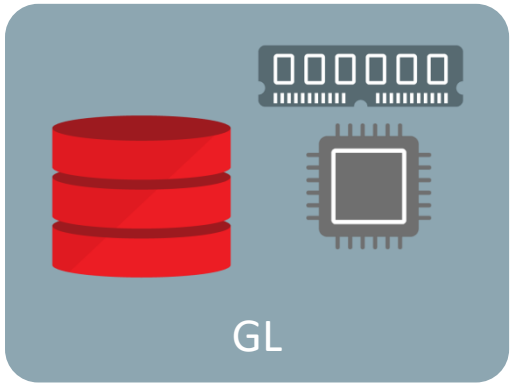
2. マルチテナント・アーキテクチャ

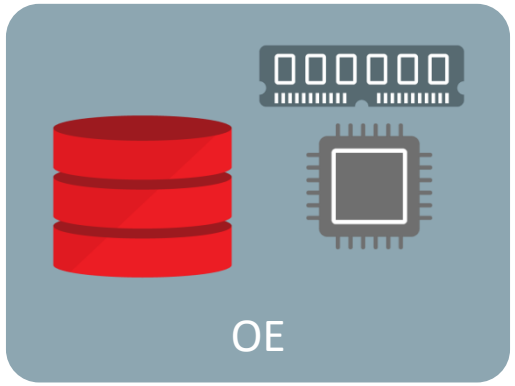
Oracle Databaseアーキテクチャ

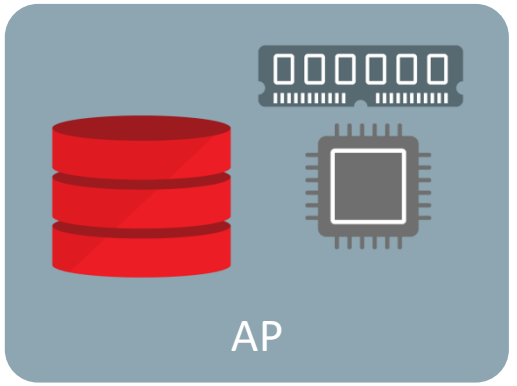
メモリー、バックグラウンド・プロセス、データファイルが必要



システムのリソース使用量

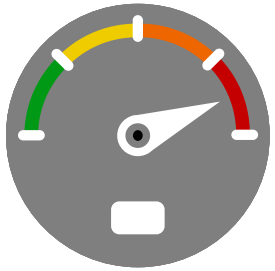




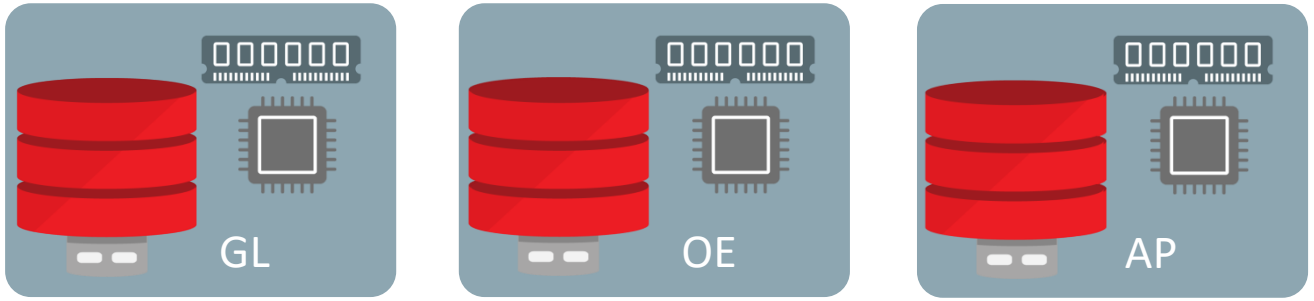


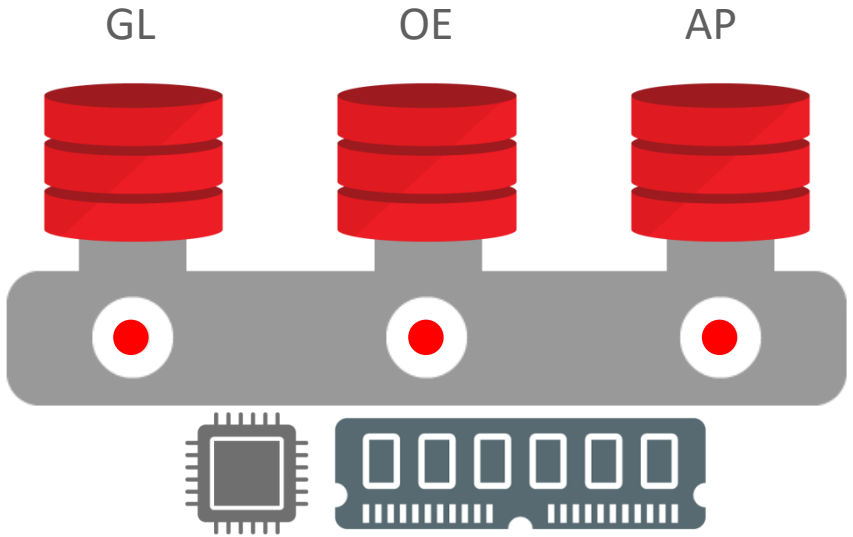
マルチテナント・アーキテクチャ

メモリー、バックグラウンド・プロセスが必要なのは、コンテナ・データベースのみ



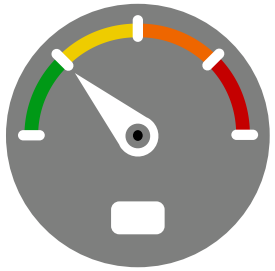
システムのリソース使用量



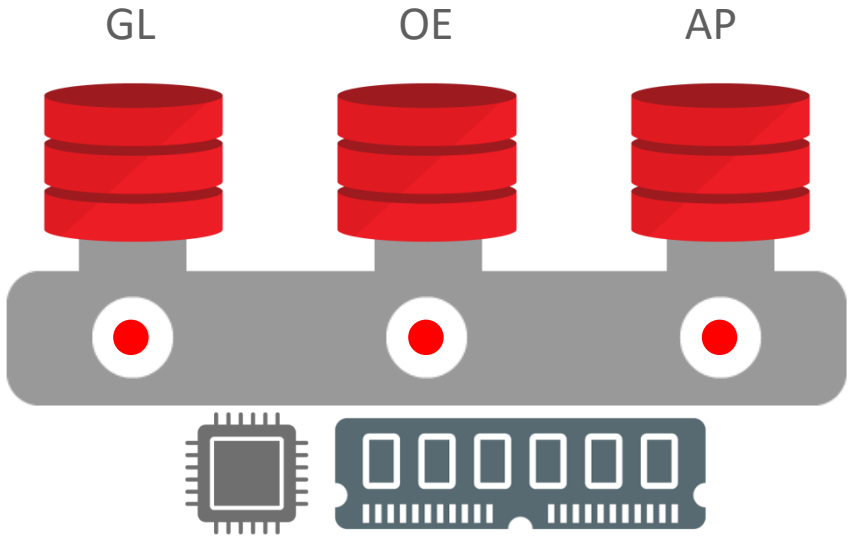


マルチテナント・アーキテクチャ

より効率のよいシステム・リソースの利用

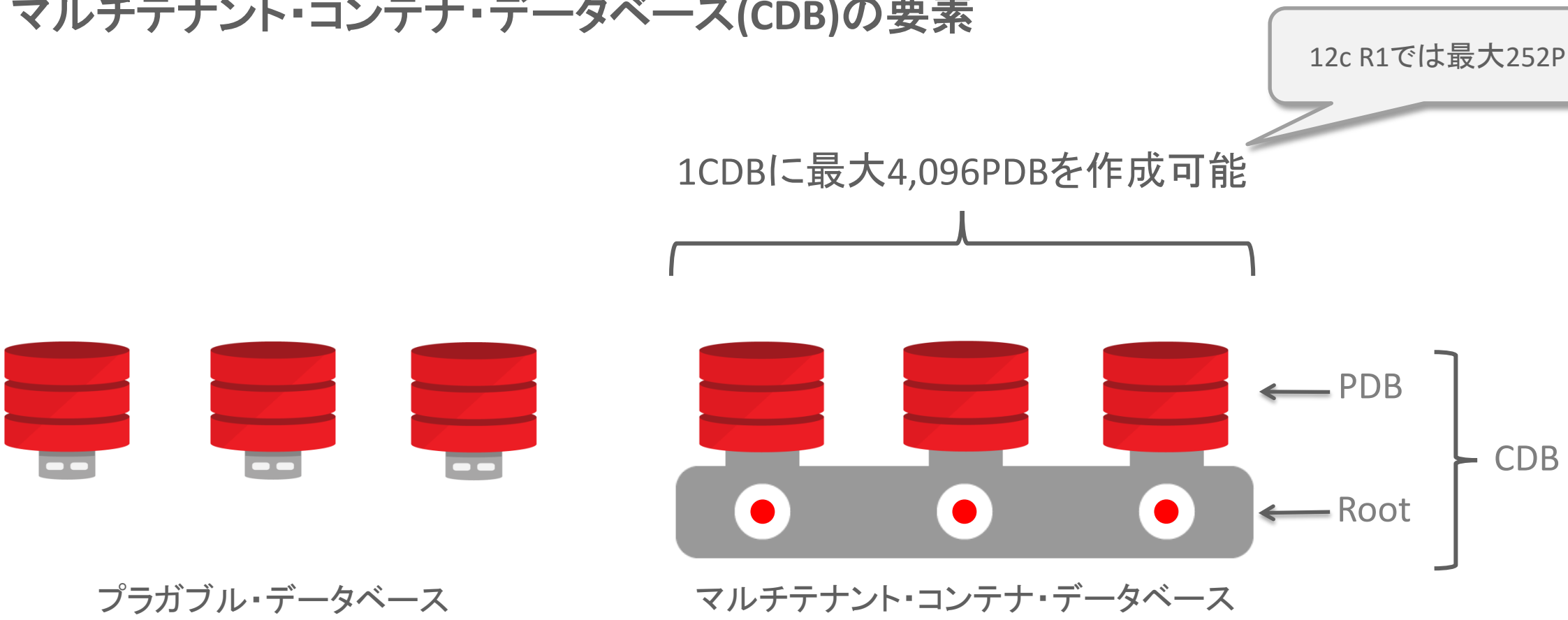


システムのリソース使用量

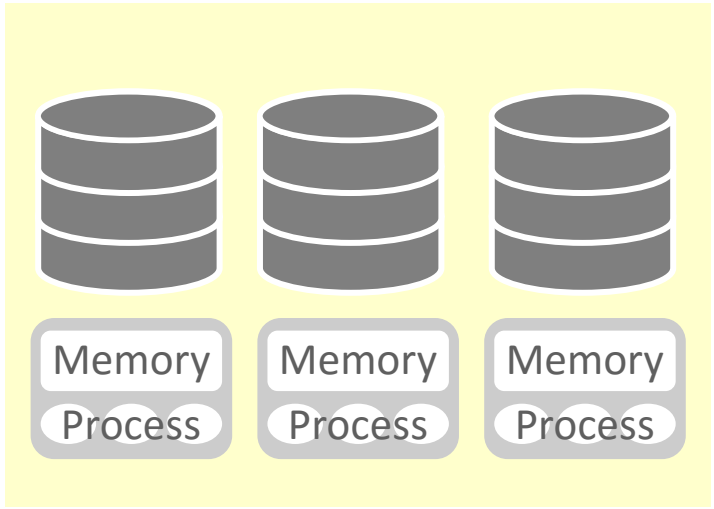


マルチテナント・アーキテクチャ

マルチテナント・コンテナ・データベース(CDB)の要素

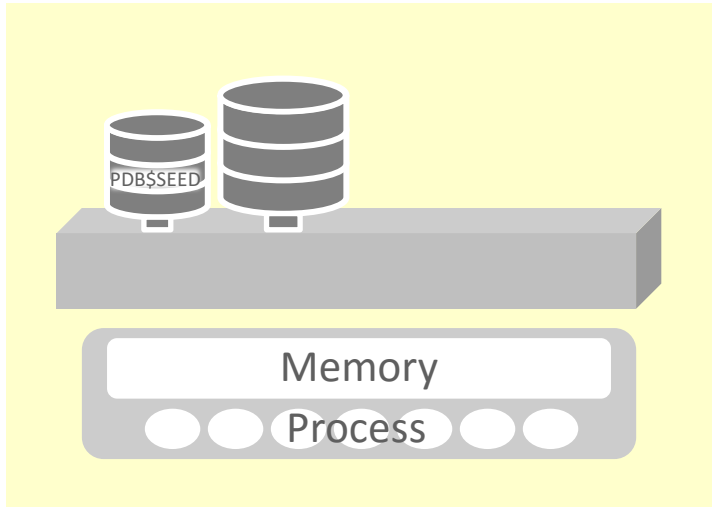


Oracle Database 12c
データベース構成の選択肢



Non-CDB構成

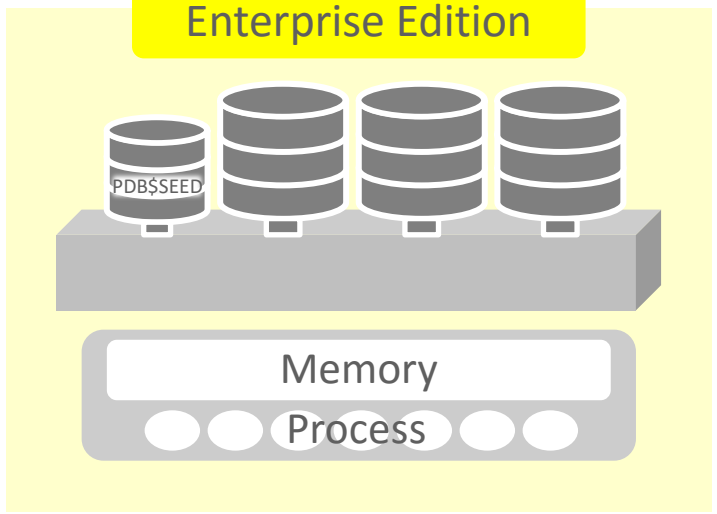
- 従来型のデータベース構成



シングルテナント構成

- CDB内にPDBを1つだけ有する構成

Oracle Multitenant
Enterprise Edition



マルチテナント構成

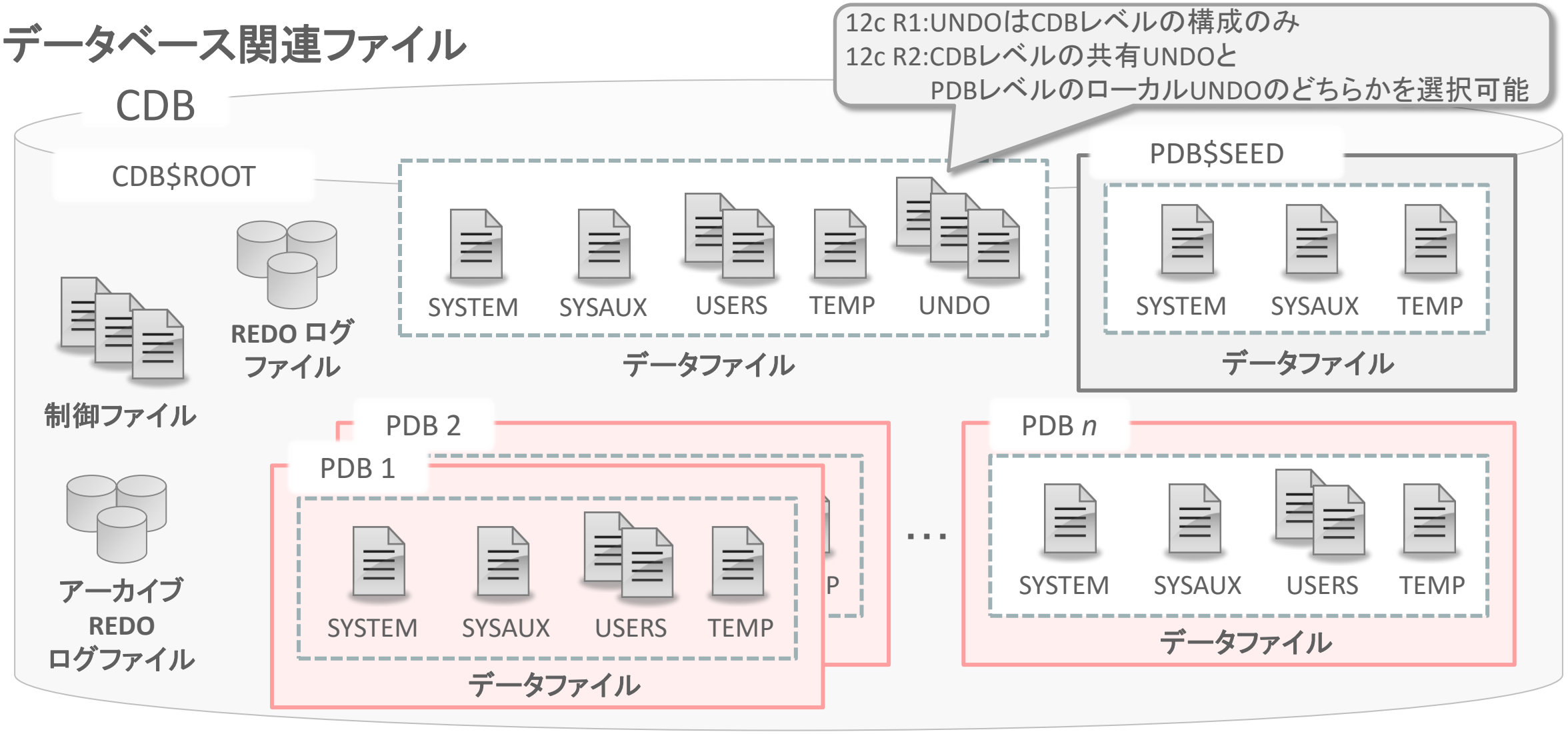
- CDB内にPDBを2つ以上有する構成

マルチテナント構成はマルチテナント・アーキテクチャのメリットを最大限に享受できる構成

1筐体内に複数のCDBを稼働させることも可能

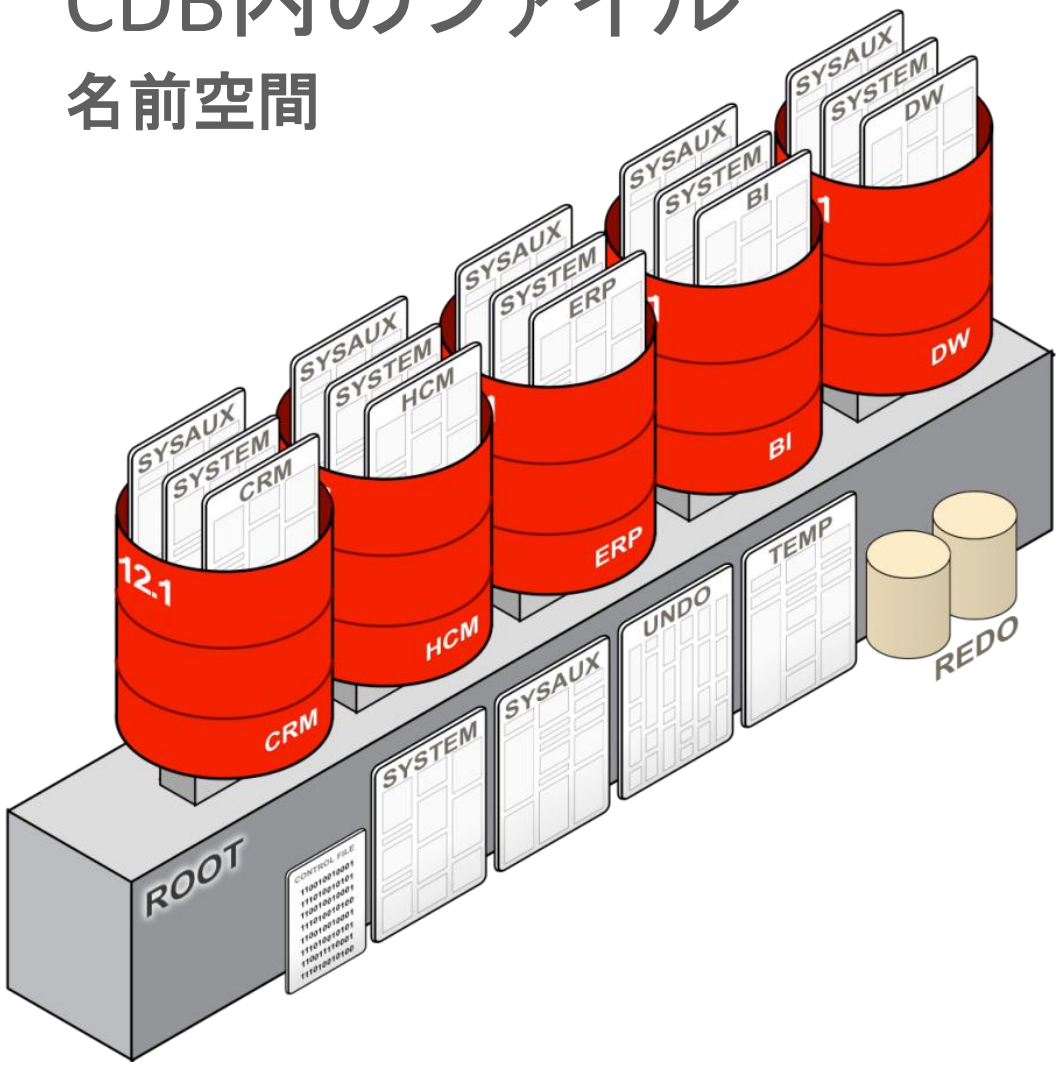
マルチテナント・コンテナ・データベースの物理構造

データベース関連ファイル



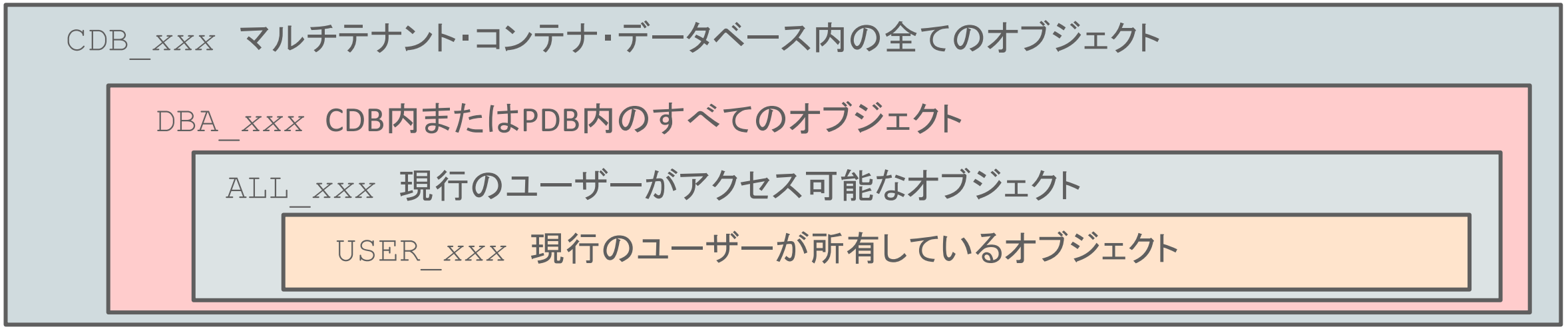
CDB内のファイル

名前空間



- それぞれのPDBは固有の表領域のセットを保持
  - SYSTEM 、SYSAUXも含まれている
- PDBはUNDO、REDO、制御ファイル、初期化パラメータファイルを共有
  - 12c R2ではUNDOはPDBごとに持つ構成が可能

データ・ディクショナリ・ビュー




```
SQL> SELECT view_name FROM dba_views WHERE view_name like 'CDB%';
```

- **CDB_PDBs**: CDB内のすべてのPDB
- **CDB_tablespaces**: CDB内のすべての表領域
- **CDB_users**: CDB内のすべてのユーザー(共通ユーザーとローカル・ユーザー)

- DBAディクショナリ・ビューはPDB内の情報が参照可能

```
SQL> SELECT table_name FROM dict WHERE table_name like 'DBA%';
```

データ・ディクショナリ・ビューの例 (CDB_*xxx*)

CDB_TABLESPACESビュー

- 接頭辞が CDB_ であるビューには、すべてのコンテナの情報が含まれる
  - ルートおよび、すべての PDB の情報を確認することが可能

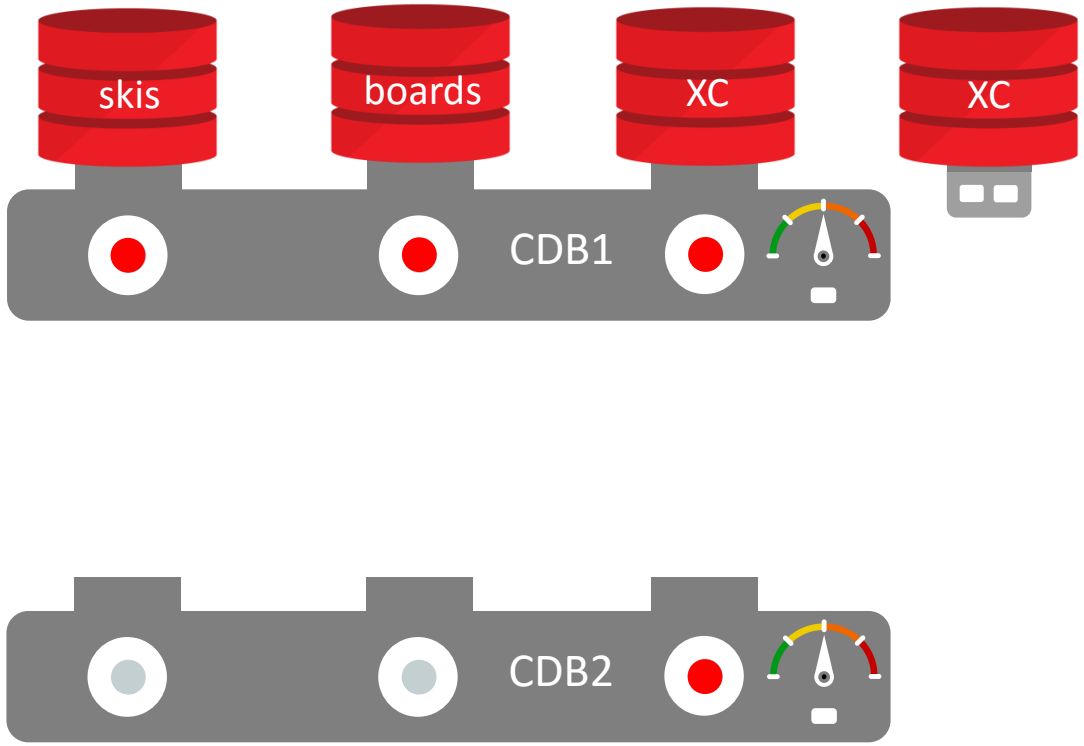
```
SQL> SELECT TABLESPACE_NAME, STATUS, CON_ID FROM CDB_TABLESPACES;

TABLESPACE_NAME      STATUS            CON_ID
-------------------- ------------ ----------
SYSTEM               ONLINE                1
SYSAUX               ONLINE                1
UNDOTBS1             ONLINE                1
TEMP                 ONLINE                1
USERS                ONLINE                1
SYSTEM               ONLINE                3
SYSAUX               ONLINE                3
TEMP                 ONLINE                3

8行が選択されました。
```

ルートの表領域

PDB の表領域

CON_ID	意味
0	CDB全体に関連するデータ
1	ルートに関連するデータ
2	シードに関連するデータ
3以上	PDBに関連するデータ

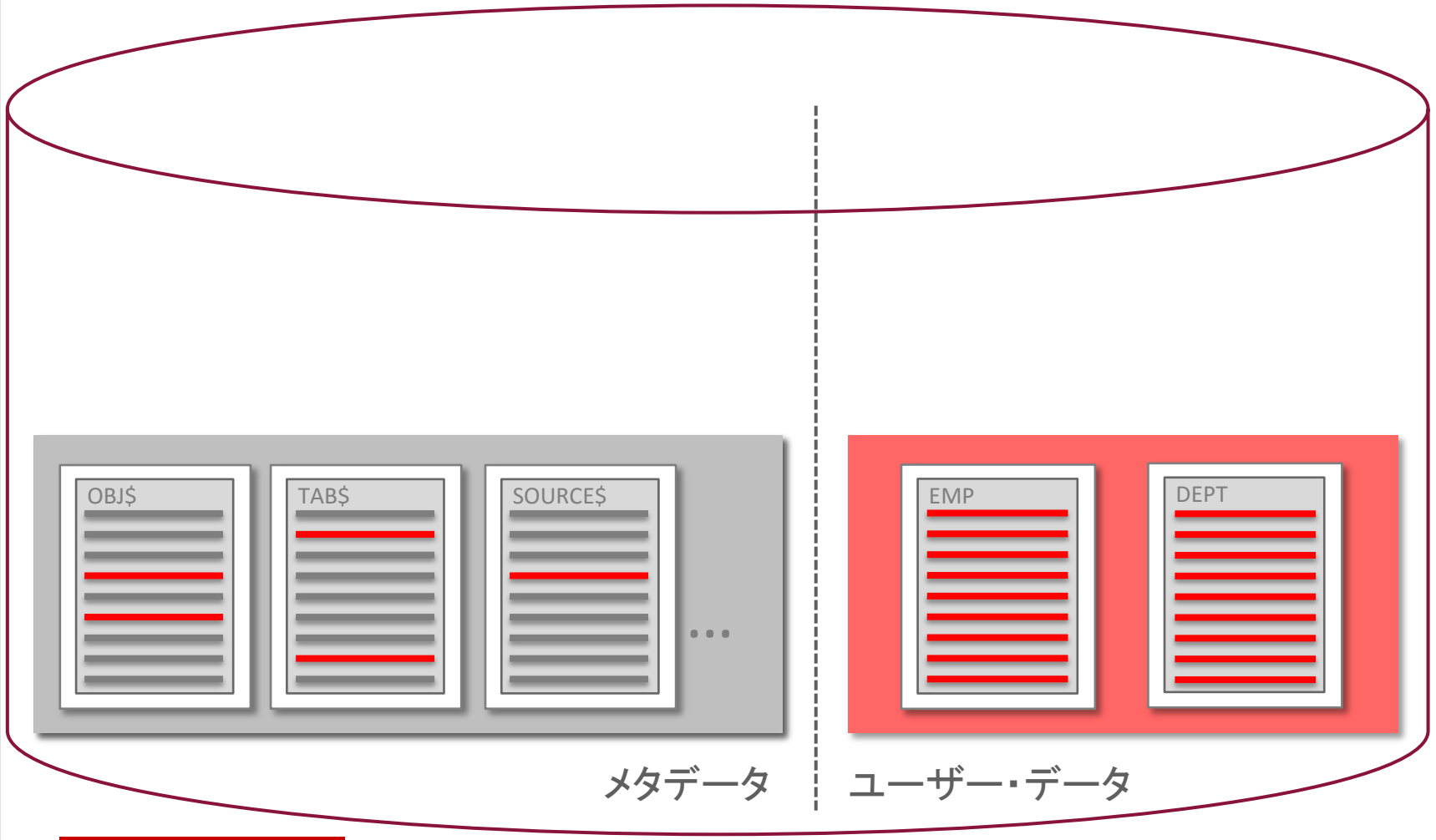
Design Goal: ポータビリティ/可搬性



- プラガブル・データベースはポータブル・データベース
- 利用していたCDBからアンプラグし…
- …新しいCDBにプラグインするだけ
- ストレージが共有されている場合は、CDB間の移動は、メタデータの移動のみ
- アンプラグされたPDBには利用可能なオプション情報、暗号化キーの情報なども含まれる

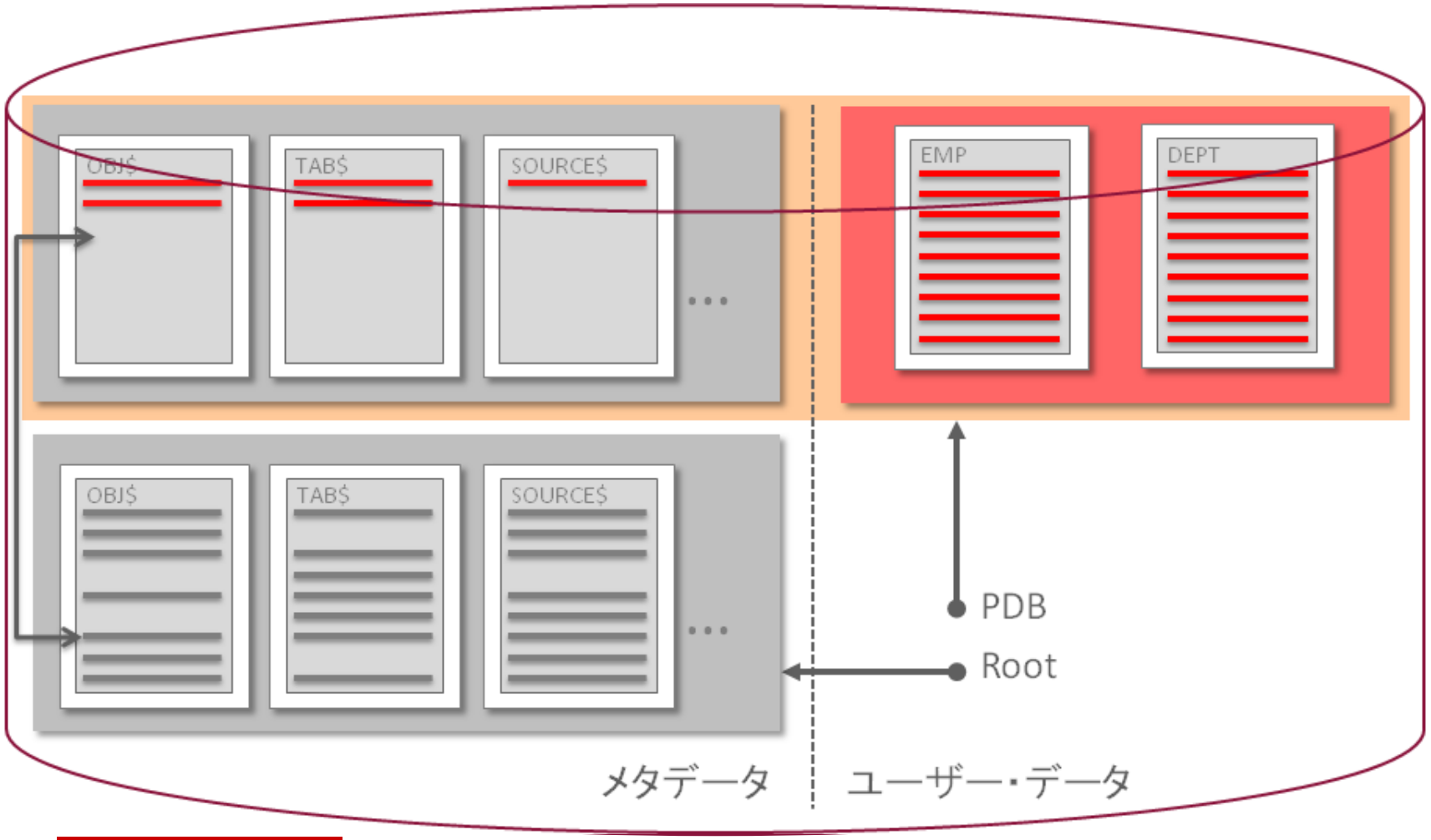
Oracleのメタデータとユーザー・データ

12.1より前: Oracleのメタデータとユーザー・データが 混在



- 新規に作成されたデータベースはメタデータのみが存在
- ユーザー・データをデータベースに格納
  - Oracleとユーザーのメタデータが混ざる
  - 可搬性の課題が生じる

Oracleのメタデータとユーザー・データ
マルチテナント・アーキテクチャによる *可搬性と互換性* の実現



- 新規に作成されたデータベースはメタデータのみが存在
- ユーザー・データをデータベースに格納
  - Oracleとユーザーのメタデータが混ざる
  - 可搬性の課題が生じる
- Multitenantによる解決: *水平分割されたデータ・ディクショナリ*
  - Oracle固有のメタデータのみがrootに存在

マルチテナント・アーキテクチャへの対応

データベースの構成タイプ	Non-CDB	CDB
シングル・インスタンス (Oracle Restart 構成を含む)	○ 対応	○ 対応
Oracle Real Application Clusters (ポリシー管理 / 管理者管理を含む)	○ 対応	○ 対応
Oracle RAC One Node (ポリシー管理 / 管理者管理を含む)	○ 対応	○ 対応
Oracle Data Guard (フィジカル / ロジカル・スタンバイを含む)	○ 対応	○ 対応

いずれのデータベース構成においても non-CDB/CDBで利用可能

CDB *vs.* PDB
各レベルでの構成/実施

共通する運用オペレーションはCDBレベル / テナントごとの設定はPDBレベル

CDBで構成/実施	PDBごとに構成/実施
Oracle Databaseのソフトウェア・バージョン	RMANによるポイント・インタイム・リカバリ
Data Guard、RAC	RMANのアドホックなバックアップ
RMANの通常のバックアップ	共有プールのフラッシュ
いくつかのパラメータやプロパティ値の設定 例:UNDOの構成(共有/ローカル)	属性のパラメータ設定 v$parameter `IsPDB_Modifiable = 'TRUE'`
REDO	

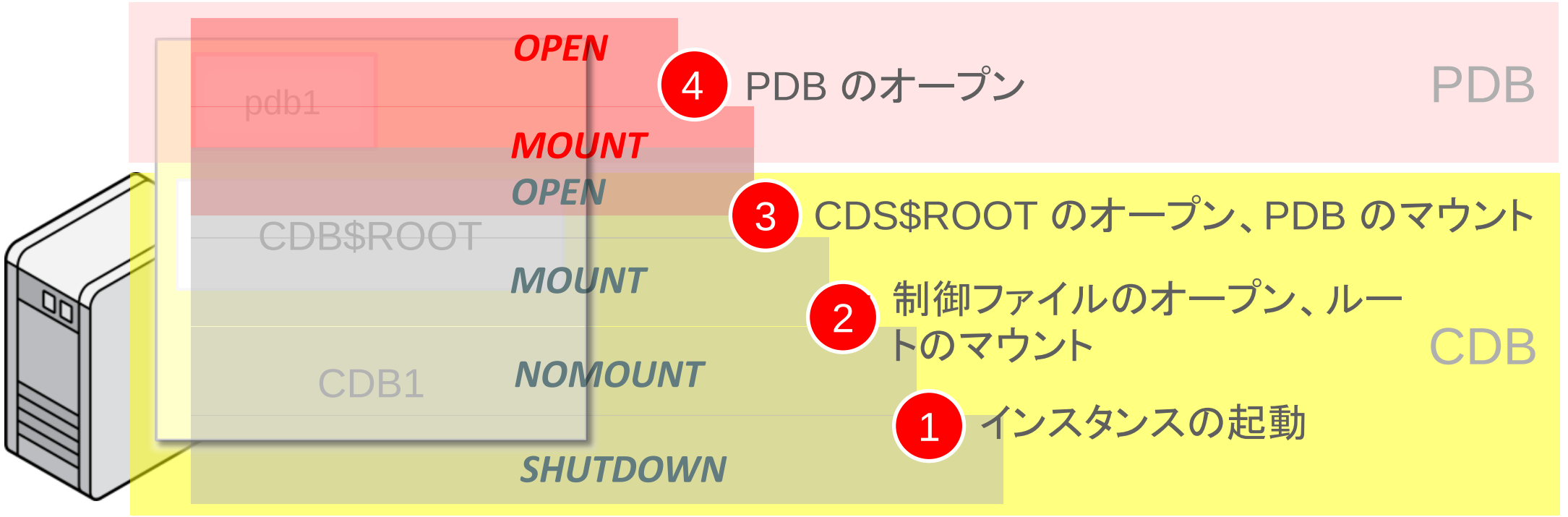
第1部

3. プラガブル・データベースの管理・運用

CDB 環境におけるデータベースの起動

起動するまでのステップ

- シャットダウンされた状態から PDB のオープンまで、次のフェーズで遷移する



プラガブル・データベースのステータス管理

PDBのオープン

- PDB のオープン操作は、CDB がオープンしていることが前提
- ルートへの接続時に PDB をオープンする場合の構文

```
ALTER PLUGGABLE DATABASE <PDB_NAME> OPEN [<OPTIONAL_CLAUSE>];
```

  - 例

```
ALTER PLUGGABLE DATABASE pdb1 OPEN READ ONLY;
```

    - 上記は PDB(pdb1)を、読み取り専用でオープンする場合のコマンド例
    - PDB名をカンマ区切りで羅列したり、ALLを指定してすることも可能
    - *<OPTIONAL_CALUSE>* には、次の指定が可能
      - オープンにおけるモードの指定
      - 制限付きモードの適用

プラガブル・データベースのオープン

制限付きモードの適用

- オープン・モード に加えて、RESTRICTED モードをオプションとして指定可能

  RESTRICTED

  - RESTRICTED SESSION 権限を持つユーザーのみ接続を許可
  - オープン・モードに　UPGRADE を指定した場合は、暗黙的に適用される

  – RESTRICTED モードの適用状況は、次のコマンドでも確認が可能

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 HR                             READ WRITE YES
         4 ERP                            MOUNTED
         5 CRM                            READ WRITE NO
```

プラガブル・データベースのステータス管理

PDB のクローズ

- オープンしている PDB のクローズとは、PDB のステータスをマウントにすることを指す

- ルートへの接続時に PDB をクローズする場合の構文
  - 例

```
ALTER PLUGGABLE DATABASE <PDB_NAME> CLOSE [<OPTIONAL_CLAUSE>];
```

    - 上記は PDB (pdb1) を、即時停止する場合のコマンド例

```
ALTER PLUGGABLE DATABASE pdb1 CLOSE IMMEDIATE;
```

CDB起動時のPDBのステータスの記録

- 指定したPDBのステータスを保存してインスタンス再起動時に適用する
  - 構文

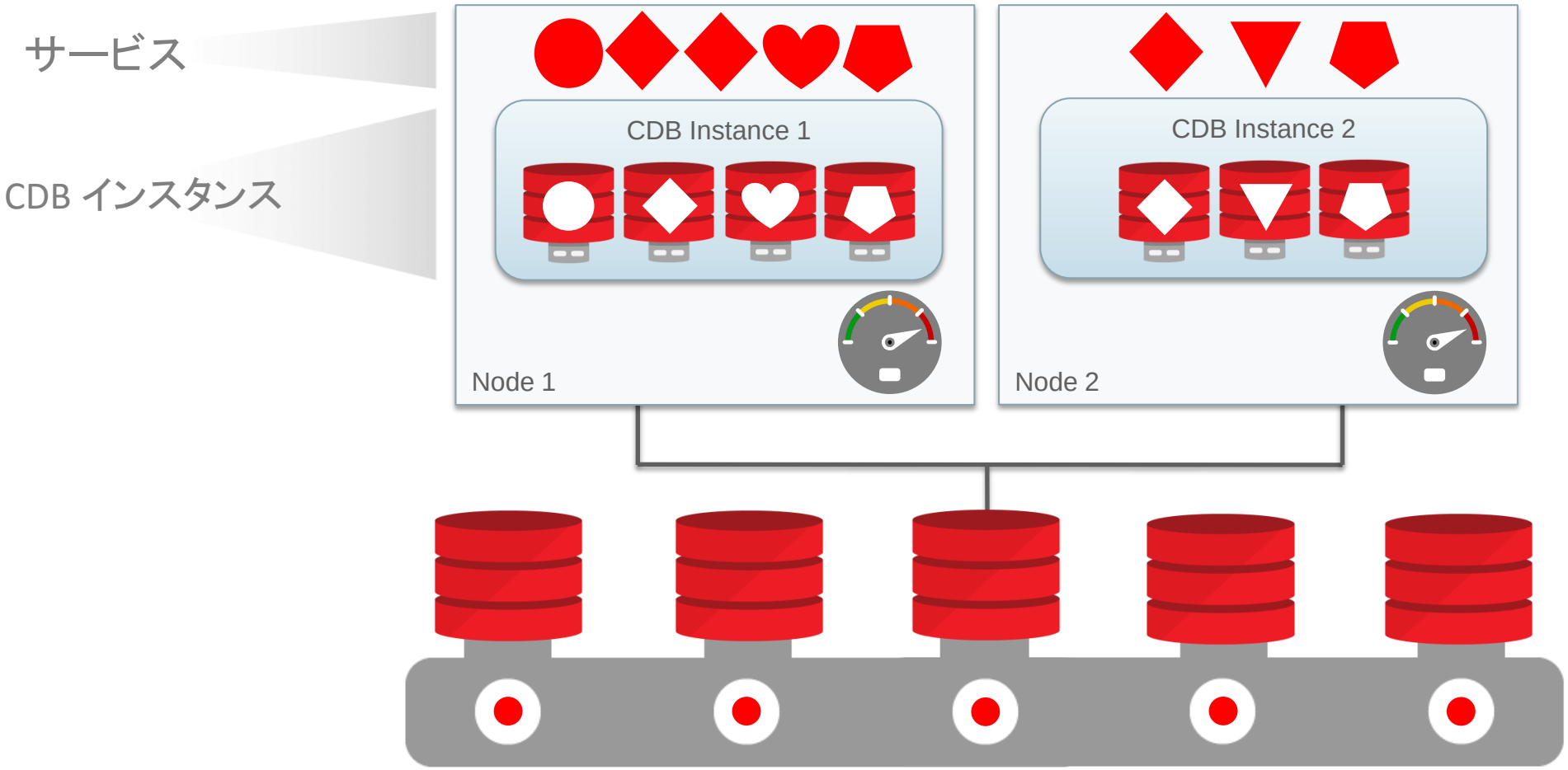
```
SQL> ALTER PLUGGABLE DATABASE pdb2 SAVE STATE;
```

- SAVE STATE / DISCARD STATE句による指定

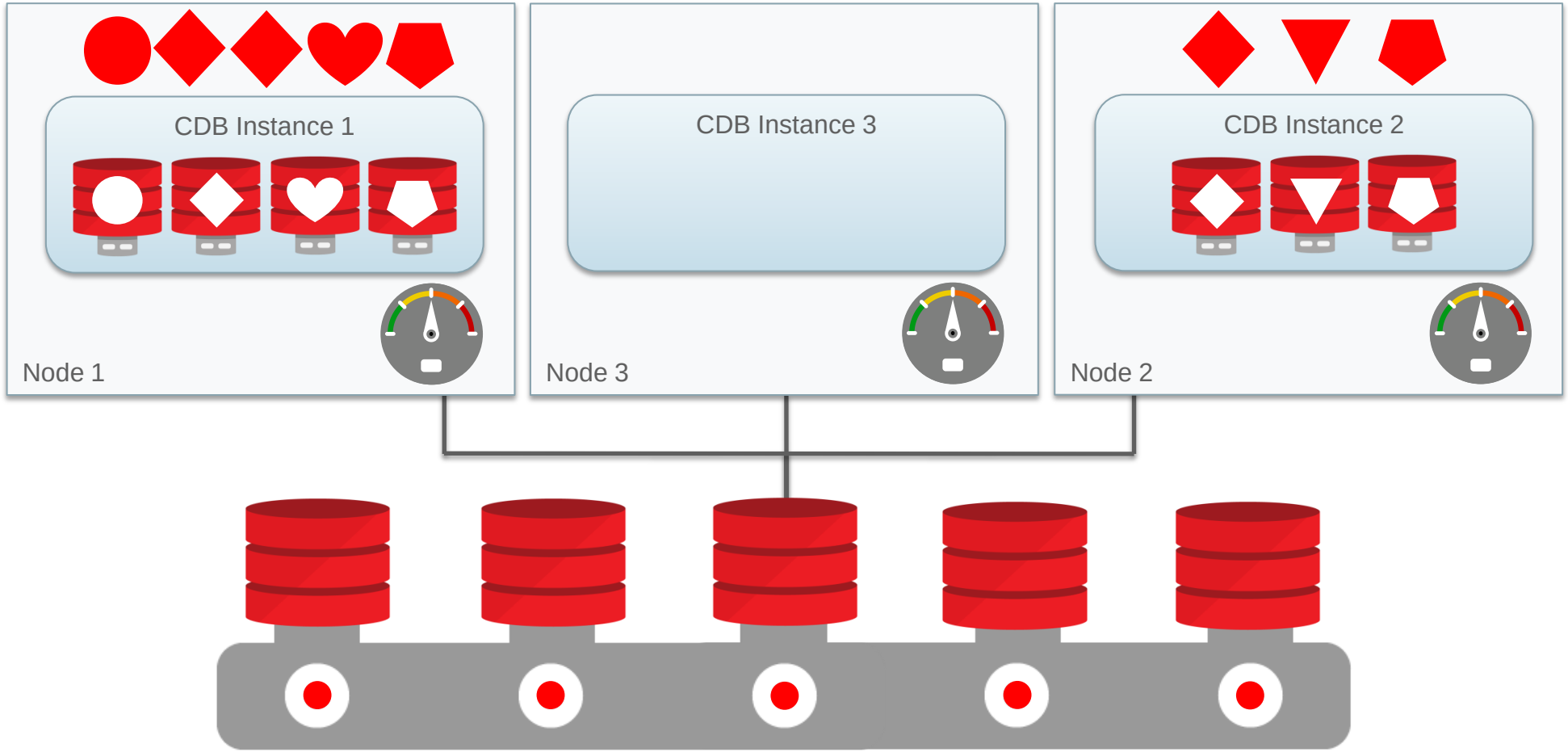
```
SQL> SELECT CON_ID, CON_NAME, STATE, RESTRICTED FROM DBA_PDB_SAVED_STATES;

    CON_ID CON_NAME         STATE           RESTRICTED
---------- ---------------- --------------- --------------------
         4 PDB2             OPEN            YES
```

- DBA_PDB_SAVED_STATESビューでステータスの保存状況を確認可能
- データベース・トリガーを利用してCDB起動時にPDBのオープンも可能

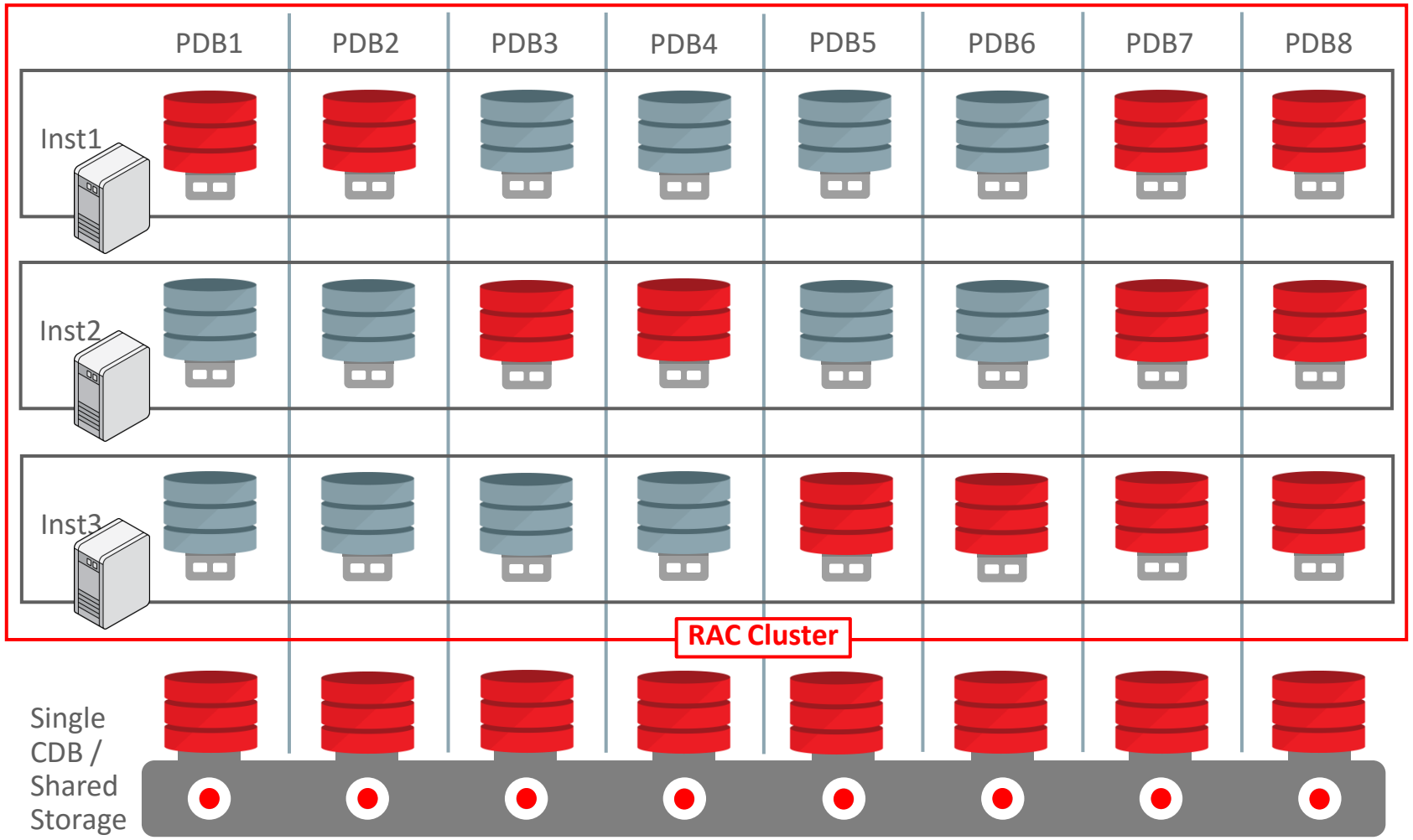
MultitenantとRACの組み合わせ:
ワークロードの変化への対応:アジリティの向上



MultitenantとRACの組み合わせ:
ワークロードの変化への対応:アジリティの向上



MultitenantとRACの組み合わせ:
アジリティ、可用性、拡張性が備わるDB統合基盤



- シングルCDBとして構成
- 1ノード上に1インスタンス
- PDBを”シングルトン”構成とし、特定のノードでオープンすることが可能
- 他のノードではマウント状態として見える
- PDBはすべてのインスタンス上でオープンし、利用することも可能

ユーザー

ユーザーの種類

ローカル・ユーザー

- 特定の PDB のみにユーザーが存在するタイプ
- ローカル・ユーザーを作成する場合は、PDB へ接続して操作を実施

共通ユーザー

- 各コンテナ(ルートと各 PDB) に同名のユーザーが存在するタイプ
- 共通ユーザーを作成する場合には、ルートへ接続して操作を実施
  - ユーザー名に接頭辞 (C## / c##) が必要

異なるユーザー・タイプの活用
ユーザーの作成例と役割

共通

ローカル

データベース全体の管理者は
共通ユーザーを使用する

CDB

CDB$ROOT

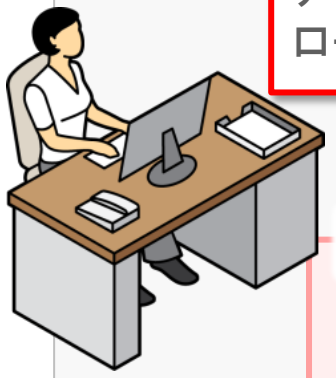
SYS

SYSTEM

PDB$SEED

SYS

アプリケーションごとのユーザーは
ローカル・ユーザーを使用する



PDB 2

PDB 1

SYS

USER01

SYSTEM

HRADM

…

PDB *n*

SYS

USER01

SYSTEM

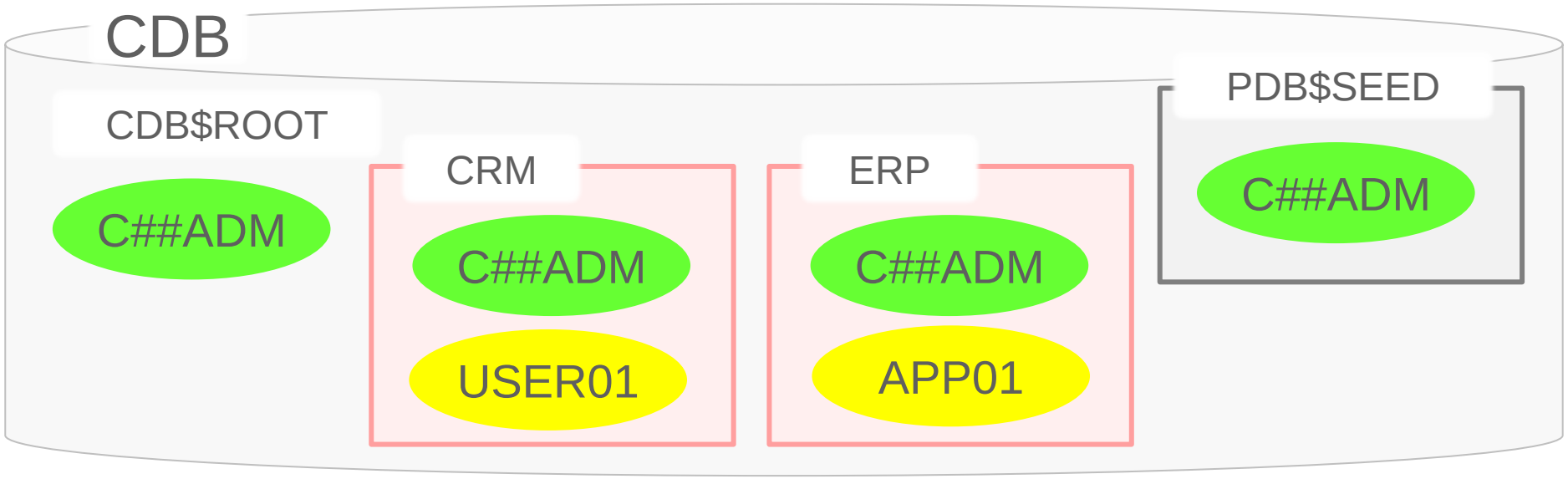
ERPADM

他の PDB とユーザー名は
重複してもよい

共通ユーザーの管理

プラガブル・データベースをアンプラグあるいはプラグする場合

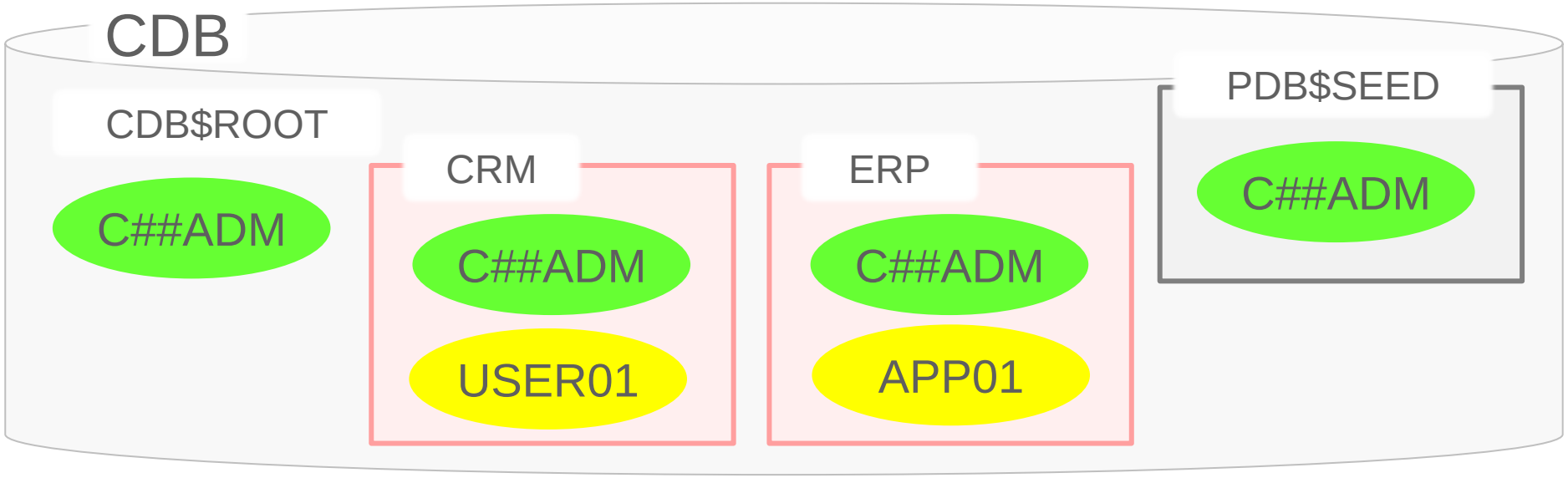
- PDBをアンプラグする場合、ローカル・ユーザーのみ設定を保持する
- PDBをプラグする場合には、接続する先のCDBの設定(共通ユーザーの作成状況)が反映される



共通ユーザーの管理

コンテナごとにロールや権限を個別に設定する場合

- PDB ごとのユーザーであれば、ローカル・ユーザーで対応する
- 既存の共通ユーザーに対して、PDBへの操作を制御したい場合にはコンテナごとにロールや権限の設定をすることが可能



ロールと権限

ロールと権限の種類

ローカル・ロール

- 特定の PDB のみに存在するロール、共通の権限は含まない

共通ロール

- ルートと各 PDB で共通のロール、共通およびローカル・ロールを含む
- 共通ロールを作成する場合には、ロール名に接頭辞(C##/c##)が必要

ローカル権限

- 特定の PDB のみに限定された権限

共通権限

- ルートと各 PDB で共通の権限、共通ユーザーによって付与される

共通ユーザーの作成

例

```
SQL> create user c##ora_Administrator
     identified by  pwd container=all;
```

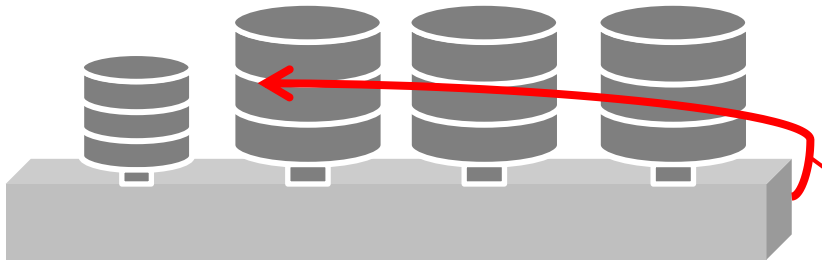
- 共通ユーザーの接頭辞を変更することも可能
- 初期化パラメータCOMMON_USER_PREFIXに接頭辞とする値を設定

```
SQL> SELECT NAME, VALUE, ISPDB_MODIFIABLE FROM V$PARAMETER
  2  WHERE NAME = 'common_user_prefix';

NAME                 VALUE                ISPDB_MODIFIABLE
-------------------- -------------------- ----------------
common_user_prefix   C##                  FALSE
```

  - 共通ユーザーの接頭辞をローカル・ユーザーが意図せず使用することを防止可能

コンテナ共通のオブジェクトに対するクエリー

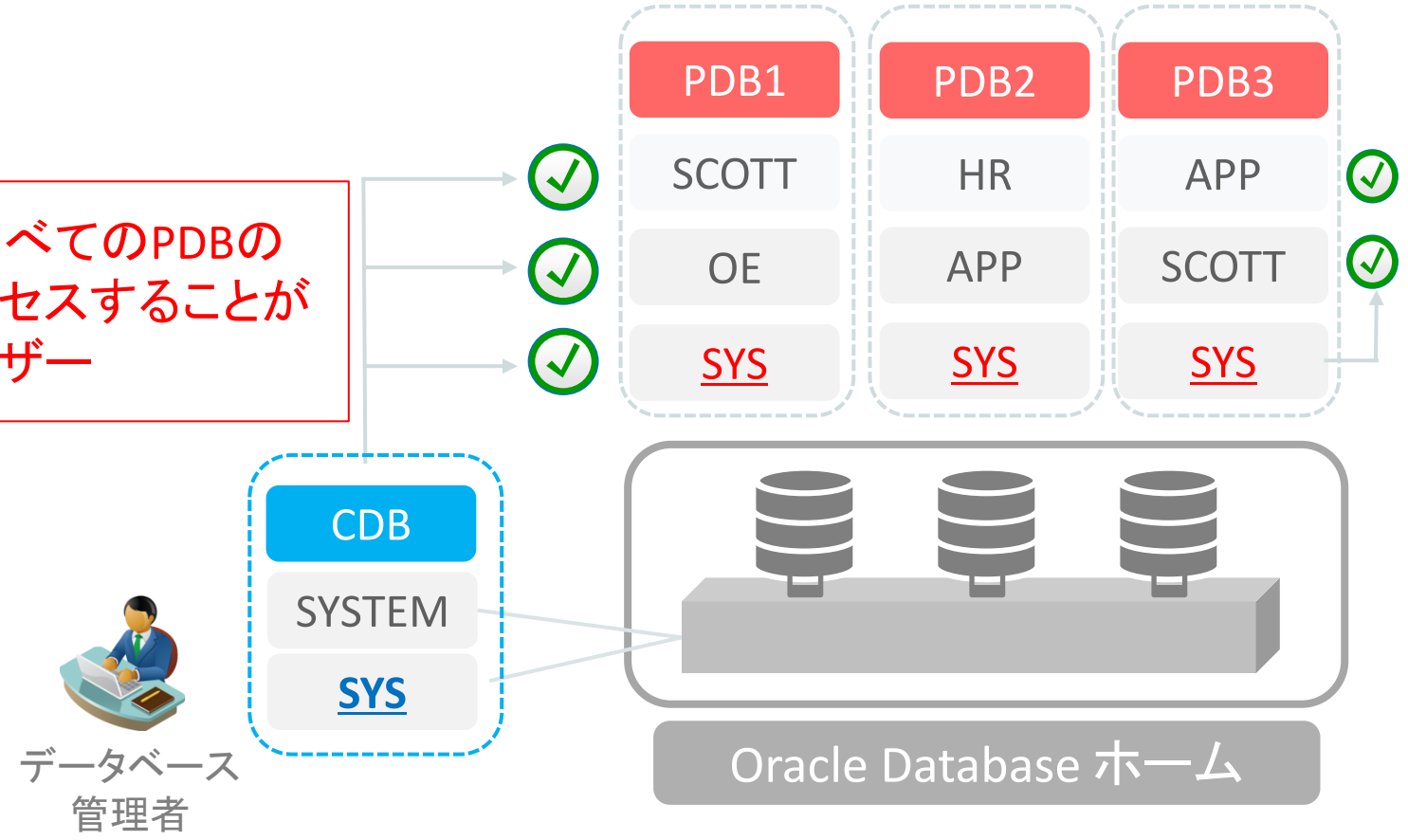
- 次のような共通のオブジェクトに対するクエリー機能の提供
  - デフォルトで事前定義されているオブジェクト
  - 共通ユーザーが所有する表、ビュー、索引
- コンテナをまたぐクエリーはルートから実行する
  - オブジェクトはルートにも存在する必要がある
  - PDBから実行した場合は、結果はそのPDBの情報に限定される



```
SQL> SELECT * FROM CONTAINERS(user.dept);
```

マルチテナント環境における特権ユーザー

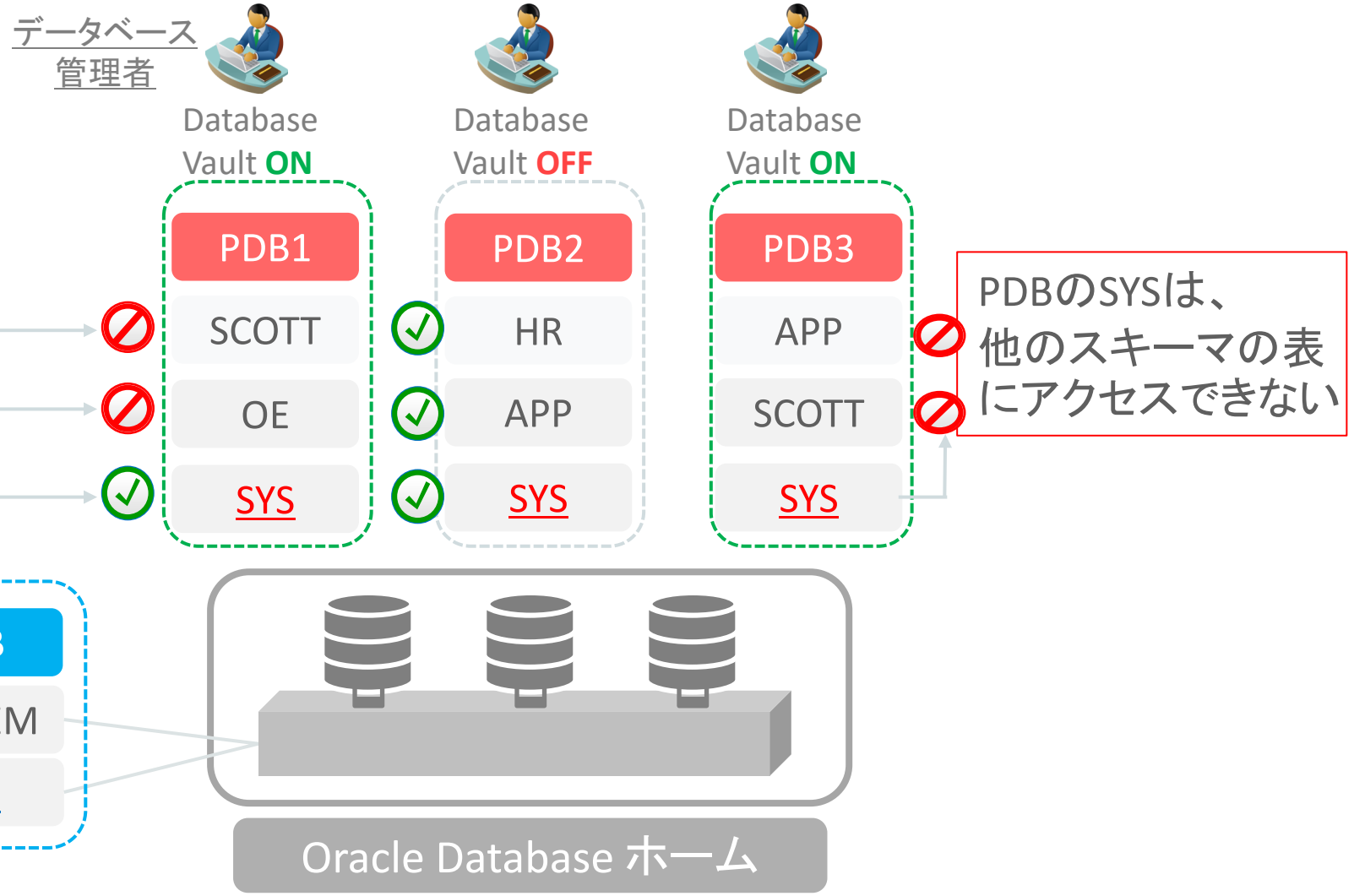
CDBのSYSは、すべてのPDBの表に自由にアクセスすることができる特権ユーザー



PDBのSYSは、PDB内の表にアクセスすることができる特権ユーザー

マルチテナント環境でDatabase Vaultを有効化した場合

それぞれのデータベース管理者が自分のPDBのみを管理する



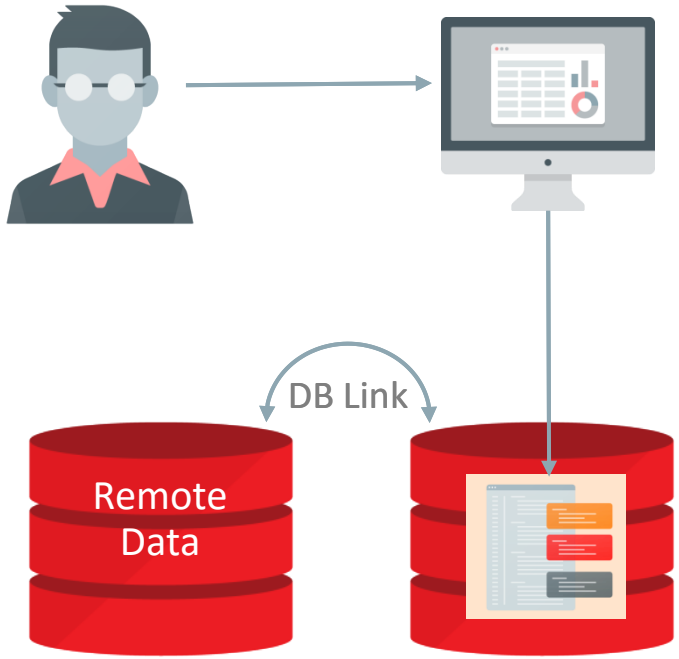
Database Vaultで保護されているPDBの表に対して、CDBのSYSユーザーはアクセスできない

PDBのSYSは、他のスキーマの表にアクセスできない

インフラ管理として必要な業務のみアクセスを許可

Design Goal: コンパチビリティ/互換性

従来のnon-CDBアーキテクチャ



- PDB / non-CDBの互換性の保証:

  接続されたクライアントからは、そのDBがPDBかnon-CDBか判別ができない

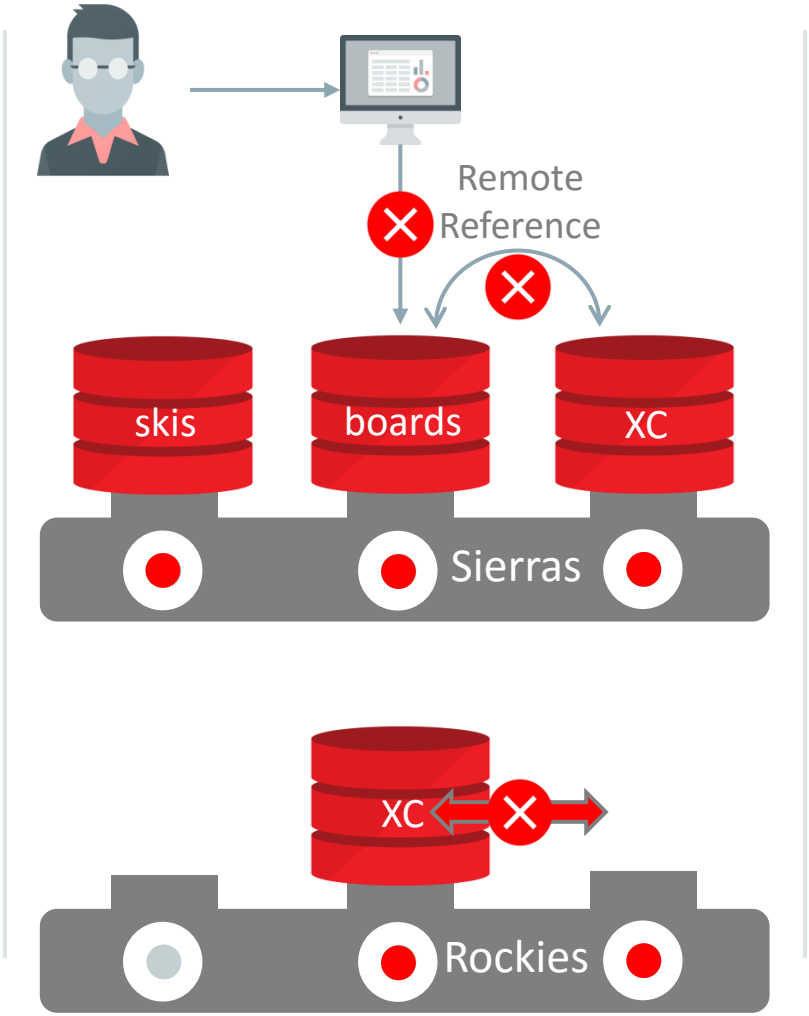
マルチテナント・アーキテクチャ

- **アプリケーションは変更不要**

2パート・ネーミングと3パート・ネーミングの比較

3パート・ネーミング

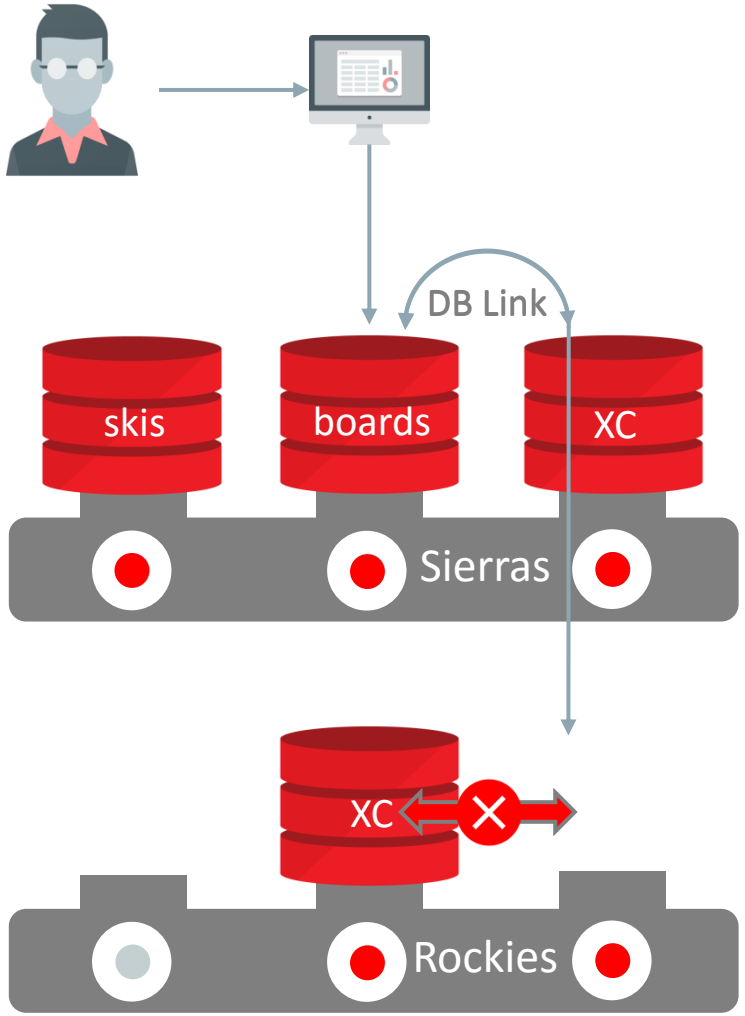
- リモート・オブジェクトの参照 database.schema.object
- データベースの名前が変更された場合 (例: 移行した場合など)、**全てのSQL文の参照先について変更が必要**
- 可搬性の大きな阻害要因



2パート・ネーミングと3パート・ネーミングの比較

3パート・ネーミング

- リモート・オブジェクトの参照 database.schema.object
- データベースの名前が変更された場合 (例: 移行した場合など)、**全てのSQL文の参照先について変更が必要**
- 可搬性の大きな阻害要因



2パート・ネーミング

- リモート・オブジェクトの参照 schema.object@DBLink
- データベースの名前が変更された場合 (例: 移行した場合など)、**1つのDBリンクの定義を変更するだけ**
- 全てのSQL文は変更せずに実行可能
- かなり現実的な可搬性

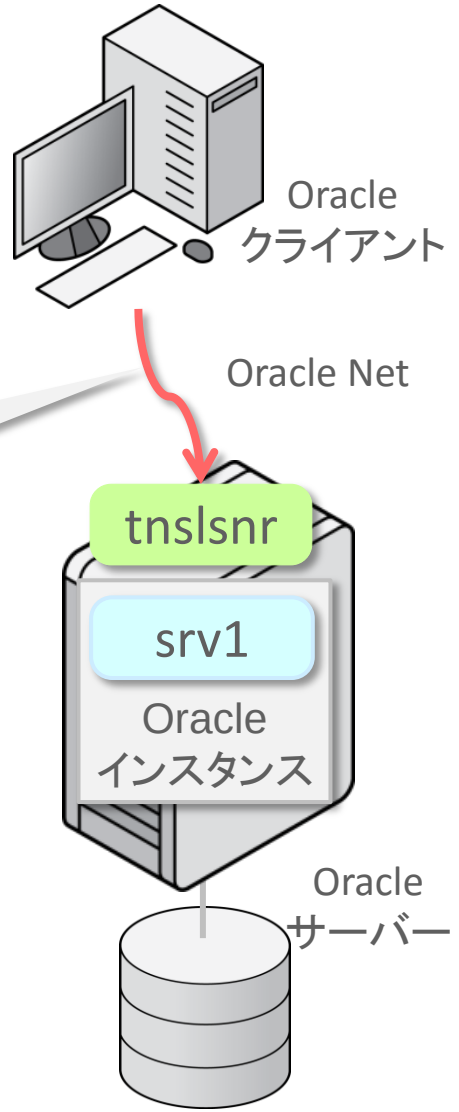
マルチテナント・アーキテクチャにおける接続

従来と同じ5つの要素を使用

- 基本的には、従来のデータベースと同様の方法で接続
  - リスナーが稼働する**サーバー、リスナーのポート、サービス名、ユーザー名、パスワード**
  - PDB 作成時にPDB名と同じ名前のサービスが作成される
  - Oracle クライアントの接続記述子にはサービス名を指定

```
(DESCRIPTION =
  (ADDRESS=(PROTOCOL=TCP)
        (HOST=node01.oracle.jp)(PORT=1521))
      (CONNECT_DATA=(SERVICE_NAME=srv1))
)
```

- PDB ごとに必要なサービスを作成して接続に利用することを推奨
  - 同一サーバー内に複数の CDBが存在する環境では、PDB名の重複 (同一名のサービス) が生じる場合があるため
- ローカル・ユーザーは作成されたPDBに対してのみ接続可能
- 共通ユーザーはCreate Sessionシステム権限をもつPDBに対してセッションを作成可能



データベースにおける接続

SQL*Plusを使用した接続方法

- これまでのデータベースでも提供されていた SQL*Plus を使用した接続方法
  - 接続記述子

```
sqlplus <USERNAME>/<PASSWORD>@<ALIAS>
```

tnsnames.ora

```
<ALIAS>=
(DESCRIPTION =
  (ADDRESS=(PROTOCOL=TCP) (HOST=<HOSTNAME1>) (PORT=<PORT>))
  (ADDRESS=(PROTOCOL=TCP) (HOST=<HOSTNAME2>) (PORT=<PORT>))
  (CONNECT_DATA=(SERVICE_NAME=<SERVICE1>))
)
```

  - EZCONNECT (簡易接続ネーミング)

```
sqlplus <USERNAME>/<PASSWORD>@<HOSTNAME>:<PORT>/<SERVICE_NAME>
```

プラガブル・データベースのリスナーへの登録

リスナーに登録されたサービスの例

```
$ lsnrctl services

LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 09-12月-2016 13:38:28
Copyright (c) 1991, 2016, Oracle.  All rights reserved.
(ADDRESS=(PROTOCOL=TCP)(HOST=)(PORT=1521)))に接続中
サービスのサマリー...
サービス"cdb1"には、1件のインスタンスがあります。
  インスタンス"cdb1"、状態READYには、このサービスに対する1件のハンドラがあります...
                                    <中略>
サービス"hrpdb"には、1件のインスタンスがあります。
  インスタンス"cdb1"、状態READYには、このサービスに対する1件のハンドラがあります...
    ハンドラ:
      "DEDICATED" 確立:30 拒否:0 状態:ready
         LOCAL SERVER30 拒否:0 状態:ready
         LOCAL SERVER
コマンドは正常に終了しました。
```

PDB のサービスに関する情報

コンテナ間における接続先の切り替え

再接続あるいはALTER SESSION文の使用

- コンテナ(ルートあるいはPDB)間における接続先の切り替え
  - SQL*Plus による再接続

```
SQL> CONNECT <USERNAME>/<PASSWORD>@<HOSTNAME>:<PORT>/<SERVICE_NAME>;
```

    - 共通ユーザーおよびローカル・ユーザーで使用可能

  - ALTER SESSION 文による接続

```
SQL> ALTER SESSION SET CONTAINER = <PDB_NAME>;
```

    - 共通ユーザーのみ使用可能
    - 管理作業、または接続プールを利用時
    - 接続先のコンテナに対してSET CONTAINER権限が必要

接続先の確認方法

SHOW コマンドによる確認

- 接続しているコンテナは SHOW コマンドなどで確認可能
  - ルートに接続している場合の出力例

```
SQL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
```

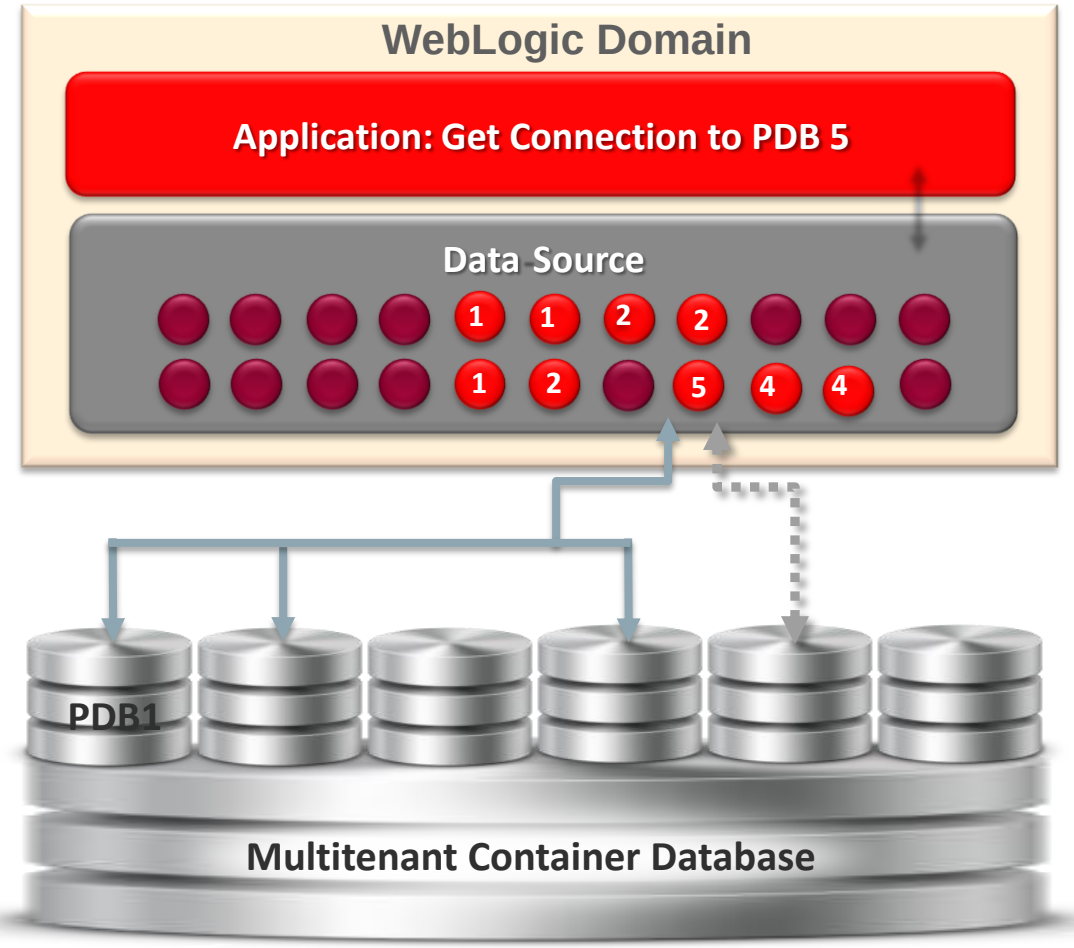
  - PDB に接続している場合の出力例

```
SQL> SHOW CON_NAME

CON_NAME
------------------------------
PDB1
```

コネクション・プール
マルチテナント環境への効率的な接続

マルチテナント・データソース

- 1つのデータソース
- プールから全てのテナント・データベース(PDB)に接続を生成
- 構成
  - グローバル・データベース・ユーザー(どのPDBにもアクセス可能な権限をもつユーザー)
  - UCPによるコネクション・ラベリングとコールバック・インターフェースの利用
  - ALTER SESSION SET CONTAINERによる接続切り替え(アプリケーション・コード側での接続生成が不要)

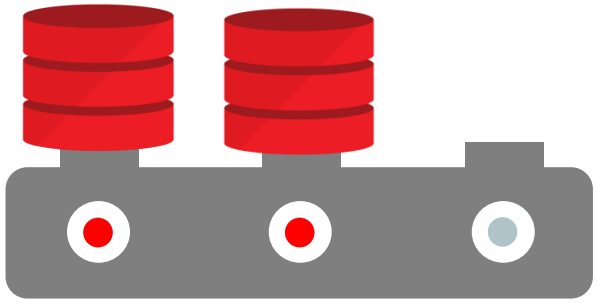


第1部

4. プラガブル・データベースのプロビジョニング

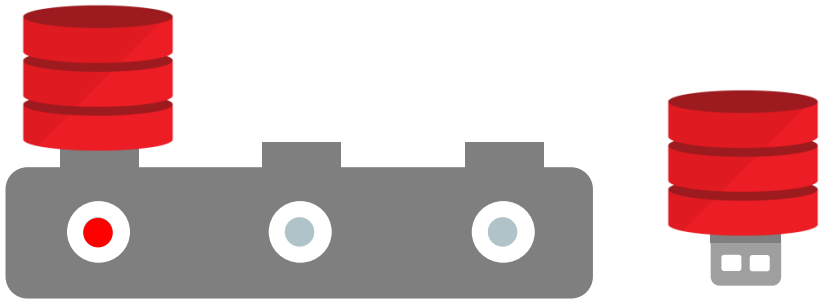
プラガブル・データベースのプロビジョニング

1. Create PDB
2. Clone PDB
3. Unplug PDB

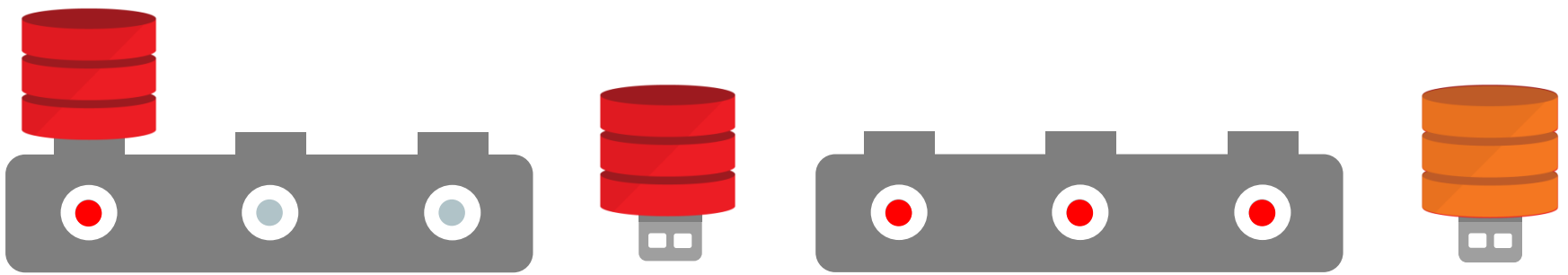


プラガブル・データベースのプロビジョニング

1. Create PDB
2. Clone PDB
3. Unplug PDB



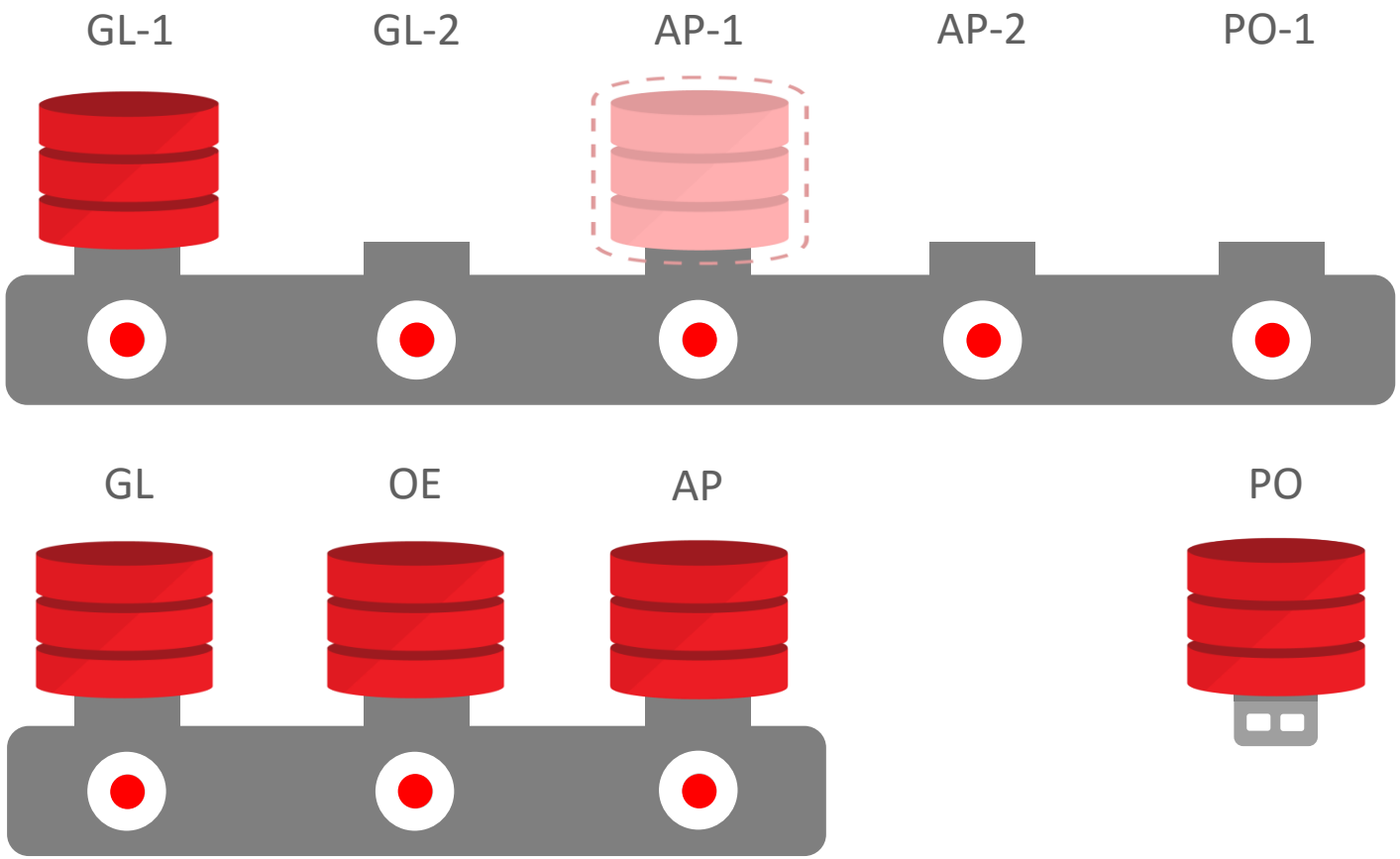
プラガブル・データベースのプロビジョニング



1. Create PDB
2. Clone PDB
3. Unplug PDB
4. Plug In PDB
5. Drop PDB
6. Clone Gold

プラガブル・データベースのプロビジョニング

PDBの高速なクローニング



- ローカルCDB上のPDBからクローン
- リモートCDB上のPDBからクローン
- non-CDBからクローン
- スナップショット・クローン(瞬時にクローン)

プラガブル・データベースの作成

前提条件

- プラガブル・データベースの作成にあたり、次の条件を満たしている必要がある
  - CDBが作成され、READ WRITEモードで起動している
  - 共通ユーザーでルートに接続している
  - 接続ユーザーがCREATE PLUGGABLE DATABASE権限を有している


```
$ sqlplus sys/Welcome1@scan.oracle12c.jp:1521/cdb1 as sysdba
```

プラガブル・データベースの作成

(1)PDB$SEED を使用した作成

- PDB$SEED から PDB を作成する
- 構文

```
CREATE PLUGGABLE DATABASE <PDB_NAME> ADMIN USER <USER_NAME>
IDENTIFIED BY <PASSWORD> [<OPTIONAL_CLAUSE>];
```

  - 例

```
CREATE PLUGGABLE DATABASE pdb1 ADMIN USER admin IDENTIFIED BY Pwd;
```

- 作成する PDB のデータファイル配置場所は、Oracle Managed Files(OMF)や初期化パラメータPDB_FILE_NAME_CONVERTの設定により異なる
- FILE_NAME_CONVERT句を用いて、明示的に指定することも可能

データファイル配置場所の指定

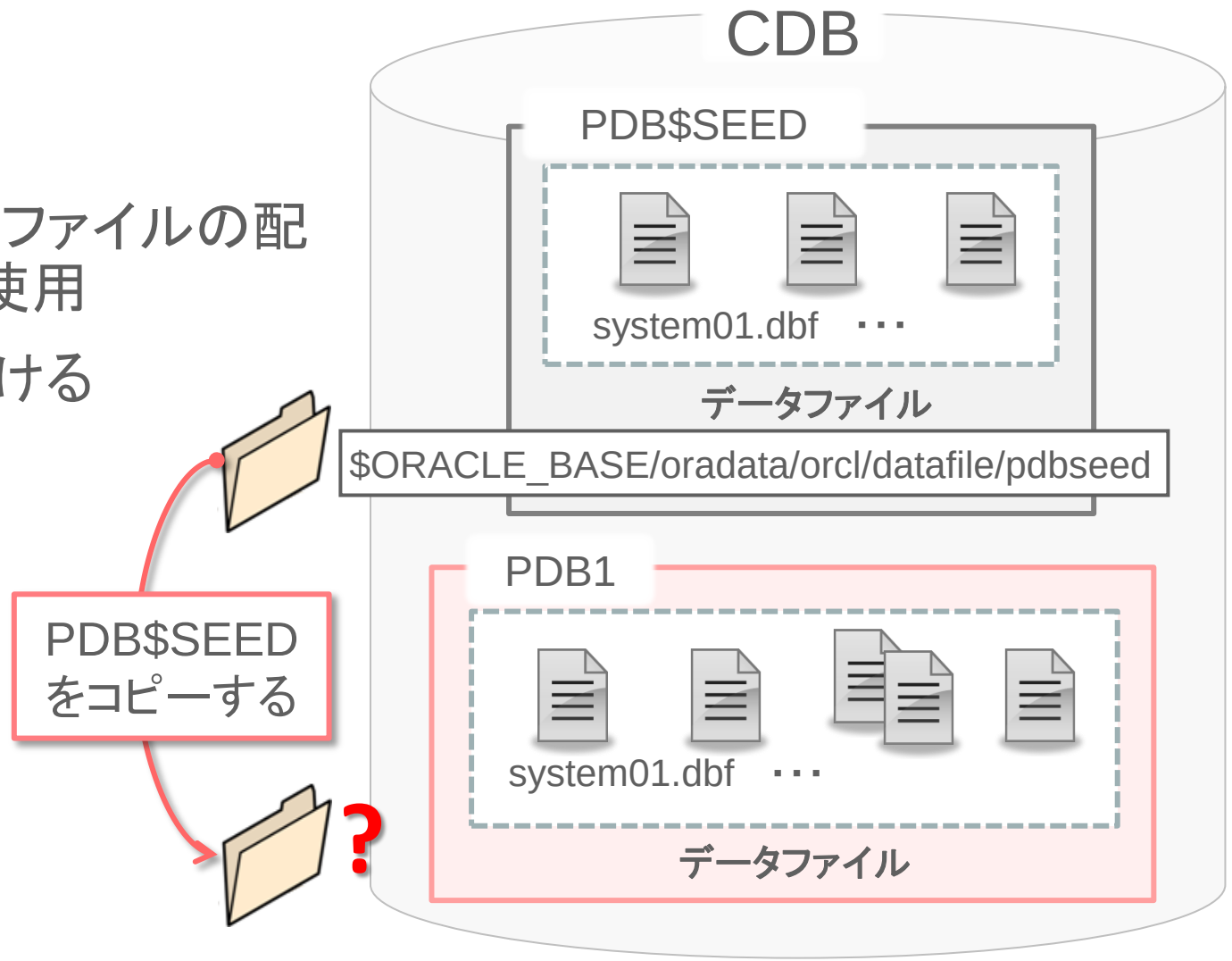
指定方法と優先度

- データファイルの配置場所は、次のいずれかの方法で指定が可能
  1. FILE_NAME_CONVERT句
  2. CREATE_FILE_DEST句
  3. Oracle Managed Files(OMF)
  4. 初期化パラメータPDB_FILE_NAME_CONVERT
- 複数の方法を組み合わせた場合は、上位の方法による指定が適用される
- PDB 関連のデータファイル管理の例
  - CDBをOMF構成で作成し、PDB関連のファイルも基本的にはOMF に準拠して配置する
  - 開発用の PDB など例外的に配置場所を変更したい場合には、作成時に FILE_NAME_CONVERT 句を使用して配置先を変更する

FILE_NAME_CONVERT 句

ファイルの配置場所を指定

- 新規作成する PDB について、データファイルの配置場所を明示的に指定する場合に使用
- ケースに応じて、指定方法を使い分ける
  - ディレクトリ単位での一括指定
  - ファイル単位での指定
  - ディレクトリ単位とファイル単位を組み合わせた指定



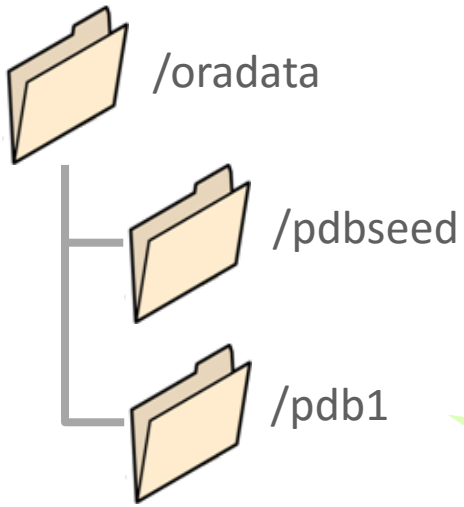
FILE_NAME_CONVERT 句の活用

ディレクトリ単位での配置場所の指定

- ファイルを配置するディレクトリを指定する場合のコマンド例と配置イメージ

```
CREATE PLUGGABLE DATABASE pdb1 ADMIN USER admin IDENTIFIED BY Admin
FILE_NAME_CONVERT=('/oradata/pdbseed','/oradata/pdb1');
```

例



PDB$SEED のファイル群

```
temp01.dbf        system01.dbf
sysaux01.dbf
```

pdb1のファイル群

```
temp01.dbf        system01.dbf
sysaux01.dbf      pdb1_users01.dbf
```

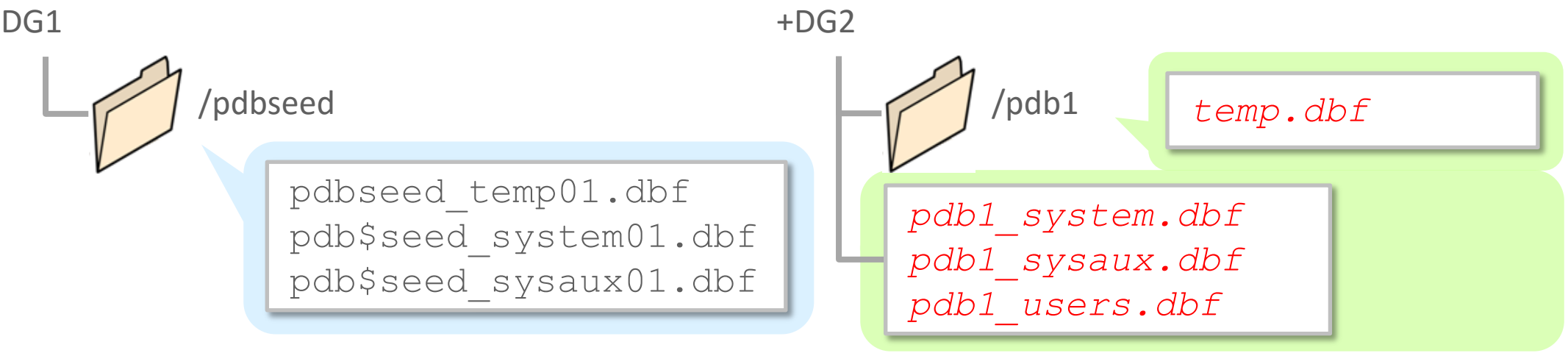
FILE_NAME_CONVERT 句の活用

ファイル単位での配置場所の指定

- ファイルごとに配置を指定する場合のコマンド例と配置イメージ

```
CREATE PLUGGABLE DATABASE pdb1 ADMIN USER admin IDENTIFIED BY Admin
FILE_NAME_CONVERT=(
'+DG1/pdbseed/pdbseed_temp01.dbf','+DG2/pdb1/temp.dbf',
'+DG1/pdbseed/system01.dbf','+DG2/pdb1_system.dbf',
'+DG1/pdbseed/sysaux01.dbf', '+DG2/pdb1_sysaux.dbf');
```

例



プラガブル・データベースの作成

(2)既存のPDB を使用した作成

- 既存のPDBから、新しいPDBを作成する
- 構文

```
CREATE PLUGGABLE DATABASE <TARGET_PDB_NAME> FROM <SOURCE_PDB_NAME>
[<OPTIONAL_CLAUSE>];
```

  - 例

```
CREATE PLUGGABLE DATABASE testpdb FROM hrpdb;
```

- 同一 CDB内 (ローカル)、あるいは異なるCDB間 (リモート)での作成が可能
- 異なるCDB間での作成する場合は、データベース・リンクを使用する
- 12c R1ではソースとするPDBは、読み取り専用(READ ONLY モード)でオープンされている、もしくは処理中のトランザクションがない状態で行う

プラガブル・データベースの作成

(3)既存のnon-CDBからの作成

- 既存のnon-CDBをPDB として作成する(同バージョンの場合)
  - あらかじめ CDB を作成し、リモート・クローンによる作成
  - 構文

```
CREATE PLUGGABLE DATABASE pdb1 FROM NON$CDB@<DBLINK>;
または
CREATE PLUGGABLE DATABASE pdb1 FROM <NONCDB_DB_NAME>@<DBLINK>;
```

  - PDBとして作成後、noncdb_to_pdb.sqlの実行が必要

```
@$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql
```

- その他の作成方法
  - DBMS_PDB パッケージ*DB 12c ~*
  - Oracle Data Pump

DBMS_PDBパッケージを使用した作成

作成手順

- PDBを作成するCDBを作成し、non-CDBをREAD ONLYモードで起動する
- DBMS_PDB.DESCRIBE プロシージャを使用して XML ファイルを作成する
  - non-CDB に対して XML ファイルを生成する場合の実行例

```
BEGIN
  DBMS_PDB.DESCRIBE(
    pdb_descr_file => '/home/oracle/nonCDBtoPDB1.xml');
END;
/
```

- 生成したXMLファイルを使用して、PDBを作成する
- USING句を含むCREATE PLUGGABLE DATABASE文で作成
  - PDB を使用する際には作成後に別途オープンを行う

プラガブル・データベースの作成

既存のnon-CDBからの作成に使用できる方法の一覧

- 使用できる方法は、non-CDBのバージョンによって異なる

作成方法 バージョン	リモート・クローン	DBMS_PDB パッケージ	Oracle Data Pump		
			トランスポータブル・データベース	トランスポータブル表領域	Export / Import
12.1.0.2以降	○ 対応	○ 対応	○ 対応	○ 対応	○ 対応
12.1.0.1	N/A	○ 対応	○ 対応	○ 対応	○ 対応
11.2.0.3 以降	N/A	N/A	○ 対応	○ 対応	○ 対応
11.2.0.3 より前	N/A	N/A	N/A	○ 対応	○ 対応

Non-CDBを最新のバージョンすることで移行の選択肢が広がる

プラガブル・データベースの作成

(4)既存のPDBのアンプラグ/プラグによる作成

- 既存の PDB をアンプラグ(取り外し)とプラグ(取り付け)することによる作成
  - 関連ファイル群の位置情報を含む XML ファイルを生成して作成に使用する
- 構文

アンプラグ

```
ALTER PLUGGABLE DATABASE <PDB_NAME> UNPLUG INTO <FILE_LOCATION>;
```

プラグ

```
CREATE PLUGGABLE DATABASE <PDB_NAME> [AS CLONE] USING <FILE_LOCATION>
[<OPTIONAL_CLAUSE>];
```

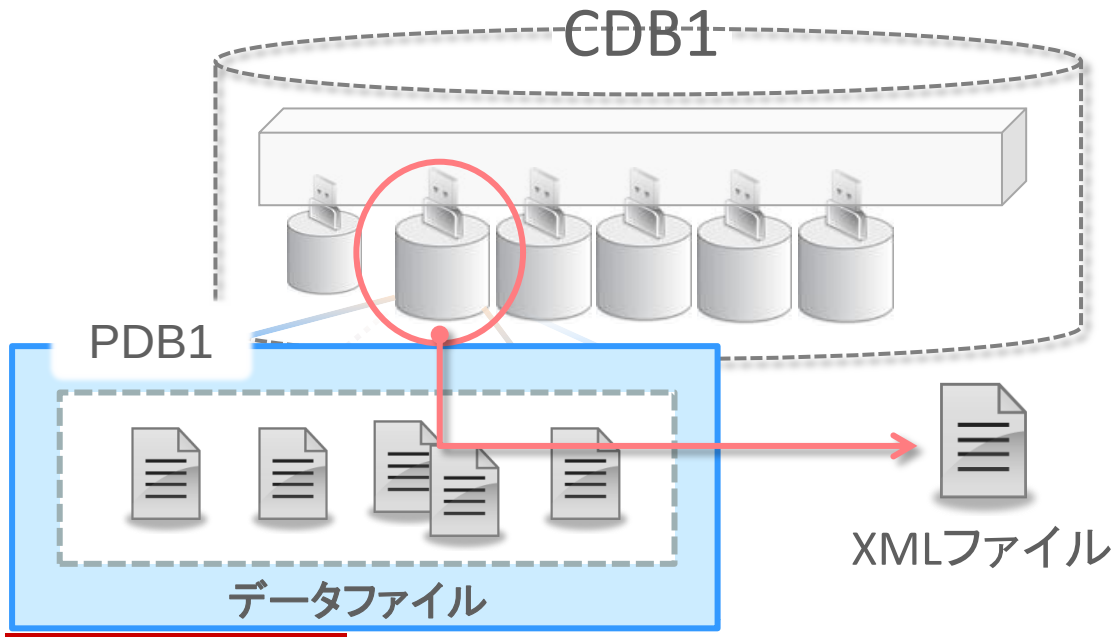
  - 例 (アンプラグの場合)

```
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO '/opt/oracle/pdb1.xml';
```

プラガブル・データベースのアンプラグ

PDBの切断とXMLファイルの作成

- アンプラグ操作ではPDBをCDBから切り離し、XMLメタデータ・ファイルを作成する
- コマンドラインあるいは DBCA などのツールを使用可能
  - コマンドラインの場合は ALTER PLUGGABLE DATABASE 文を使用する



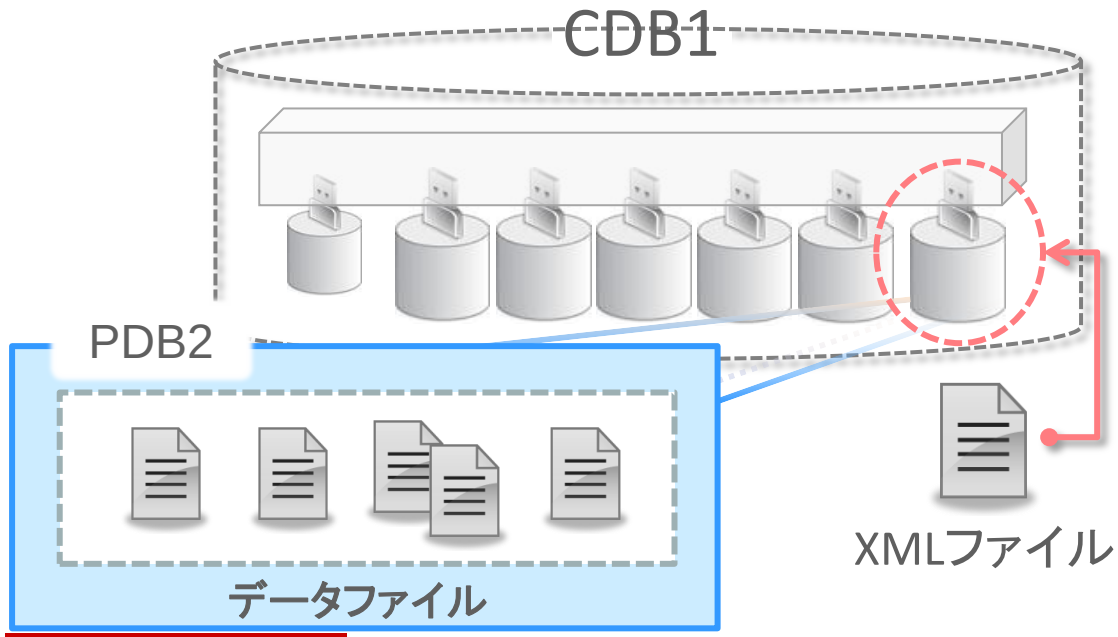

```
ALTER PLUGGABLE DATABASE pdb1 UNPLUG
INTO '/opt/oracle/pdb1.xml';
```

1 アンプラグ時にXMLファイルを生成する

プラガブル・データベースのプラグ

XMLファイルを使用したPDBの作成

- プラグ操作では、アンプラグ時に作成したXMLメタデータ・ファイルを使用する
- コマンドラインあるいはDBCAなどのツールを使用可能
  - コマンドラインの場合はCREATE PLUGGABLE DATABASE文を使用する




```
CREATE PLUGGABLE DATABASE pdb2
USING '/opt/oracle/pdb1.xml';
```

2 プラグ時にはXMLファイルの情報を使用して PDBを作成する

プラガブル・データベースのアンプラグとプラグ

ステータスの確認

- CDB_PDBS表からステータスを確認可能
  - アンプラグしたPDBはUNPLUGGED として表示する
  - プラグしたPDBはNEWとして表示する
    - 一度でもオープンしたPDBのステータスは NORMAL と表示される
    - 下記は PDB2 をアンプラグして、PDB3 をプラグした場合の表示例

```
SQL> SELECT PDB_NAME, STATUS FROM CDB_PDBS;

PDB_NAME                       STATUS
------------------------------ ------------------------------
PDB$SEED                       NORMAL
PDB1                           NORMAL
PDB2                           UNPLUGGED
PDB3                           NEW
```

プラガブル・データベースのアンプラグとプラグ

- ソースとターゲットのプラットフォームは、次の要件を満たしている必要がある
  - endiannessが同じ
  - 同じセットのデータベース・オプションがインストールされている
    - ターゲットのデータベースにインストールされているオプション(コンポーネント)(*1)に対して、ソースのデータベースが(*1)と同じまたはサブセットである場合も可能

    => Windows(X86_64)で稼働するCDB上のPDBを、Linux(X86_64)で稼働するCDB上にプラグできる

    => Standard EditionのCDB上のPDBを、Enterprise EditionのCDB上にプラグできる
- Application Express(APEX)の構成が異なる場合も同じである必要がある
  - インストールされているか
  - インストールされている場合はバージョン
- プラグ時に非互換性が確認された場合はPDB_PLUG_IN_VIOLATIONSビューで確認可能
  - DBMS_PDB.CHECK_PLUG_COMPATIBILITYプロシージャを利用して互換性を事前に確認可能

プラガブル・データベースの削除

DROP PLUGGABLE DATABASE文による削除

- 既存の PDB をデータベースから削除する
- 構文

```
DROP PLUGGABLE DATABASE <PDB_NAME> [<OPTIONAL_CLAUSE>];
```

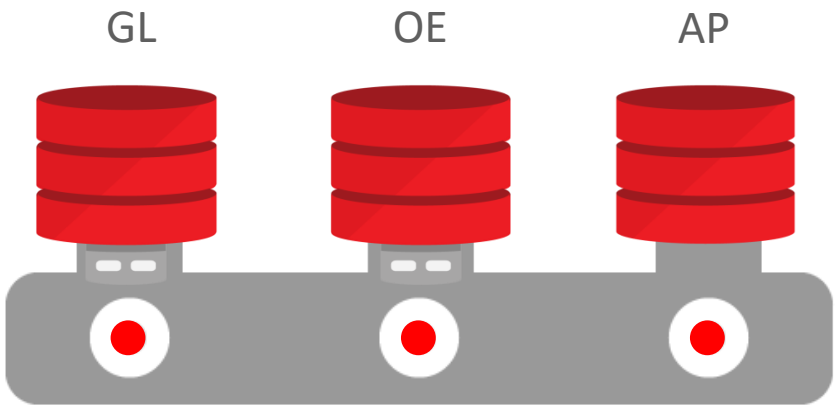
  - 例

```
DROP PLUGGABLE DATABASE pdb1 INCLUDING DATAFILES;
```

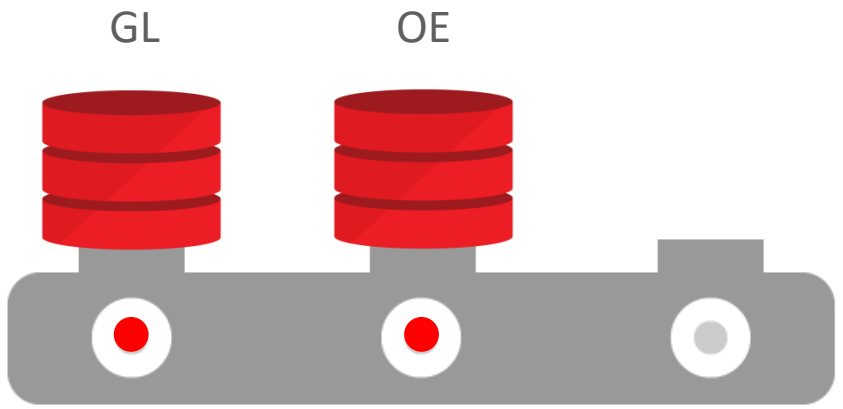
- コマンドでの削除は PDB をクローズしておく(オープン中の削除操作は不可)
- 削除としては、制御ファイルにリストされているデータファイルの削除を実行

マルチテナント・アーキテクチャのパッチ適用とアップグレード

データベースのパッチ適用とアップグレードの柔軟な選択が可能



<現在、運用中> Container Database 12.1

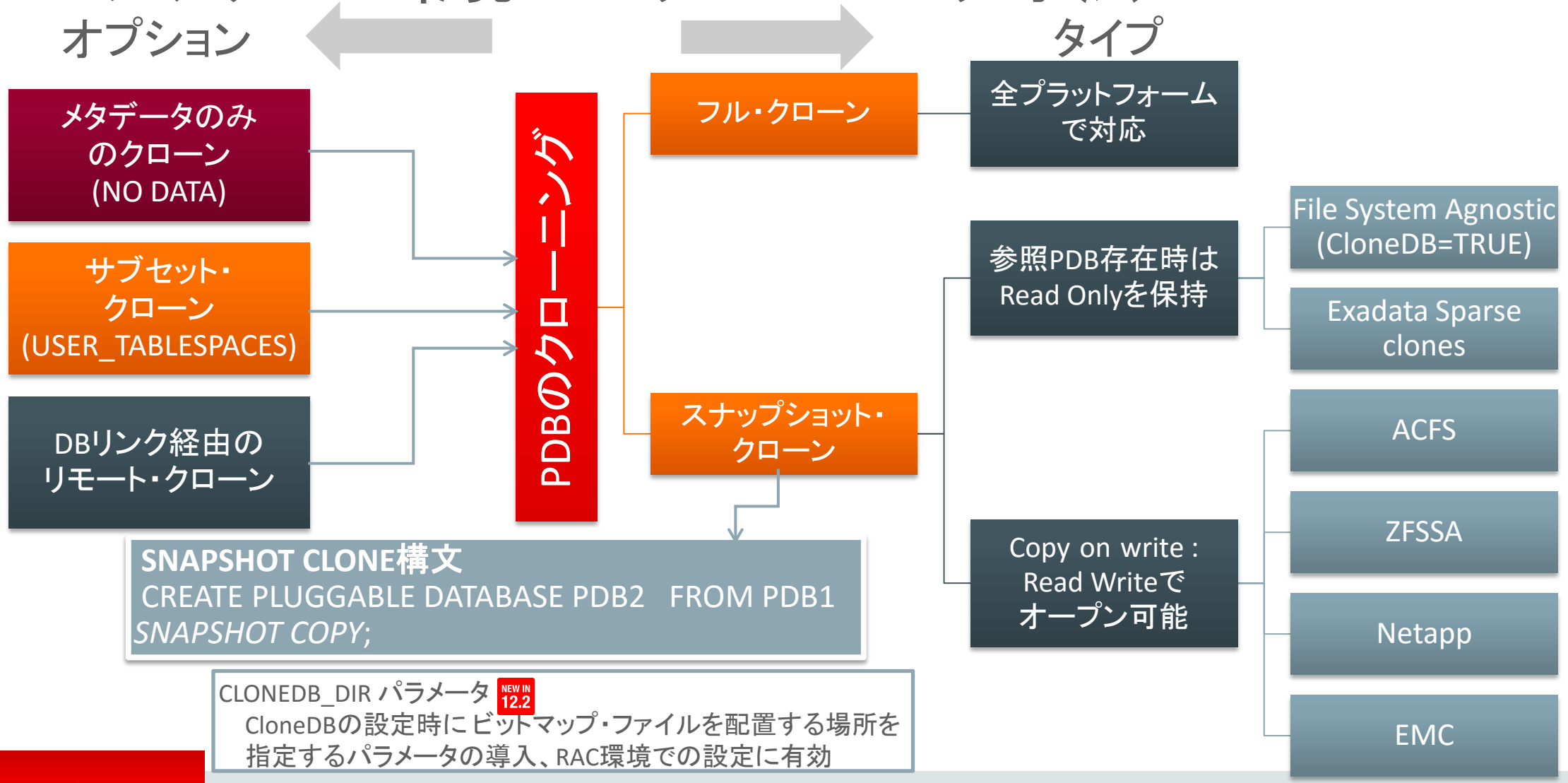


<アップグレード済み> Container Database 12.2

プラグ後にdbupgrade ユーティリティを実行

```
$ dbupgrade -c pdb1
```

マルチテナント環境でのクローニング手法

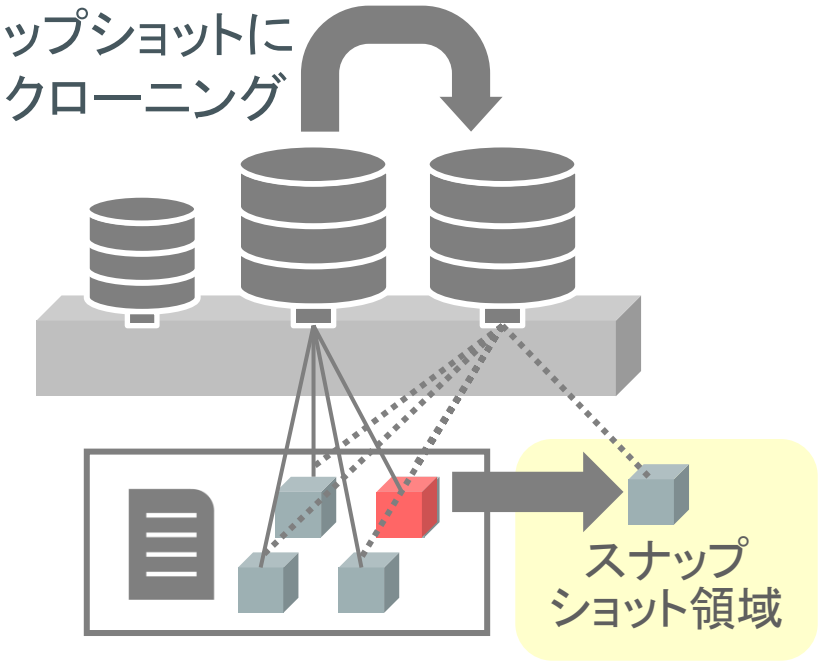


スナップショットを利用したクローニング

- スナップショットを用いたPDBのクローニング
  - 構文

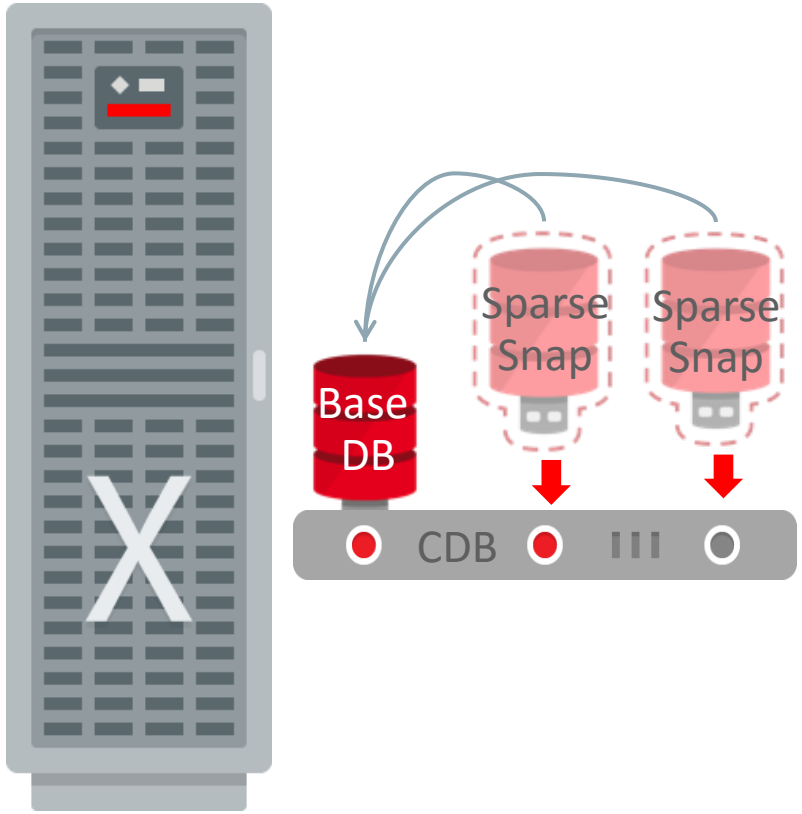
```
SQL> CREATE PLUGGABLE DATABASE <NEW_PDB> FROM <SOURCE_PDB> SNAPSHOT COPY;
```

- コピー・オン・ライト方式により作成時はブロックへのポインタのみを記録
  - 短時間でのクローニングが可能
  - 必要なディスク容量の削減が期待できる
- データ更新時には、更新を実行する前に
  該当ブロックをスナップショット領域へコピー
- 開発やテスト環境でのPDBクローニングに便利
  - PDBの利用期間が短いが多くのクローンが必要、
    またデータの変更が少ないような場合



Exadata上の高速なデータベース・スナップショット

- 高速で**ストレージ容量効率のよいスナップショット**による
  データベース作成
  - スパース・ディスク・グループをExadataストレージ上で作成
  - Copy-on-WriteでスナップショットDB/PDBを作成
- PDBとの統合により、スナップショットPDBは
  1コマンドまたは1クリック(EM)で作成可能
- **全てのExadataの機能**がスナップショット上で動作
  スマート・スキャン、スマート・フラッシュ・キャッシュ、リソース管理…
- I/Oとリソースの優先度設定が全データベース間で有効



特定の表領域のみをコピーするサブセット・クローン

- 既存データベースの表領域を指定してPDBを作成する
  - 構文

```
SQL> CREATE PLUGGABLE DATABASE pdb1 USING '/tmp/noncdb.xml' copy
     USER_TABLESPACES = 'usertbs01,usertbs03' TEMPFILE REUSE;
```

- USER_TABLESPACES句による指定
  - SYSTEM、SYSAUX、TEMP表領域は指定できない
  - ユーザー定義の表領域はカンマ区切りで複数指定することが可能
  - 指定しなかった表領域はOFFLINEとして表示される

メタデータのみのクローン

- データ・ディクショナリのみを対象にPDBのクローニングを実行する
  - 構文

```
SQL> CREATE PLUGGABLE DATABASE pdba FROM pdb1 NO DATA;
```

- NO DATA句は PDBのクローニング時のみ指定可能
- SYSTEMおよびSYSAUX表領域に含まれるユーザー・データは対象外
- PDBに以下のタイプの表を含む場合は実行できない
  - 索引構成表、キュー表、クラスタ表等

PDBのアンプラグ/プラグによるサービス・レベルの向上

データベース構成が異なるCDBへ迅速で容易な移行

休憩

第2部

DB12c R2新機能で広がるユースケース

第2部:DB12c R2新機能で広がるユースケース
Agenda

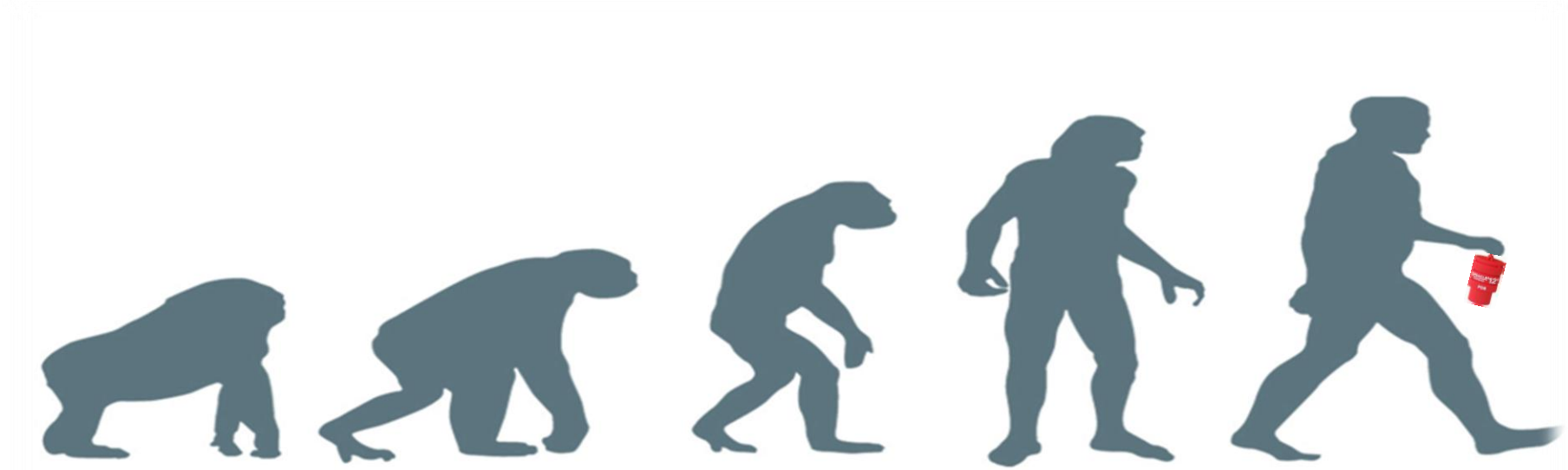
1. プロビジョニング機能の強化
2. PDBの独立性と管理機能の向上
3. データベース統合手法による比較
4. Oracle Multitenantによる統合で広がるユースケース
5. まとめ

第2部

1. プロビジョニング機能の強化

オンライン操作の拡充

PDBクローンの進化



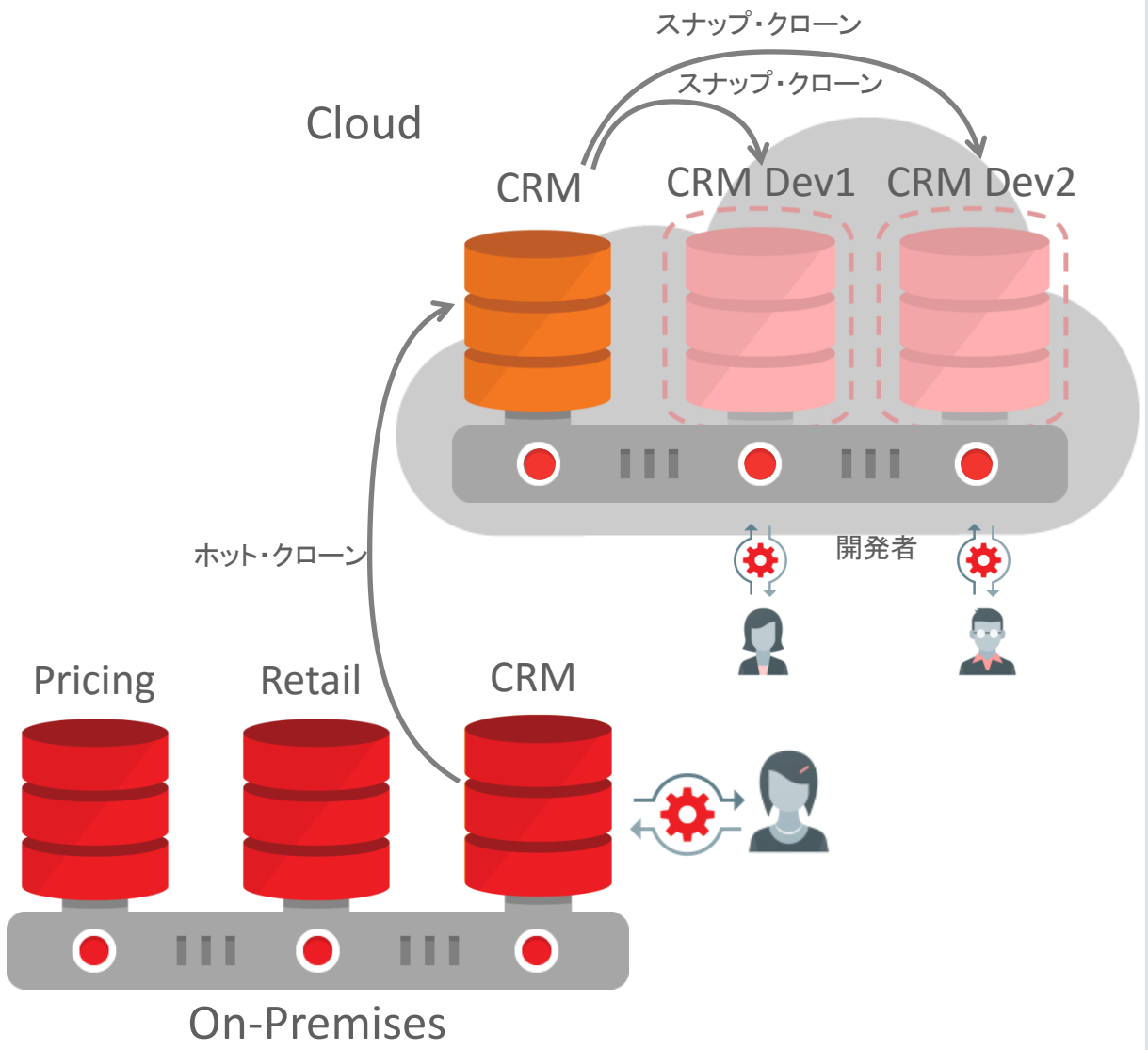
クローン元PDBが
読取り専用 –
コールド・クローン/
リモート・クローン

クローン元PDBが
読取り/書込み可能 –
ホット・クローン/
リフレッシュ・クローン

オンライン再配置

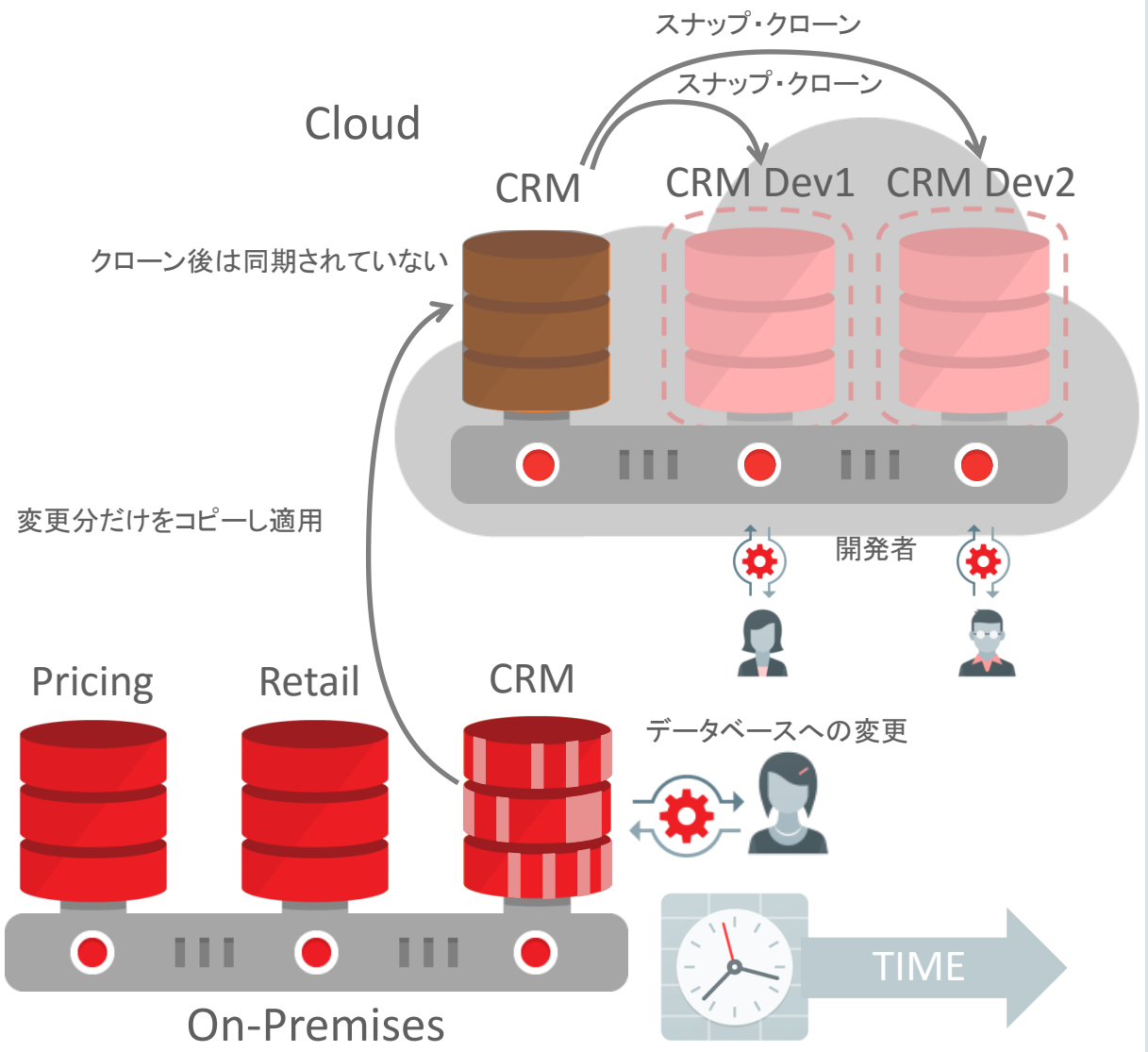
PDBホット・クローン

- PDBホット・クローン
  - オンラインでテスト・マスターを作成



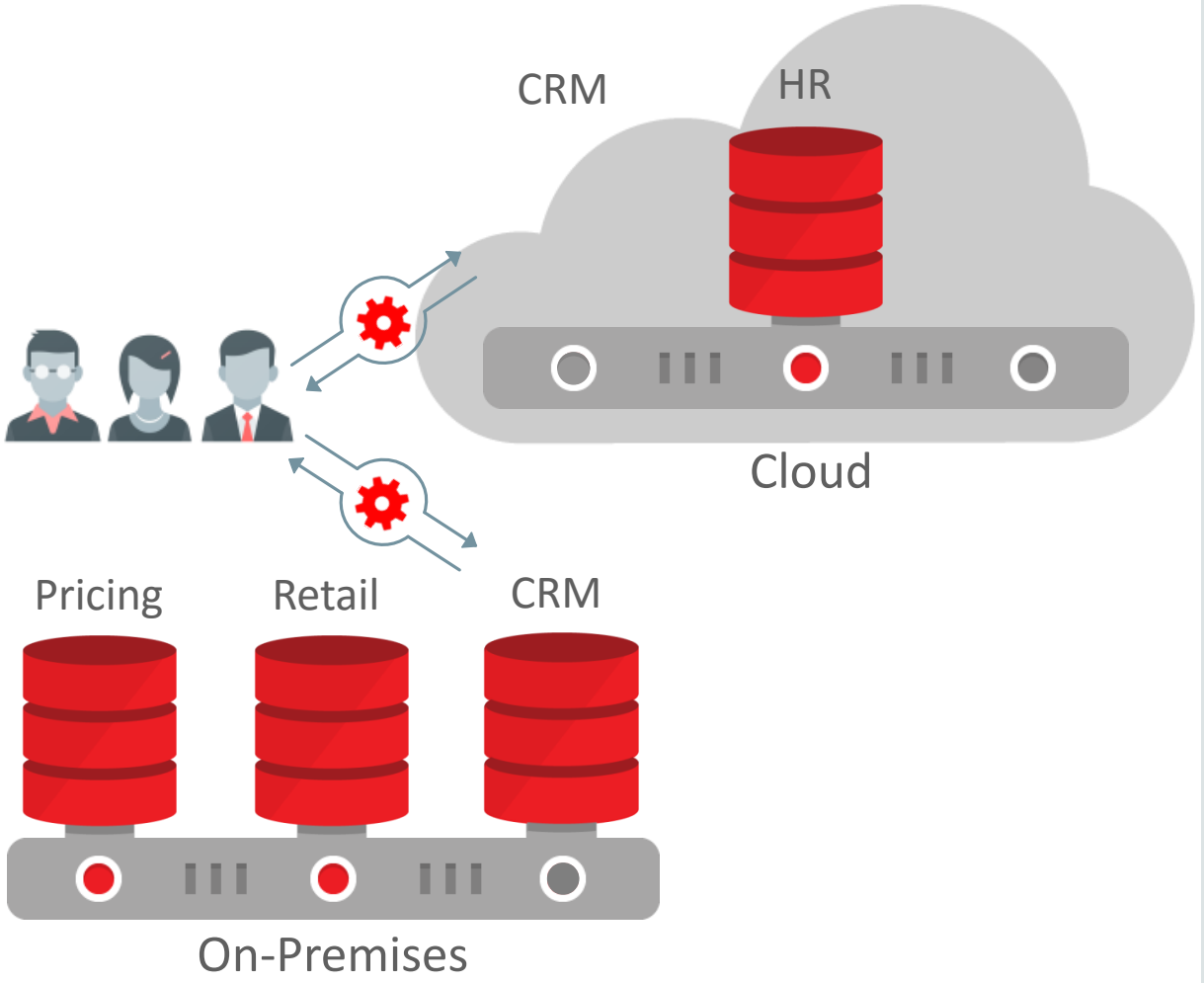
PDBリフレッシュ

- PDB Hot Clone
  - オンラインでテスト・マスターを作成
- PDBリフレッシュ
  - 最新データによって既存のクローンを増分リフレッシュ



PDB再配置

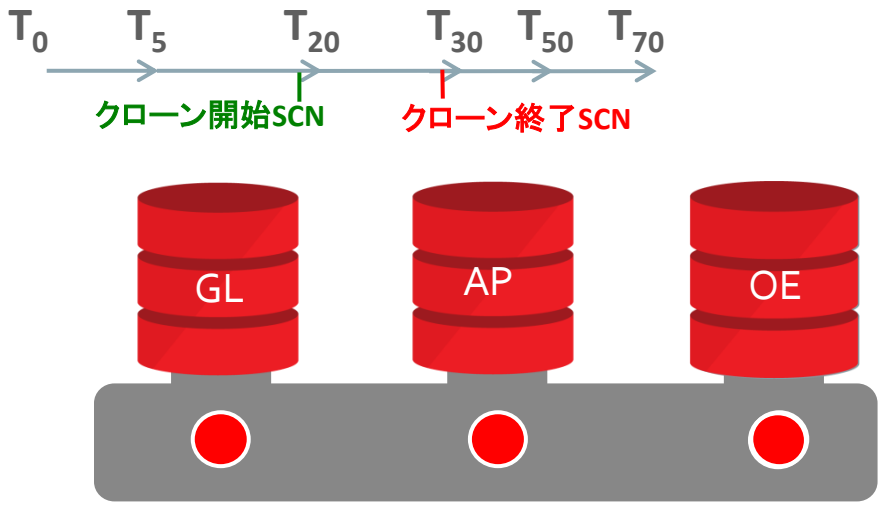
- PDB Hot Clone
  - オンラインでテスト・マスターを作成
- PDB Refresh
  - 最新データによって既存のクローンを増分リフレッシュ
- PDB再配置
  - ダウンタイム無しでPDBを再配置



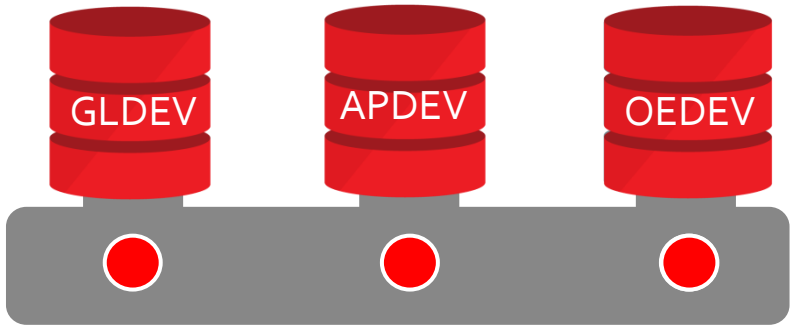
ホット・クローン

PDBホット・クローン

PRODUCTION



DEVELOPMENT



T_{20} 1. *create pluggable database oedev from oe@dblink;*

REDOの最終コピーとロールバック T_{30}

T_{30} 2. *alter pluggable database oedev open;*

PDBホット・クローン – 設定と実行手順

クローン元のPDBが稼働するCDB(ソース)の構成を確認

- アーカイブ・ログ・モード
- ローカルUNDOモード

ソース側で共通ユーザーを作成し、リモートPDBのクローニングを行うための権限を付与

```
SQL> create user c##admin identified by <password> container=all;
SQL> grant create session, sysoper to c##admin container=all;
```

PDBをクローンするCDB(ターゲット)側でリモート・クローニングを行うためのデータベース・リンクを作成

```
SQL> create public database link dblink connect to c##admin
     identified by <password> using '<tns alias>';
```

ターゲット側でホット・クローンの実行

```
SQL> create pluggable database oedev from oe@dblink;
```

PDBクローン時のデータ・ファイルのコピー

- PDBクローン時のデータ・ファイルのコピーは内部的に処理
  - 並列処理、セグメント化されたファイルコピー処理
    - デフォルトの並列度はCPU数
    - *create pluggable database mypdb admin user admin identified by admin* parallel 8;
  - ファイルの転送時間は、ネットワークのレイテンシーとバンド幅に依存
- ファイル・コピーの進捗はv$session_longopsから確認可能:

```
SQL> select opname, message from v$session_longops
OPNAME                MESSAGE
--------------------  ----------------------------------
kpdbfCopyTaskCbk      kpdbfCopyTaskCbk: /u01/app/oracle/oradata/cdb1/CDB : 904448 out of 904448 Blocks done
kpdbfCopyTaskCbk      kpdbfCopyTaskCbk: /u01/app/oracle/oradata/cdb1/CDB : 904448 out of 904448 Blocks done
..
```

対応するOPNAMES:
kpdbfCopyTaskCbk (データファイルのコピー)
kcrfremnoc (REDOファイルのコピー)

PDBホット・クローン

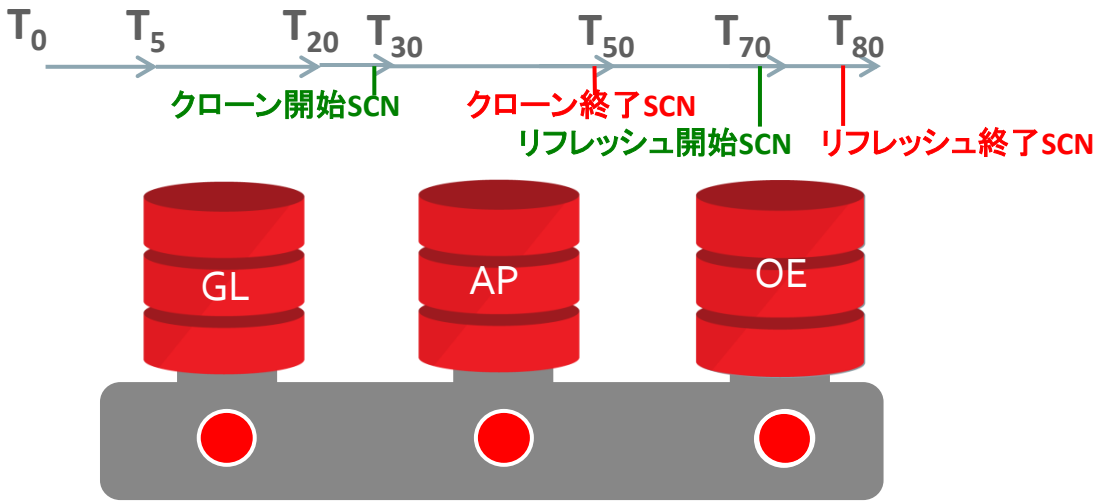
- 構成
  - 同じCDB上でもPDBホット・クローンが可能
  - 異なるCDBへクローンを行う場合、データベース・リンクを使用
    - 新しくPDBが作成されるターゲット側のCDBから、クローン元であるソース側のCDBまたはクローン対象のPDBに対してデータベース・リンクを作成
  - 同じエンディアンであれば、異なるプラットフォームでもPDBホット・クローン可能
    - Windows (x86_64) => Linux (x86_64)

    など
- 優位性
  - 継続的にクローン元のPDBでのアプリケーションの稼働を可能とする
  - クローン元データベースへの影響を最小化
  - RDBMSに統合
    - 3rdパーティのソフトウェアは不要
  - アプリケーション開発とリリースまでの時間を短縮
  - データベースのプロビジョニング・コストを削減

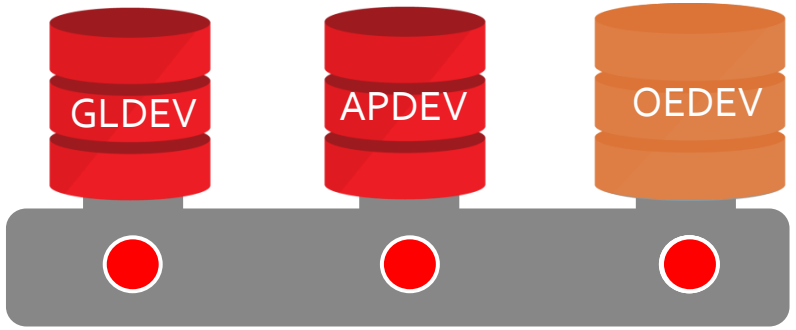
PDBリフレッシュ

PDBリフレッシュ – 自動モード

PRODUCTION



DEVELOPMENT



1. *create pluggable database oedev from oe@dblink* ***refresh mode every 360 minutes****;*

REDOの反復コピーとロールバック T_{80}

リフレッシュ時はPDBをクローズ

T_{50} 2. *alter pluggable database oedev open read only;*

PDBリフレッシュ – 設定と実行手順

PDBホット・クローンの設定

ターゲット側CDBでリフレッシュ可能なクローン用マスターPDBを作成

手動リフレッシュ

```
SQL> create pluggable databse oedev from oe@dblink refresh mode manual;
```

自動リフレッシュ

```
SQL> create pluggable databse oedev from oe@dblink refresh mode every N minutes;
```

PDBを読取り専用(read only)でオープン
マスターPDBを基にしてクローンを実施可能

```
SQL> alter pluggable database oedev open read only;
```

クローン用マスターPDBをクローズし、PDBリフレッシュの実行

```
SQL> alter pluggable database oedev close;
```

手動リフレッシュ:PDB内でリフレッシュを実行

```
SQL> alter session set container=oedev;
SQL> alter pluggable database oedev refresh;
```

PDBリフレッシュ

- リフレッシュのソースとターゲットは異なるCDB上で設定
  - 同じCDB上でのリフレッシュ可能PDBを作成は不可
- リフレッシュ可能PDBを自動リフレッシュで作成した場合も、手動でリフレッシュ可能
- 自動リフレッシュの最短インターバルは1分間隔
- 手動リフレッシュと自動リフレッシュの変更、インターバル(自動リフレッシュ)の変更が可能
  - ALTER PLUGGABLE DATABASE文で変更
- REMOTE_RECOVERY_FILE_DESTパラメータ
  - ソースのアーカイブ・ログがアクセス可能でない場合、リフレッシュ時に参照する異なるディレクトリを指定することが可能

PDBリフレッシュ

- リフレッシュ実行時は対象のリフレッシュ可能PDBをクローズしておく
  - クローズしていない場合の動作
    - 手動リフレッシュ: エラーが返る
    - 自動リフレッシュ: リフレッシュが実行されない、次回の自動リフレッシュのタイミングまで実施されない

```
SQL> alter pluggable database refresh;
alter pluggable database refresh
行1でエラーが発生しました。:
ORA-65025:
プラガブル・データベースOEDEVはすべてのインスタンスでクローズしていません。
SQL> shutdown
プラガブル・データベースがクローズされました。
SQL> alter pluggable database refresh;
プラガブル・データベースが変更されました。
```

- リフレッシュ可能PDBを通常のPDBに変更可能
  - 一旦、リフレッシュを無効(NONE)にした場合は、リフレッシュ可能PDBには変更は不可

```
SQL> alter pluggable database oedev open;
alter pluggable database oedev open
行1でエラーが発生しました。:
ORA-65341: cannot open pluggable database in read/write
   mode
SQL> alter pluggable database refresh mode none;
プラガブル・データベースが変更されました。
SQL> alter pluggable database refresh mode manual;
alter pluggable database refresh mode manual
行1でエラーが発生しました。:
ORA-65261: プラガブル・データベースOEDEVはリフレッシュに対応していません
```

PDBリフレッシュ

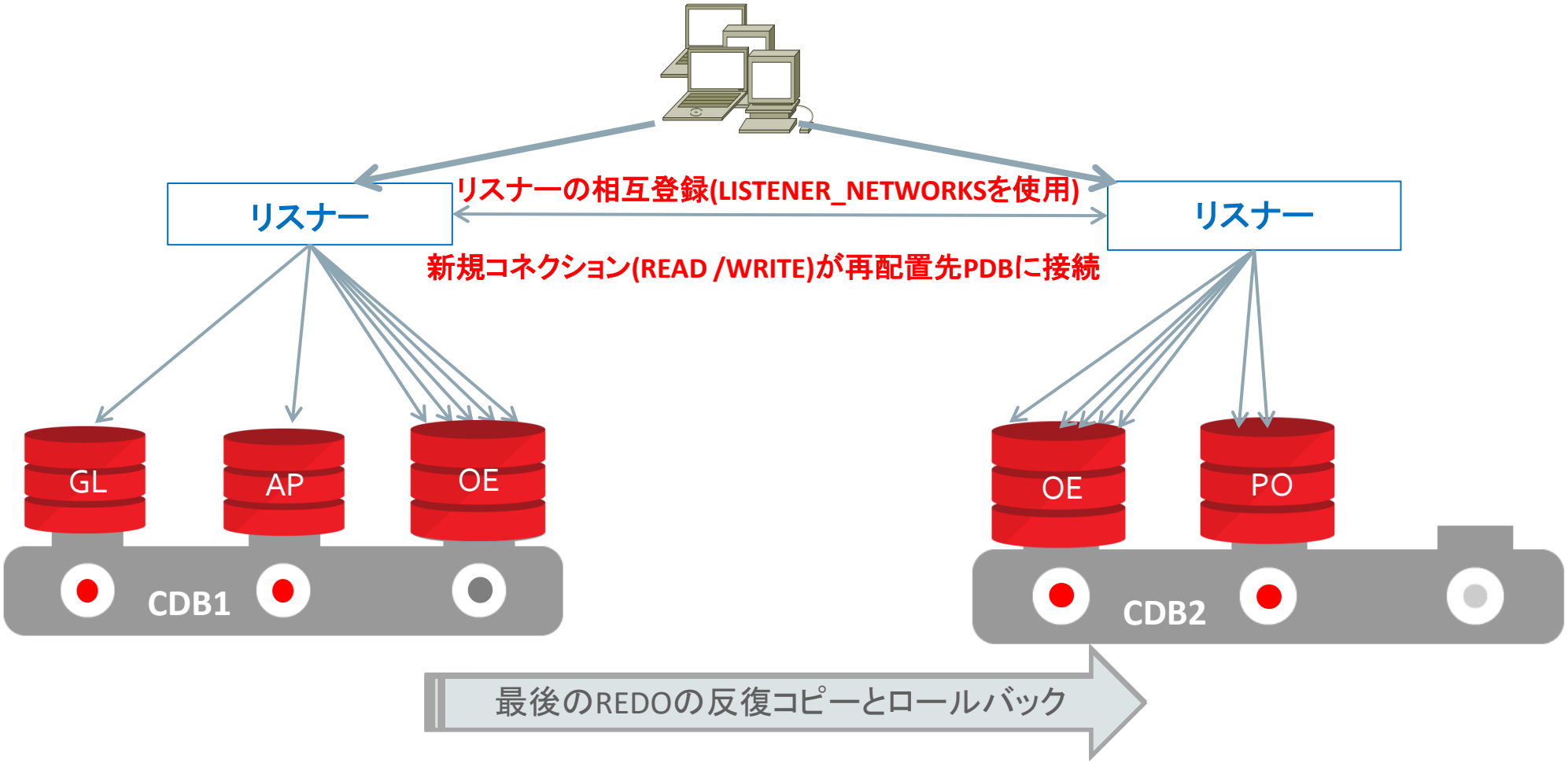
- 優位性
  - 継続的にクローン元のPDBでのアプリケーションの稼働を可能とする
  - クローン元データベースへの影響を最小化
  - RDBMSに統合
    - 3rdパーティのソフトウェアは不要
  - アプリケーション開発とリリースまでの時間を短縮
  - データベースのプロビジョニング・コストを削減
  - クローン元PDBとの差分リフレッシュによる軽い処理
  - 時間粒度の細かいクローニング
    - 2つのモード - 手動と自動
    - Oracleのスケジューラー・ジョブとして事前定義

PDB再配置

PDB再配置

- データベースが再配置してもアプリケーションを継続利用可能
  - クライアントからの処理要求(read/write)への影響を最小化
  - 再配置元サーバーとネットワークへの影響を最小化
    - 仮想マシン(VM)によるマイグレーションより非常に優位
- アプリケーションの変更は不要
- 接続設定の変更も不要
- 最小限のダウンタイムでサーバー側のロード・バランスを実施
- データベースの運用コストを削減
- 2つの再配置モード
  - クライアントからの接続の転送をクライアント側にて制御 (availability normal)
  - クライアントからの接続の転送をサーバー側にて制御 (availability max)

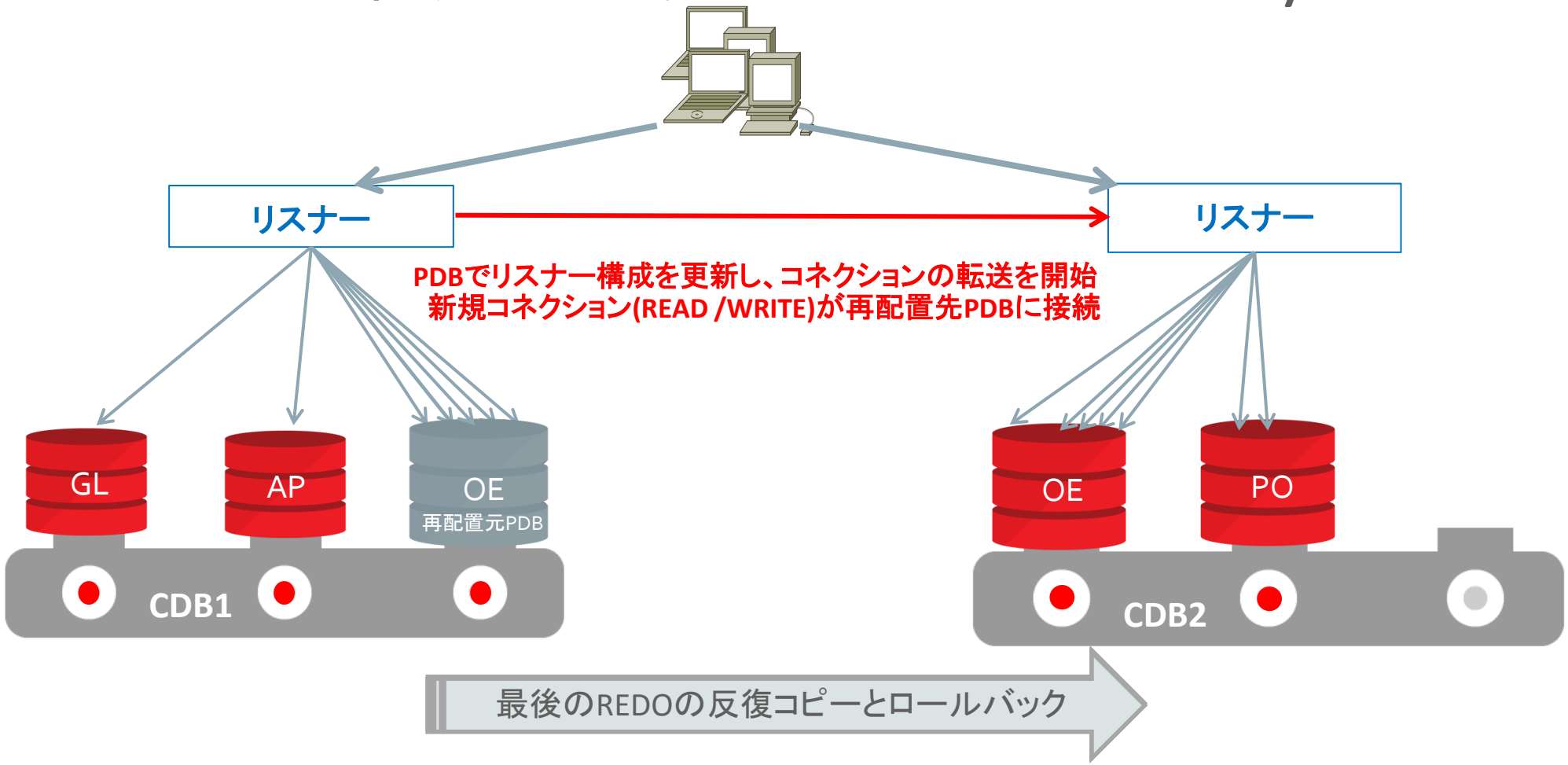
PDB再配置:
相互のリスナーに登録しているケース



create pluggable database OE from OE@CDB1_dblink relocate;

alter pluggable database OE open;

PDB再配置
リスナーによる転送を行うケース – availability max



create pluggable database OE from OE@CDB1_dblink relocate availability max;

alter pluggable database OE open;

PDB再配置 – 設定と実行手順

PDBホット・クローンの設定

PDB再配置のオプション(relocate / relocate availability max)の検討

- ネットワーク構成、クライアントの接続状況

再配置先のCDBでPDB再配置の開始

Availability Normalオプション (省略化)

```
SQL> create pluggable database oe from oe@dblink relocate;
```

Availability Maxオプション: コネクションのリダイレクト

```
SQL> create pluggable database oe from oe@dblink relocate availability max;
```

留意事項:
データベース・リンクはターゲット側のCDBから、ソース側のCDBに対して作成
PDBに対してではないことに注意
ホット・クローン/リフレッシュはソース側のCDB/PDBのいずれでも可

再配置先のCDBでPDBを起動

```
SQL> alter pluggable databse oe open;
```

Availability Maxオプション指定時は、全てのクライアントの接続設定を更新してから再配置元のPDBを削除

PDB再配置 + アプリケーション・コンティニュイティ

ゼロ・ダウンタイムでPDBを移動

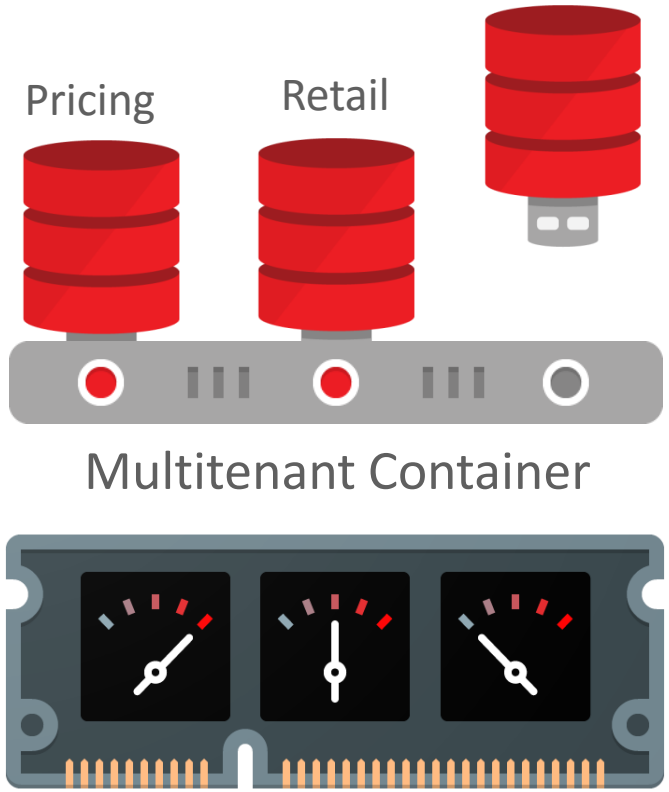
第2部

2. PDBの独立性と管理機能の向上

統合における障壁を排除

規模の経済性と独立性の両立

- CDBあたり4kPDB
  (4,096 : 252(12c R1)から増加)
- メモリー、I/Oリソースの制御
  (CPUに加えて、拡張)
- ロックダウン・プロファイルによる隔離構成
- PDBレベルのフラッシュバック
- PDBごとのキャラクタ・セットのサポート
- PDBレベルのアラート、トレース、AWR
- Data Guard Brokerによる
  PDBレベルのフェイルオーバー機能



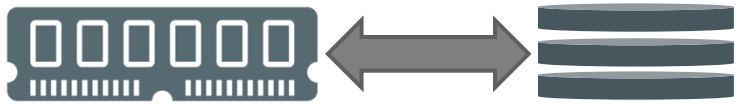
リソース制御

PDBレベルのCPU、メモリー、I/Oリソースの制御

Multitenantにおけるリソース・マネージャーの拡張

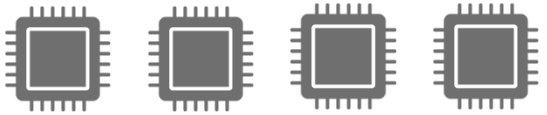
メモリー管理

- 強く要望された機能
  - 12.1では未実装
- PDB単位でメモリー・パラメータの設定が可能
- 新規パラメータ:SGA_MIN_SIZE
  - PDB単位のメモリー分割
  - 集約度の低い、重要なコア・アプリケーション向け
  - その他のシステムでは使用すべきではない



コモディティ・サーバー上のI/O管理

- 2つの新規PDBレベル・パラメータ
  - MAX_IOPS / MAX_MBPS
  - 動的に変更可能
- PDBでのみ設定可能
  - CDB$ROOTでは設定できない
  - **Exadata上のPDBは対象外**
- 12.1ではIORMはExadata storageでのみ可能
  - Exadata IORMはより柔軟
    - シェアにもとづく自動調整
    - DBA は具体的な数値を使用せずに、IOPS とMBPSを調整可能

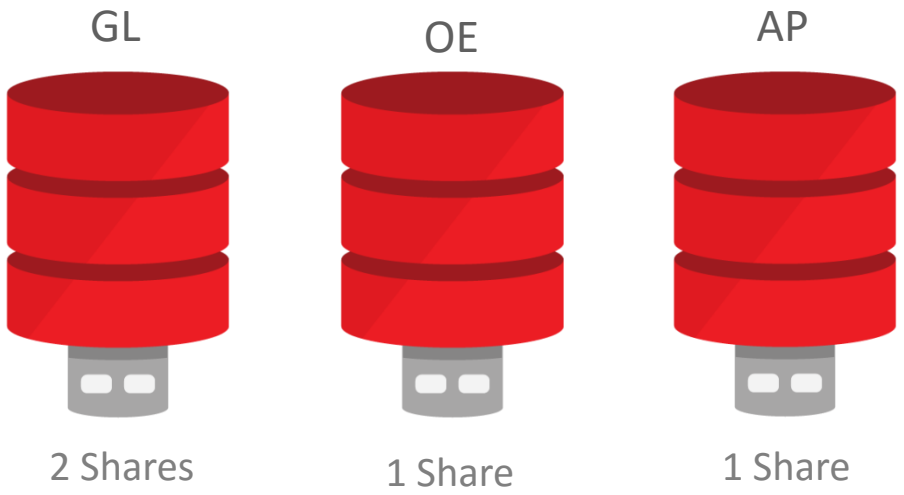


PDBごとのCPU_COUNTパラメータ

- PDB毎にCPUの使用を制限
  - 12.1ではCDBリソース・プランにシェアで設定
- 12.2ではPDBレベルのパラメータとしてCPU_COUNTを設定可能
  - PDBが構成の違うサーバーに移動しても、シェアを再計算する必要がない
  - シェアも互換性のために引き続きサポート
  - より低い値が有効

リソース・マネージャによるCPUリソースの制御

CDB Resource Planでは、PDB間でCPUリソースの分配にシェア(*shares*)を使用

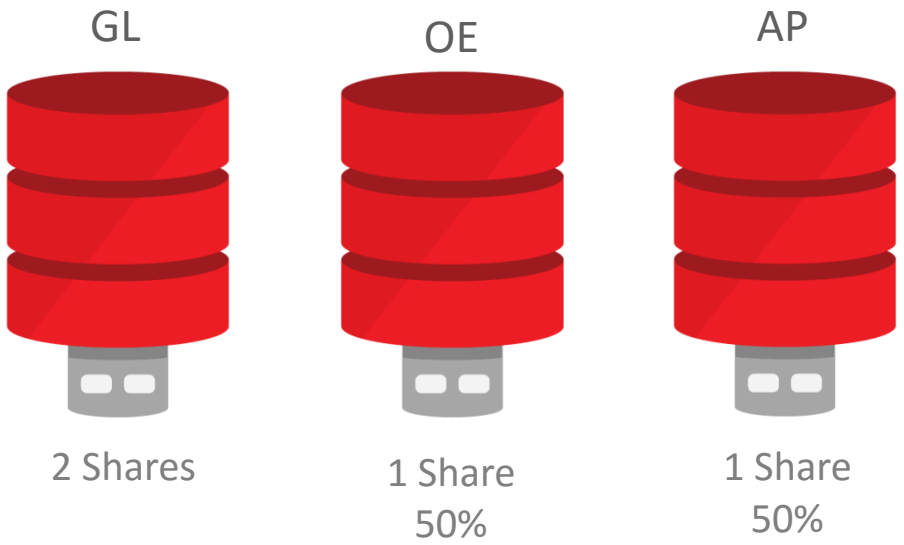


割り当てられている以上にCPUリソースを要求する場合は、待機イベント*"resmgr:cpu quantum"* でセッションが待機

Pluggable Database	Shares	Guaranteed CPU	Maximum CPU
GL	2	2/4 = 50%	100%
OE	1	1/4 = 25%	100%
AP	1	1/4 = 25%	100%

リソース・マネージャによるCPUリソースの制御

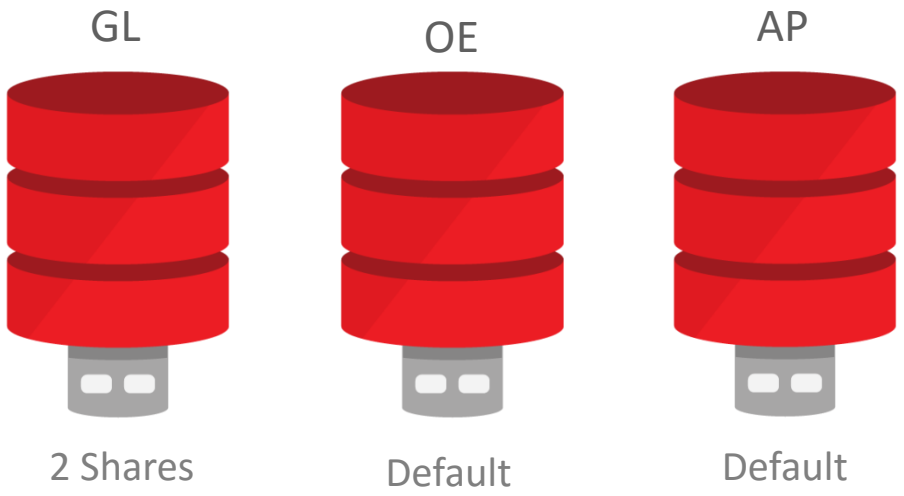
CDB Resource PlanはPDBのCPUリソースの使用量の制限のため、*utilization limit* を指定



Pluggable Database	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
GL	2	2/4 = 50%		100%
OE	1	1/4 = 25%	50%	50%
AP	1	1/4 = 25%	50%	50%

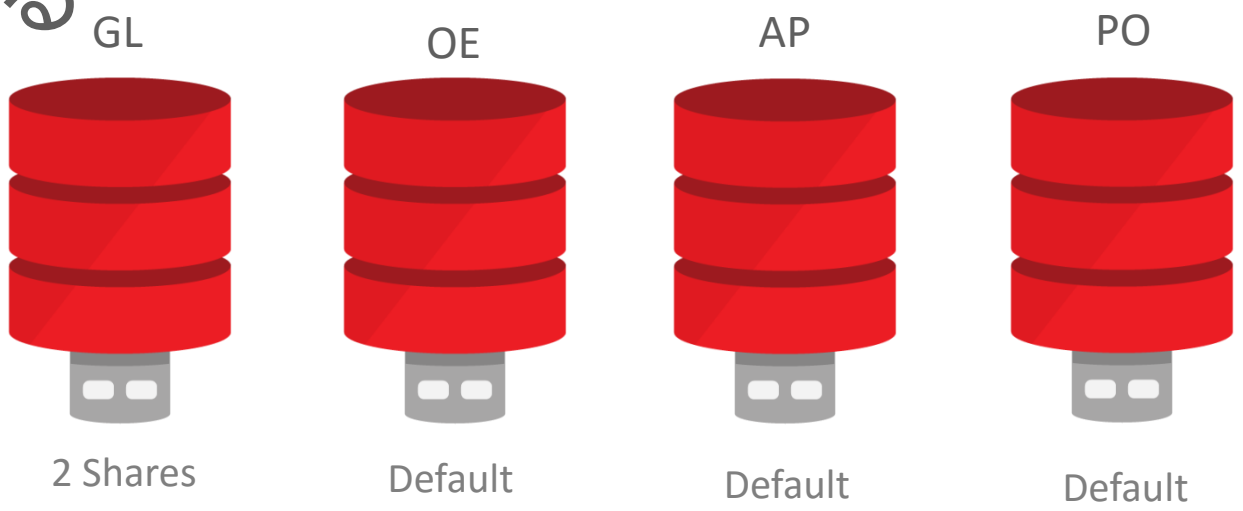
リソース・マネージャによるCPUリソースの制御

*default directive*の構成:
PDBに対するsharesとutilization limitのデフォルト値を定義



Pluggable Databa	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
(Default Directive)	1		50%	
GL	2	2/4 = 50%		100%
OE	Default (1)	1/4 = 25%	Default (50%)	50%
AP	Default (1)	1/4 = 25%	Default (50%)	50%

リソース・マネージャによるCPUリソースの制御



default directiveにより、PDBのプラグ/アンプラグ時にリソース・プランの修正が不要

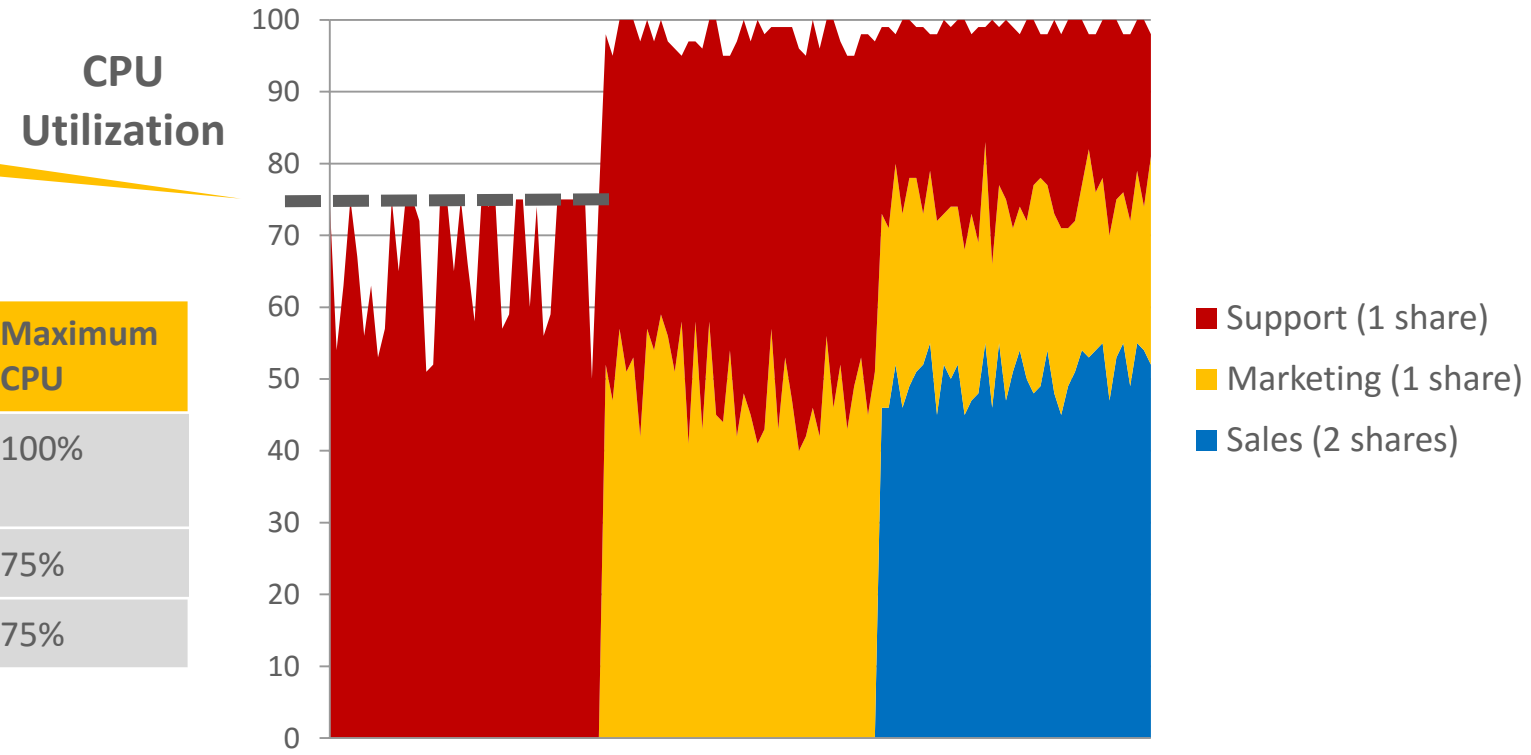
Pluggab	se	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
(Default D	)	1		50%	
GL		2	2/5 = 40%		100%
OE		Default (1)	1/5 = 20%	Default (50%)	50%
AP		Default (1)	1/5 = 20%	Default (50%)	50%
PO		Default (1)	1/5 = 20%	Default (50%)	50%

リソース・マネージャによるCPUリソースの制御

PDBごとのCPUリソース利用の制御

PDB SUPPORTのutilization limitの設定
”75%”により、CPUリソースに空きがあっても、
リソース割り当てを制限

CDB Resource Plan				
Pluggable Database	**Shares**	**Utilization Limit**	**Guaranteed CPU**	**Maximum CPU**
Sales	2		2/(2+1+1) = 50%	100%
Marketing	1	75%	25%	75%
Support	1	75%	25%	75%



Utilization Limitにより、クライアントに対して
一貫性のあるパフォーマンスを提供

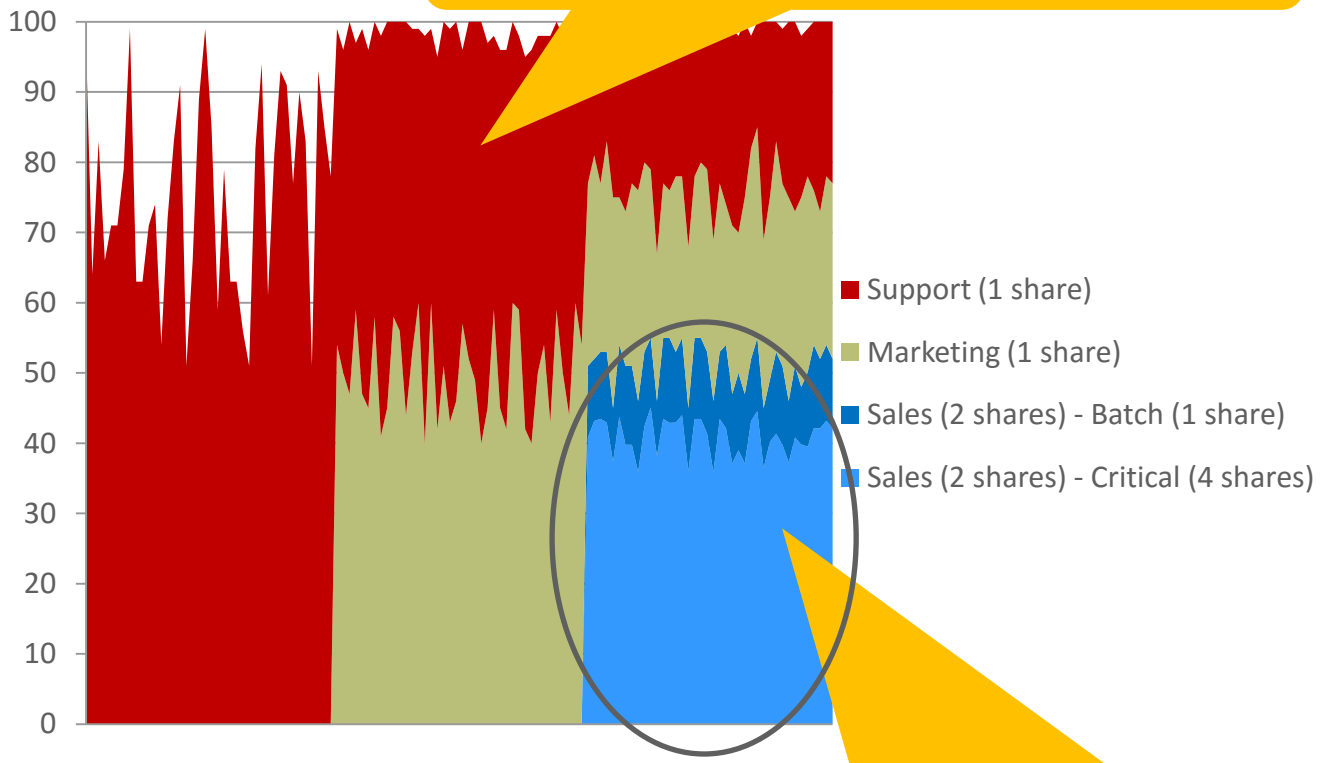
リソース・マネージャによるCPUリソースの制御

CDBとPDBのリソース・プランによるCPUリソース制御

CDB Resource Plan				
Pluggable Database	**Shares**	**Utilization Limit**	**Guaranteed CPU**	**Maximum CPU**
Sales	2		2/(2+1+1) = 50%	100%
Marketing	1	75%	25%	75%
Support	1	75%	25%	75%

PDB Resource Plan (Sales)				
Consumer Group	**Shares**	**Utilization Limit**	**Guaranteed CPU**	**Maximum CPU**
Critical	4		50%	100%
Batch	1		12.5%	100%
Maintenance	2	90%	25%	90%
Other	1		12.5%	100%

CDBリソース・プランはCPUリソースをPDB間でどのように分配するかを制御



PDBリソース・プランはCPUリソースをPDB内のコンシューマ・グループ間で分配するかを制御

PDB単位のCPUリソース管理

制限の強制

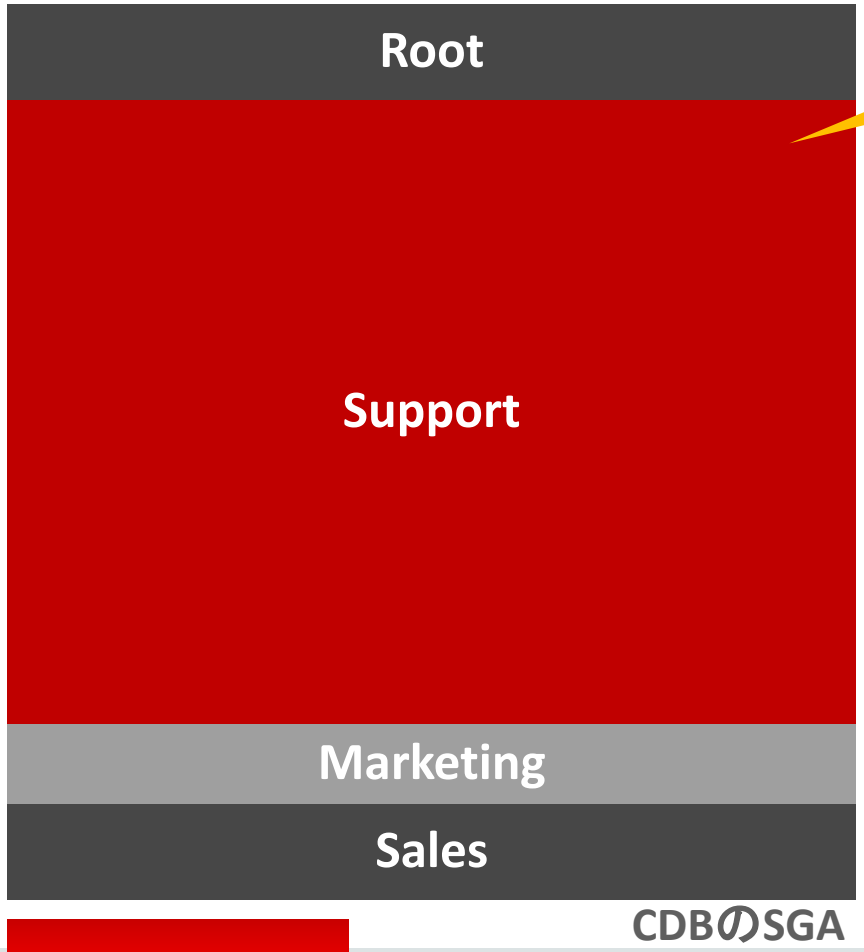
PDBごとにCPU_COUNTパラメータを設定

Pluggable Database	CPU_COUNT	Maximum CPU
Gold	54	75%
Silver	36	50%
Bronze-1	**18**	**18 / 72 = 25%**
Bronze-2	18	25%
Bronze-3	18	25%

このPDBは最大18 CPUスレッドを利用可能
サーバーが72CPU搭載されていれば、最大のCPU使用率は25%

“PDBケージング” は想定するCPUリソース使用の超過を防ぐことが可能。
クラウド環境での統合は重要

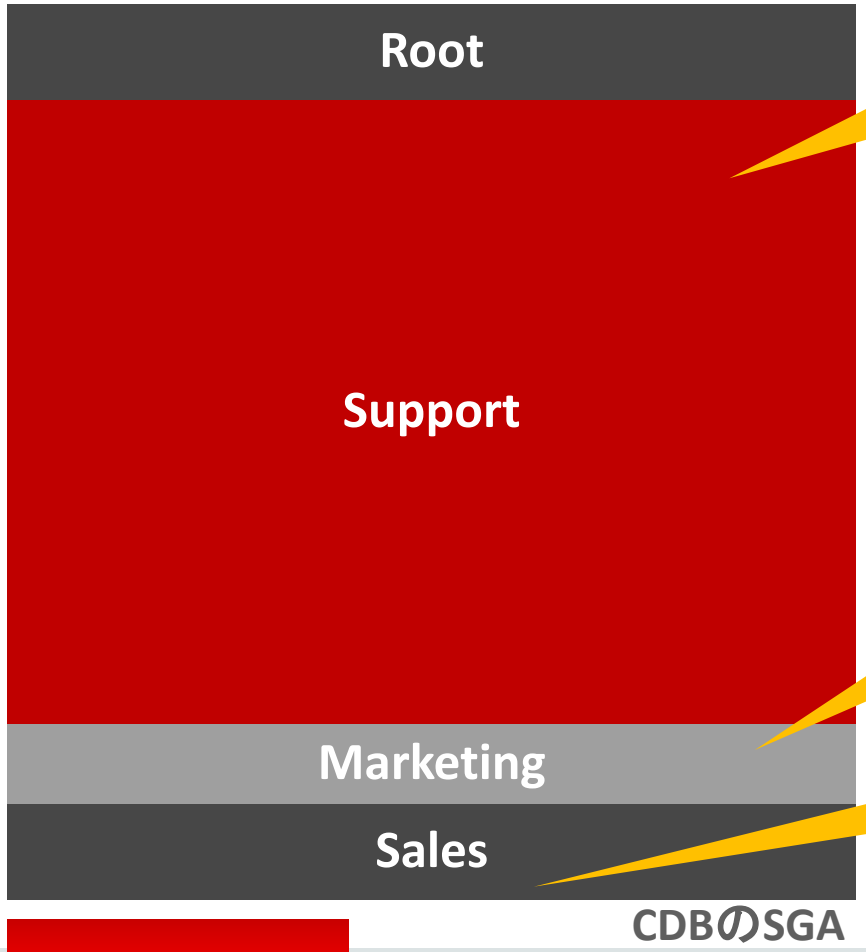
PDBのSGA管理



Support PDBはSGAのほとんどを使って良いか?

- SGAはメモリーを効率的に使用するために存在
- SGAの大半は、繰り返しアクセスされるオブジェクトのキャッシュ
  - バッファ・キャッシュ
  - 共有プール
  - インメモリー列ストア
- 高負荷なPDBは、SGAのキャッシュを占有しがち

PDBのSGA管理

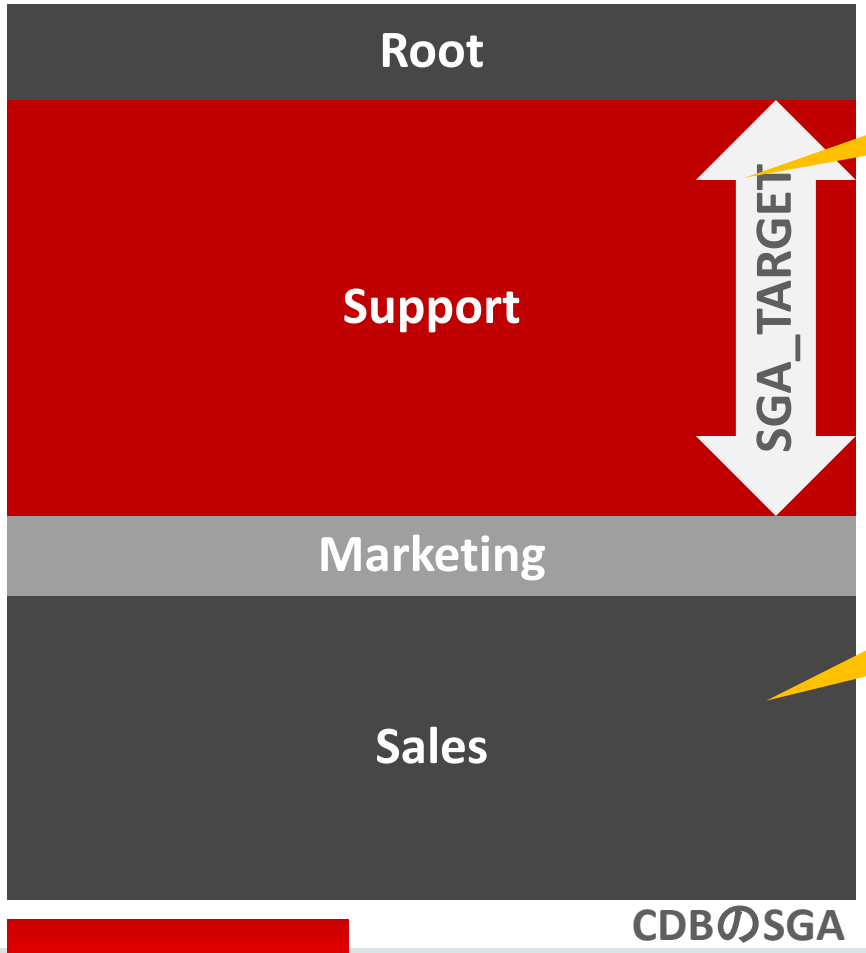


Support PDBはメモリーを良く使うワークロードを実行しており、SGAの大半を占有

Marketing PDB はほんの少し、SGAを使う

Sales PDBの性能はバッファ・キャッシュとパース済みカーソルに依存.
Support PDBはより高負荷で、このデータを追い出す

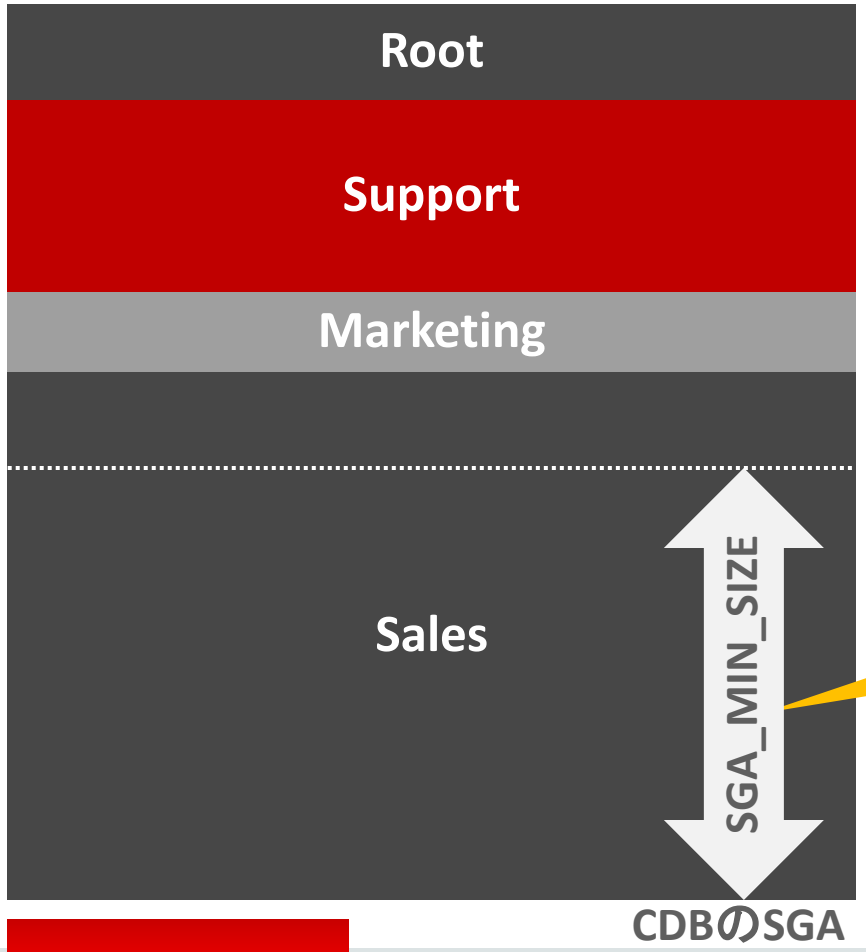
PDBのSGA管理



SGA_TARGETをPDBに設定
PDBのSGA使用の**hard limit**
を設定

特定のPDBのSGAを制限すること
で、他のPDBによりSGAを提供!

PDBのSGA管理



SGA_MIN_SIZEをPDBに設定.
PDBに最低限確保されるSGAを**保証**

リソース・マネージャによるPDBごとのメモリー管理

従来はCDBレベルのパラメータが12.2ではPDBレベルで設定可能

Parameter	Description
SGA_TARGET	PDBへのSGAの最大サイズ
SGA_MIN_SIZE **New in 12.2!**	PDBに保証されるSGAのサイズ (バッファキャッシュと共有プール) SGA_MIN_SIZEにはSGA_TARGET の設定値の**50%以下**の値を設定
DB_CACHE_SIZE	PDBに保証されたバッファキャッシュのサイズ
SHARED_POOL_SIZE	PDBに保証された共有プールのサイズ
PGA_AGGREGATE_LIMIT	PDBの最大PGA サイズ
PGA_AGGREGATE_TARGET	PDBのターゲットPGAサイズ

- 各パラメータはCDBで設定した値以下に設定
- パラメータごとに設定できる上限が存在

リソース・マネージャによるPDB I/Oレート制限

- 1つのPDBによってストレージ・システムが占有されることを防ぐ
  - バッファ・キャッシュの過剰な読み書き
  - 過剰なスキャンI/O
  - インポート/エクスポートによる過剰な読み書き
- 2つの新しいPDBパラメータ
  - MAX_IOPS: 一秒当たりの最大I/Oリクエスト数
  - MAX_MBPS: 一秒当たりの最大I/O 転送量 (単位: Mega bytes)
  - これらのパラメータは動的に変更可能
- Exadata以外のシステム上のPDBに設定可能
  - Exadata環境では設定できない
    - より高機能なExadata I/O Resource Management (IORM)が利用可能なため
- PDBからのI/O要求がMAX_IOPSまたはMAX_MBPSを超える場合に制限される
  - 待機イベント **“resmgr:io rate limit”** が発生

PDBレベルのリソース使用状況の確認

- V$RSRCPDBMETRICビュー:PDBレベルのリソース使用状況が確認可能

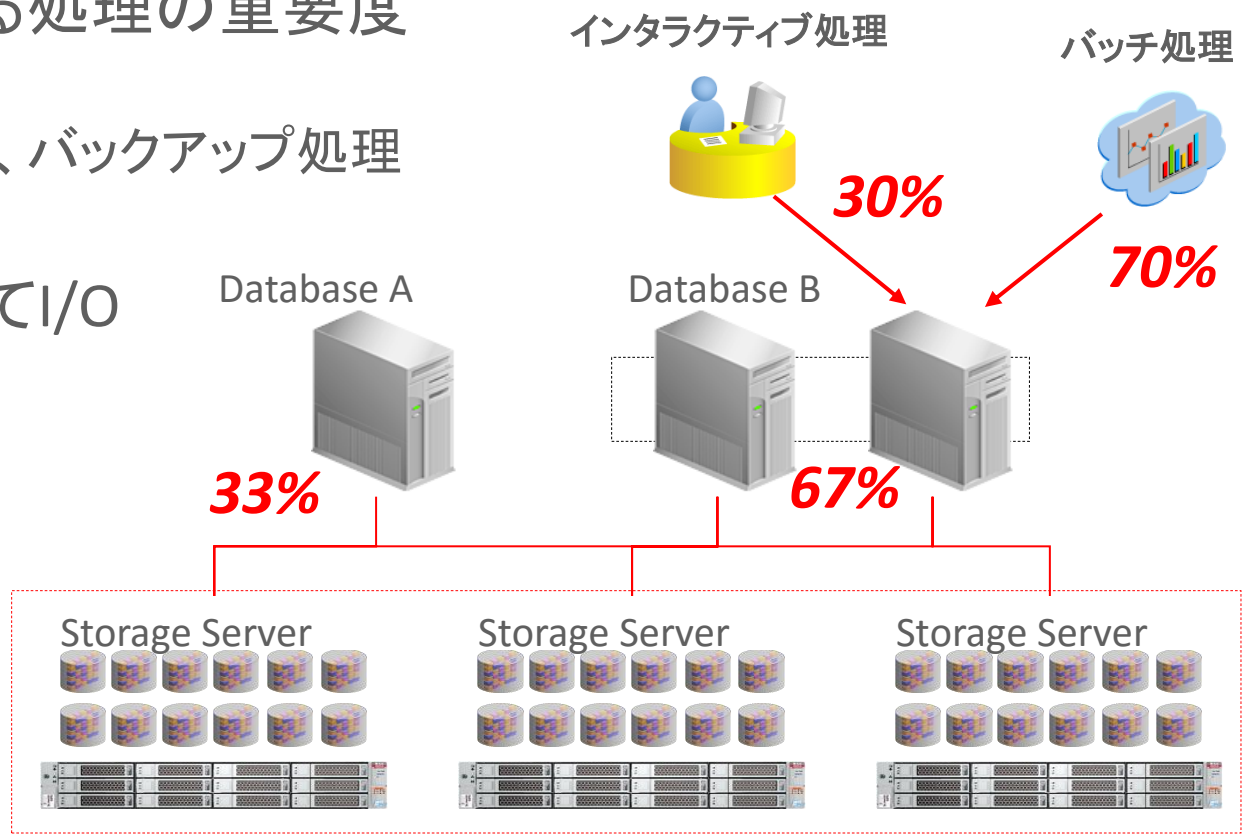
Parameter	Description
NUM_CPUS	PDBで設定されているCPU_COUNTの値、設定されていなければシステムで利用可能な値
CPU_UTILIZATION_LIMIT	利用できる最大のCPU使用率
IOPS	過去1分間の1秒間あたりのIOPS
IOMBPS	過去1分間の1秒間あたりのI/O量(MB単位)
IOPS_THROTTLE_EXEMPT	I/O制御の対象とならなかった過去1分間の1秒間あたりのIOPS
IOMBPS_THROTTLE_EXEMPT	I/O制御の対象とならなかった過去1分間の1秒間あたりのI/O量(MB単位)
SGA_BYTES	現在割り当てられているSGAサイズ
BUFFER_CACHE_BYTES	現在割り当てられているバッファ・キャッシュ・サイズ
SHARED_POOL_BYTES	現在割り当てられている共有プール・サイズ
PGA_BYTES	現在割り当てられているPGAサイズ

- V$RSRCPDBMETRIC_HISTORYビュー:V$RSRCPDBMETRICの直近1時間のヒストリを表示

Exadata I/O Resource Management (IORM)

ストレージI/O帯域幅制限機能により、統合DB環境でも安定したI/O性能を保証

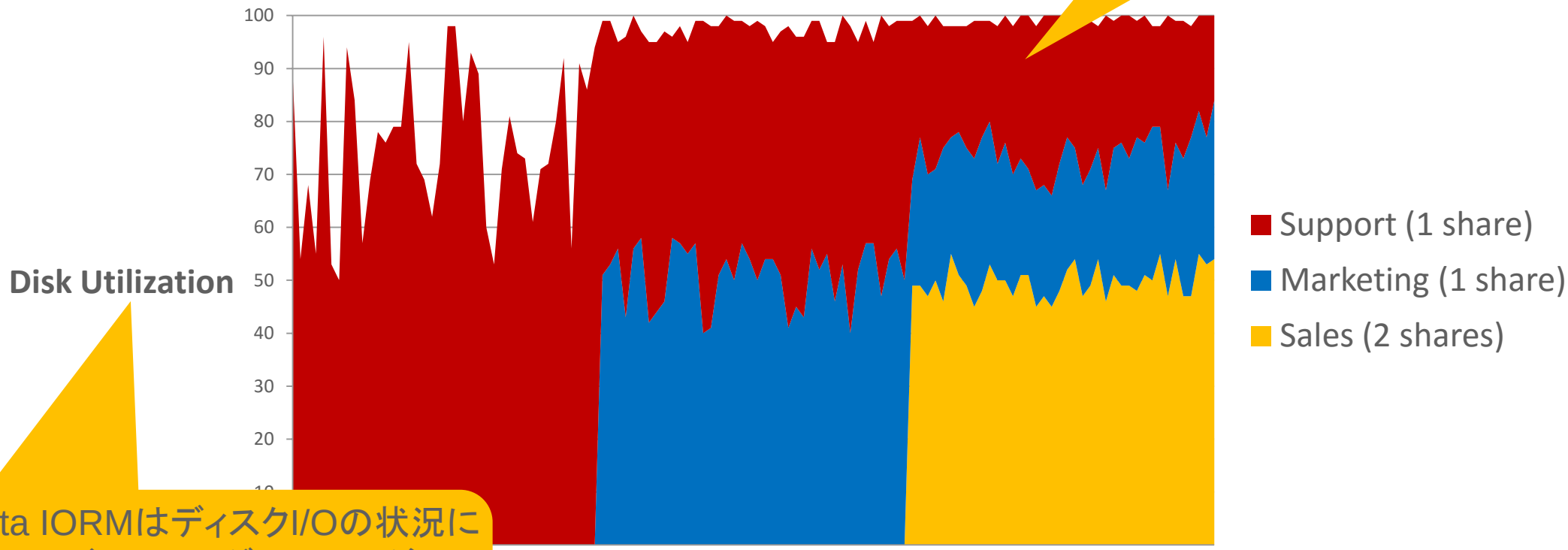
- 統合環境では各システムで実行される処理の重要度を考慮する必要がある
  - 重要なOLTP処理の裏でAd-hocクエリやETL、バックアップ処理が実行されるようなケース
- IORMにより、処理の優先度に基づいてI/O帯域を調整
  - 優先度の高いI/Oが優先されるため **重要な業務が性能劣化する可能性を排除**
- データベース統合基盤として CPU/メモリーはマルチテナントで効率化しつつ、**IORMで性能リスクを低減**



Exadata IORM

PDBごとのディスクI/Oのスケジューリング

Exadata IORMはCDB Resource Planによって、PDBごとのディスクI/Oを制御



Exadata IORMはディスクI/Oの状況に応じてスケジューリングし、DBAがストレージのIOPSやMBPSをワークロードを知っておく必要がない

参考: Exadata Express Cloudの設定

PDBサービスのリソース制御を実装している層

制御項目 / パラメータ	目的
DB_PERFORMANCE_PROFILE	PDBサービスの層を指定
CPU_COUNT	CPU使用率の制限
SHARE	CPUリソース、ディスクI/O、フラッシュI/Oの分配を配置
SGA_TARGET	SGAの使用量を制限
PGA_AGGREGATE_TARGET PGA_AGGREGATE_LIMIT	PGAの使用量を制限
SESSIONS	セッション数を制限
MAX_IDLE_TIME	長時間アイドルのセッションの切断
IORM	ディスクとフラッシュのI/Oの公平性の実装

セキュリティ

PDBレベルのアクセス制御の強化、ロックダウン機能

共通アクセスの潜在的脆弱性への対応

クラウドなどCDBを共有する環境でPDBごとの詳細なアクセス制御を実現

ネットワークアクセス	共有ユーザーとオブジェクトアクセス	DB管理操作	OSコマンド実行	ファイルアクセス
PDB内からAdvanced Queuing (AQ)やUTL_SMTPなどを利用したネットワーク・アクセスを制限	共通ユーザーを介した別PDBのオブジェクトや共通オブジェクトへのアクセスを制限	データベースのオプションの利用や、ALTER SYSTEMなどのDB管理操作の実行をPDBごとに詳細に制限	DBサーバー上でOS操作を行う際のOSユーザーをPDBごとに個別に指定	PDBがアクセスできるDBサーバー上のディレクトリを特定の場所以下に限定
PDBロックダウン・プロファイル			PDB OSクレデンシャル	パス・プレフィックス / CREATE_FILE_DEST パラメータ

ロックダウン・プロファイル: 権限に対する制限

- ロックダウン・プロファイル
  - grantによる権限管理の仕組みを補完
  - grantのみでは“all or nothing”
  - grantで許可された操作に、より粒度の細かい制御を追加

```
SQL> grant alter system
     to pdb_user;
```

```
SQL> alter lockdown
     profile p1 disable
     statement=
       ('ALTER SYSTEM')
     clause=('SET')
     option= ALL EXCEPT
     ('cursor_sharing',
      'optimizer_mode');
```

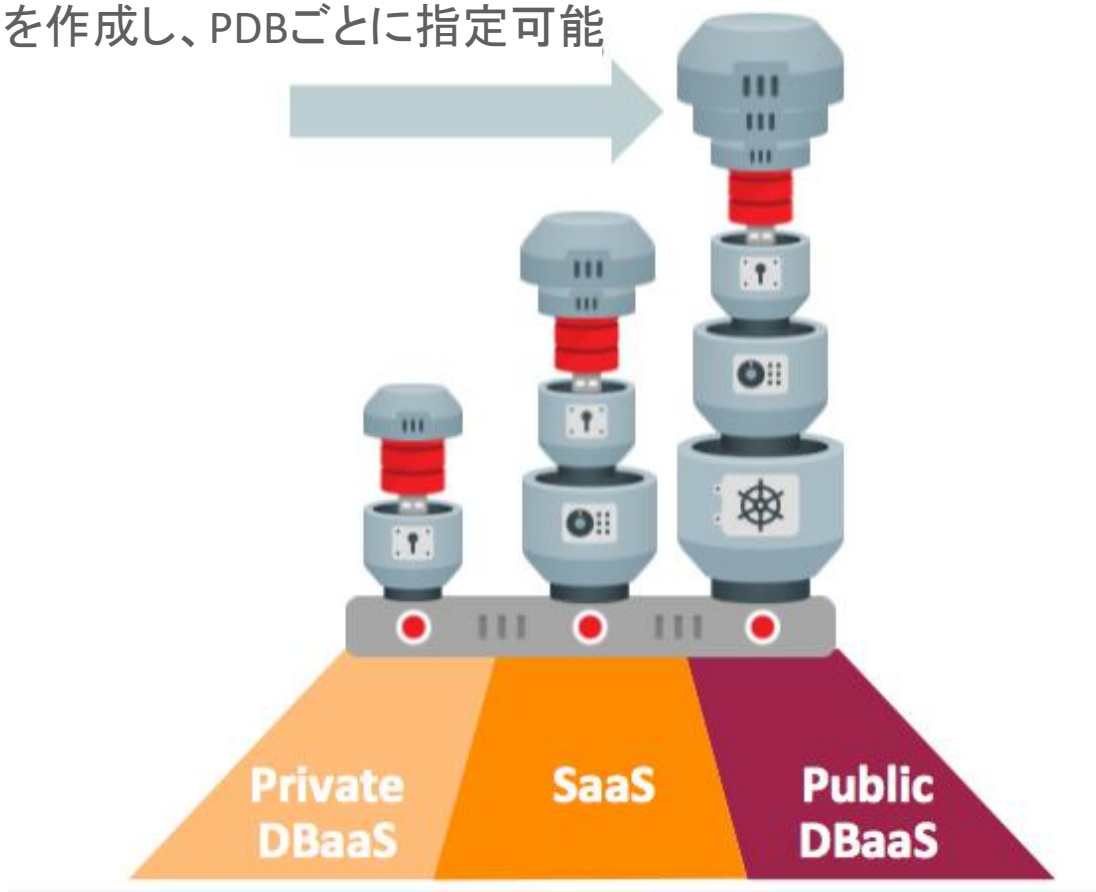
- ‘alter system’のスコープ
  - ~~approx_for_percentile~~
  - ~~common_user_prefix~~
  - cursor_sharing
  - optimizer_mode
  - ~~trace_enabled~~
  - …

ロックダウン・プロファイルによる設定可能な分離性

- 宣言的にPDBに対するアクセスをブロックする手段:
  - ネットワーク
  - 管理的な機能
  - 共通ユーザーと共通オブジェクト
- 制限の適用範囲を指定可能

ロックダウン・プロファイルを指定するコンテナ	適用範囲
CDB$ROOT	すべてのPDB
アプリケーション・ルート	すべてのアプリケーションPDB
それぞれのPDB	そのPDBのみ

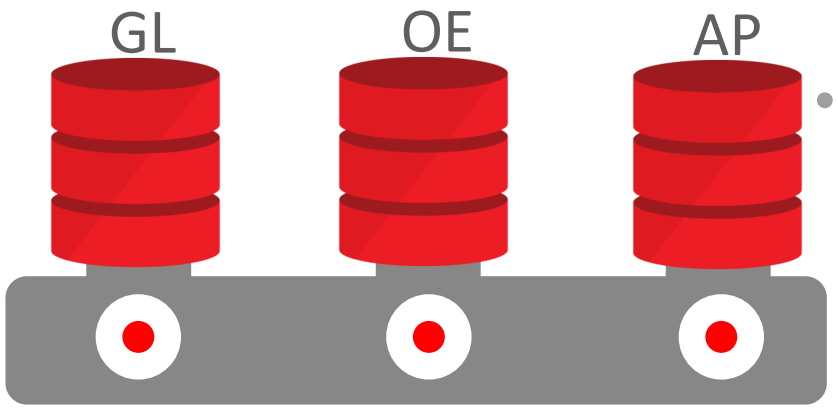
制限の強度ごとにプロファイルを作成し、PDBごとに指定可能



PDBロックダウン・プロファイル

PDBごとに利用できる機能や管理操作を詳細に制限

	GL	OE	AP
AWRへのアクセス	○	×	×
共有スキーマへのアクセス	×	○	×
UTL_SMTPの利用	×	×	×
UTL_FILEの利用	×	×	○
PARTITIONINGの利用	○	×	○
ALTER SYSTEM SET CPU_COUNTの実行	○	×	×
ALTER SYSTEM SUSPENDの実行	×	×	×

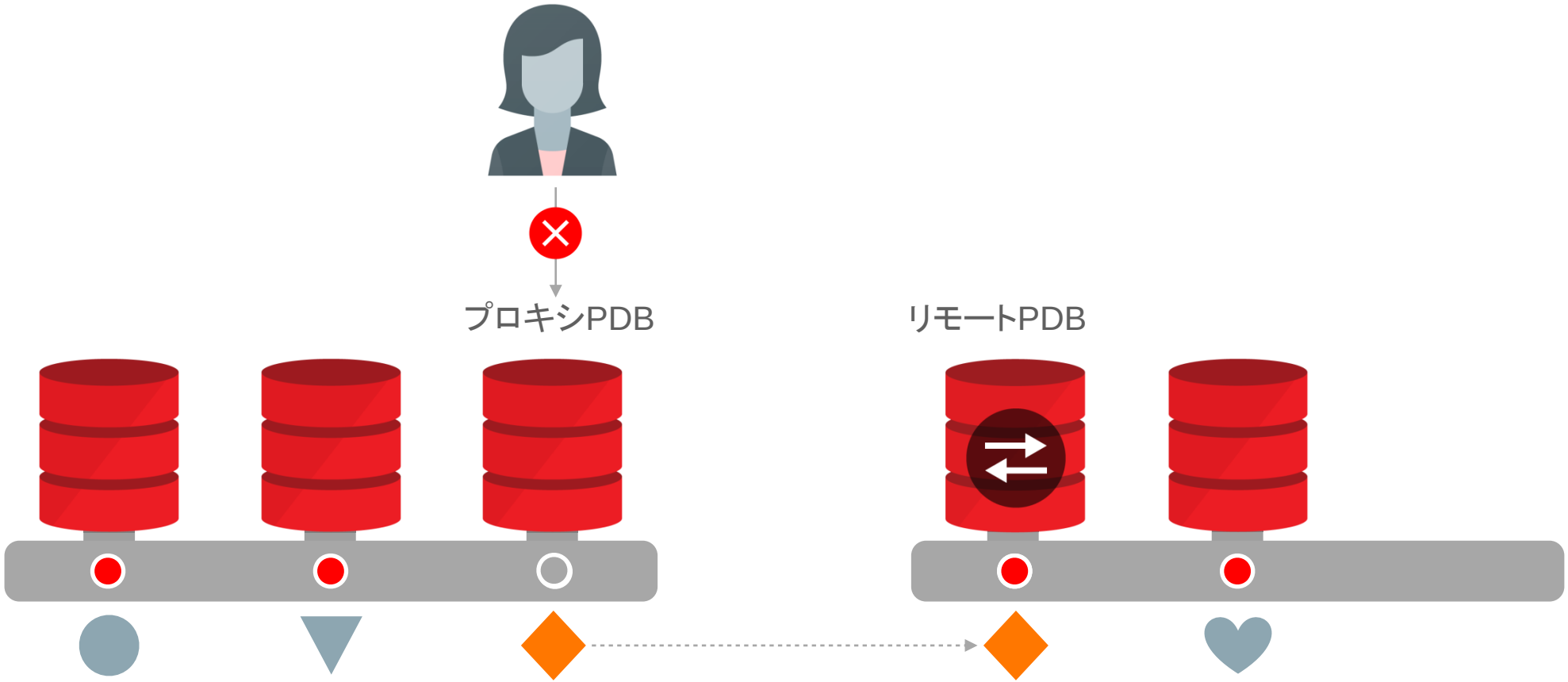


- **機能制限**
  - AWRへのアクセス
  - 共有スキーマへのアクセス
  - Oracle_Text
  - JAVA
  - ネットワークアクセス(UTL_SMTP等)
  - OSアクセス(UTL_FILE等)
- **オプション利用制限**
  - ADVANCED QUEUING
  - PARTITIONING
- **SQL文実行制限**
  - ALTER DATABASE
  - ALTER PLUGGABLE DATABASE
  - ALTER SESSION
  - ALTER SYSTEM

作成されたロックダウン・プロファイルはdba_profilesビューから確認可能

プロキシPDBによる位置透過性の実現

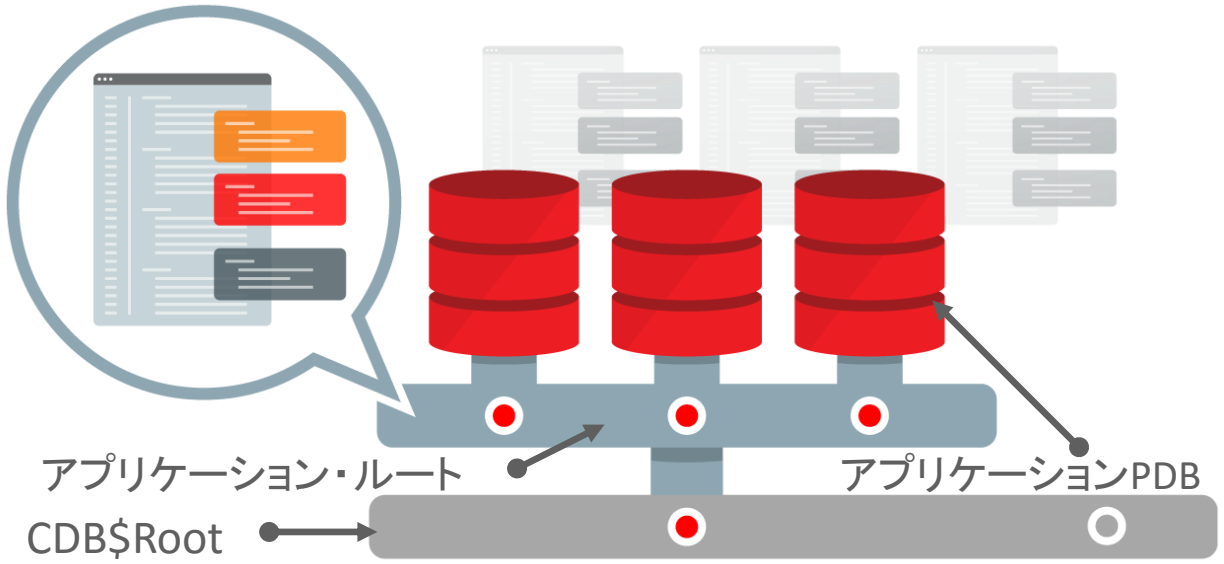
プロキシPDBによりリモートPDBをローカルPDBと同様に利用



アプリケーション・コンテナ

- PDB間でオブジェクトを共有
  - コード、メタデータおよびデータ
  - アプリケーションのインストールは一度だけ
- さらに容易な管理
  - アプリケーションPDBの即時プロビジョニング
  - アプリケーション・コンテナにアップデート適用

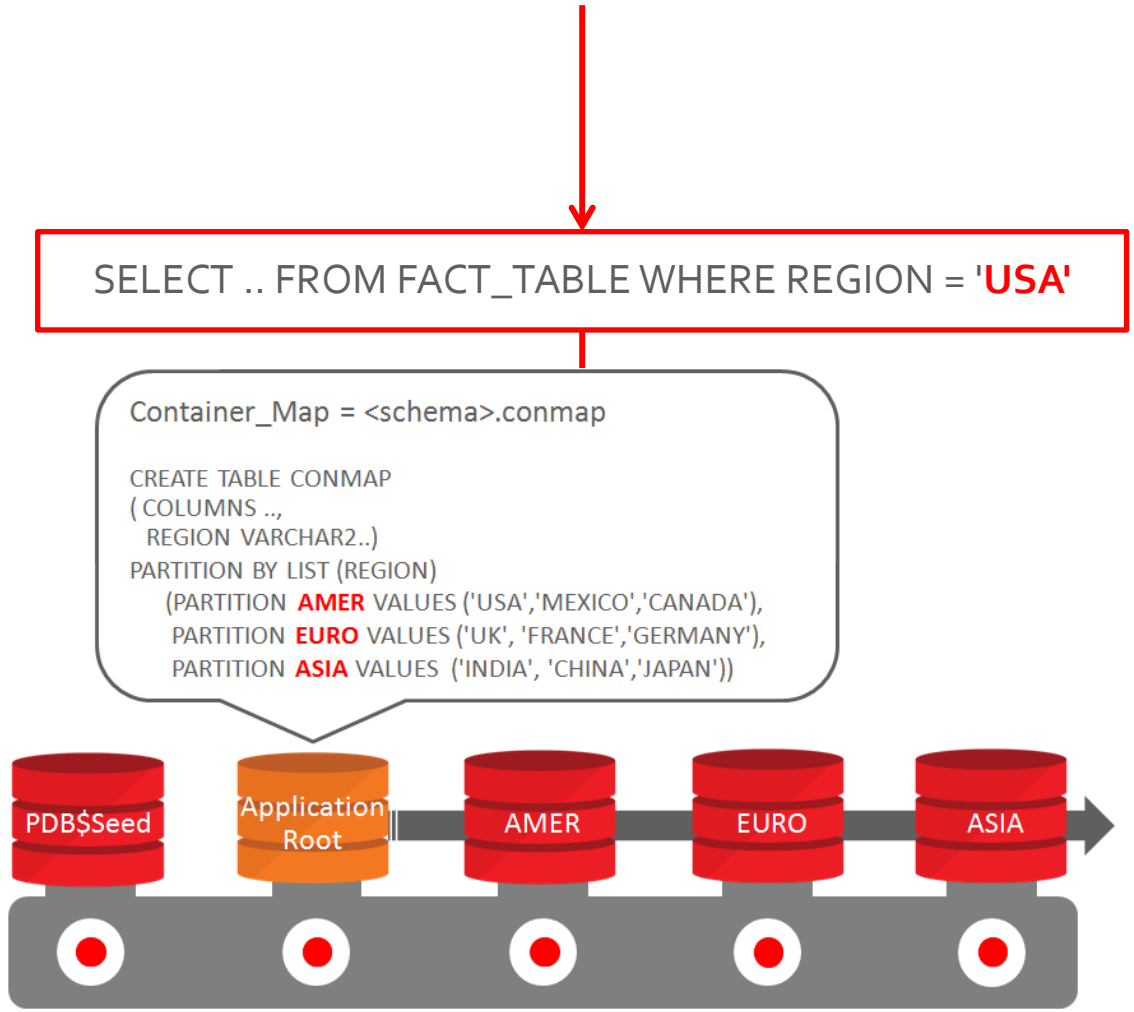
パッケージ・アプリケーション / SaaS	• 行(Row)ベースのテナント管理 • スキーマ・ベースのテナント管理
データ管理	• ロジカル・データ・ウェアハウス • マスター・データ管理
アプリケーション Dev & Test	• アプリケーション開発用のデータベースの迅速な配布
開発の自動化	• 開発やテストで必要となる共通データ、ユーティリティの伝播や配布



アプリケーション・コンテナでの"アプリケーション"
• アプリケーションのバック・エンドのデータベース・オブジェクト

コンテナ・マップ

- 列の値を基にPDBを論理的にパーティション化
  - アプリケーション・コンテナで利用
  - パーティション定義用のテーブル(マップ・オブジェクト)を使用
- 多くのクエリーで頻繁に利用される列をパーティション・キーとして指定
  - 例：地域名、部署名、日付データなど
- 使用可能なパーティション手法
  - レンジ
  - リスト
  - ハッシュ

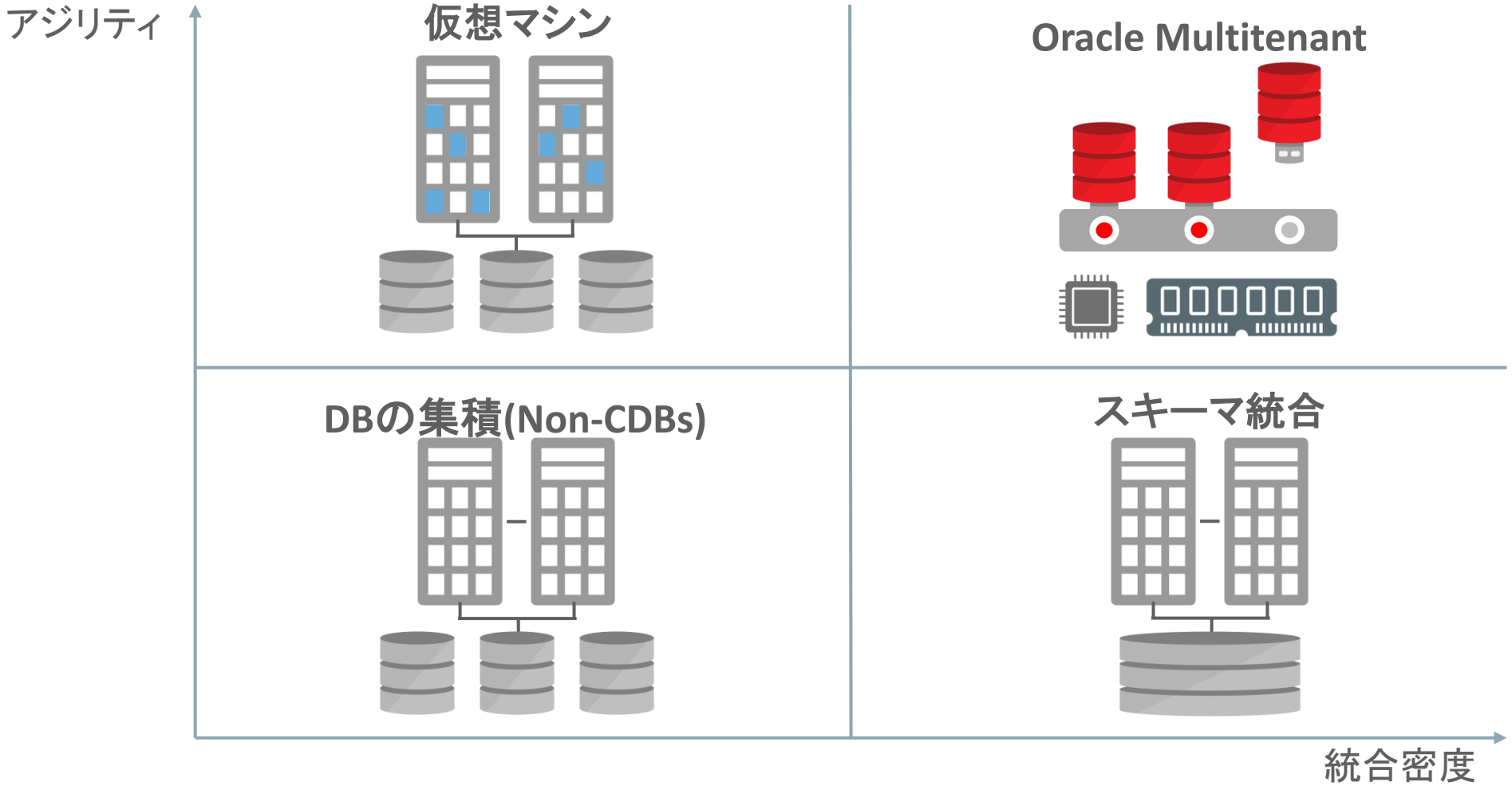


第2部

3. データベース統合手法による比較

Oracle Multitenantの強み

データベース統合手法のマジック・クアドラント



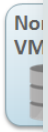
データベース統合手法による比較検証

詳細はAppendixをご参照ください

データベース統合

- 検証1:「仮想マシン」

Non-CDB VM
VS.
CDB PDB



– 環境
  - Commodity IA Server
  - VM環境

– ワークロード
  - Light weightな処理、低し

ORACLE

VM(仮想マシン)による統合と比べた Oracle Multitenan

- より多くのデータベース
  – オーバーヘッドの少ない統
    - Hyper Visorを介さず、OSの
      – Hyper VisorをOSによるリ
    - CPU、メモリーなどのリソー
      – 余剰リソースを減らすこ
- データベースのアーキテ
  – 共通する操作はCDB$Root
  – Hyper Visorレイヤーを含ま
  – 可用性: 少ないリソースで、
    - リソース再配置/フェイルオ

VM環境でMulti
データベース・レイ

ORACLE

DBの集積による統合と比べた Oracle Multitenantによる統合の優位性

- リソースの効率的な利用 **高い集約率**
  – CDB構成は**バックグラウンド・プロセスの数**を抑えられる
    - サーバー・プロセスがSQL処理に使えるCPUリソースが増える
    - コンテキスト・スイッチの発生回数が低減
  – 共有プールおよびバックグラウンド・プロセスが使う**PGAの合計サイズ**が抑えられる
    - 共有プールを複数のPDBで利用
      – SQLカーソルの有効利用
    - メモリーのフットプリントが軽くなり、余剰をバッファ・キャッシュに割り当てられる
      – バッファ・キャッシュ・ヒット率の向上
  – バッファ・キャッシュをPDB間で共有するため、**特定のPDBの急なワークロード増加に強い**

- パフォーマンス特性 **高いパフォーマンス**
  – 同じ数(252個)のDB(non-CDB構成)/PDB(CDB構成)の比較では、CDB構成が**より少ないリソース消費でより高いパフォーマンス**を実現
  – REDOログやダーティ・バッファの書き込みが**複数のPDB分**をまとめて効率よく行われる
    - I/Oサイズが大きくなり、ストレージIOPSが下がる
    - scalable log writer architecture
      – 複数のLog Writer Slave(LG*nn*)による並列書き込み

ORACLE

184

ORACLE

高いアジリティ

Manage many as oneによる管理作業の共通化: 一方で、粒度の細かい制御も可能

要件	スキーマ統合	Oracle Multitenant
パッチ適用 / アップグレード	• DB全体の実施か無実施の2択**(All or Nothing)** • メンテナンス・ウィンドウの**全テナントと調整が必須**	• DB全体に対して**一括実施**も可能 • **各テナントPDBごと**に、アンプラグ/プラグによるパッチ適用/アップグレード
テナントの可搬性 • 負荷分散 • クラウドへ/クラウドから	**煩雑で時間のかかる手順、業務停止も伴う** • RMAN リストア • Data Pump Export/Import	**速くて、シンプル** • アンプラグ/プラグ • **PDB再配置 (オンライン)**
ポイント・イン・タイム・リカバリ	• RMAN または Data Pump	
新しいテナントのプロビジョニング	• スキーマ作成、インストール・スクリプトの実行 • RMAN リストア または DataPump インポート	• PDBクローニング(ローカル、リモート)
クローニング	• フル・データセットかつフル物理コピーによるクローニング • データ・セットの一部のみは不可 • シン・プロビジョニングは不可 • RMAN または Data Pump	• 速くて、シンプルなPDBクローニング • データ・セット: フル、一部、またはメタデータのみ • ホット・クローン、リフレッシュ • 複製方法: フル・コピー、またはスナップショット・クローン

適用の容易さ

スキーマ統合

- アプリケーションの“バック・エンド”にあたるデータベースのオブジェクトは、慣例にならい命名、管理が行われる
  - スキーマ名、オブジェクト名(表名、表領域名など)
- テナント/スキーマとして統合する場合、DB内の他のスキーマとの名前空間の競合を避けるために**アプリケーションの変更**を余儀なくされる
  - 一般的なスキーマ名
  - パブリック・シノニム

Oracle Multitenant

- テナント/PDB間の独立性により、アプリケーションの“バック・エンド”定義には影響を一切与えない

• サービス	接続
• 表領域	ストレージ
• 名前空間	論理的なアプリケーションのデザインに基づき、PDB内で自由に設定可能

- **アプリケーションの変更は不要**のため、適用が容易

その他のスキーマ統合に対するOracle Multitenantの強み

セキュリティ

- 権限の範囲はPDB内で閉じる
  - アプリケーションのインストール時に強力な権限を要求されるケースがある
- サービスと接続はPDBレベルで分離される

リソース分離

- リソース・マネージャによるリソース制御
  - CPU
  - IO (Exadataのみ)
  - セッション数
  - パラレル・サーバー・プロセス数
- キャッシュ管理の分離
  - バッファ・キャッシュ/共有プール
    - alter system flush shared pool 文をPDB内で実行した場合、そのPDBに関する共有プール上のキャッシュのみフラッシュされる

メトリック収集

- PDBレベルの情報
  - EM
  - AWRレポート
  - ディクショナリ・ビュー
- ツールがPDBに直接接続可能

Oracle Multitenantによるコスト削減効果

全てのカテゴリでコストを削減できる唯一のアーキテクチャ

	容量の効果的利用	管理レイヤーの削減	負荷のオーバーヘッドの分散・共有	スナップショットの活用	一括管理	細かい単位の管理	セルフ・サービス	適用の容易性
Oracle Multitenant	✓	✓✓ (OS &RDBMS)	✓✓ (CPU & Mem)	✓	✓	✓	✓	✓
スタンドアロン・サーバー						✓		✓
データベース集積	✓	✓ (OS only)	✓ (CPU only)			✓		✓
スキーマ統合	✓	✓✓ (OS &RDBMS)	✓✓ (CPU & Mem)		✓			
仮想マシン	✓		✓ (CPU only)			✓	✓	✓

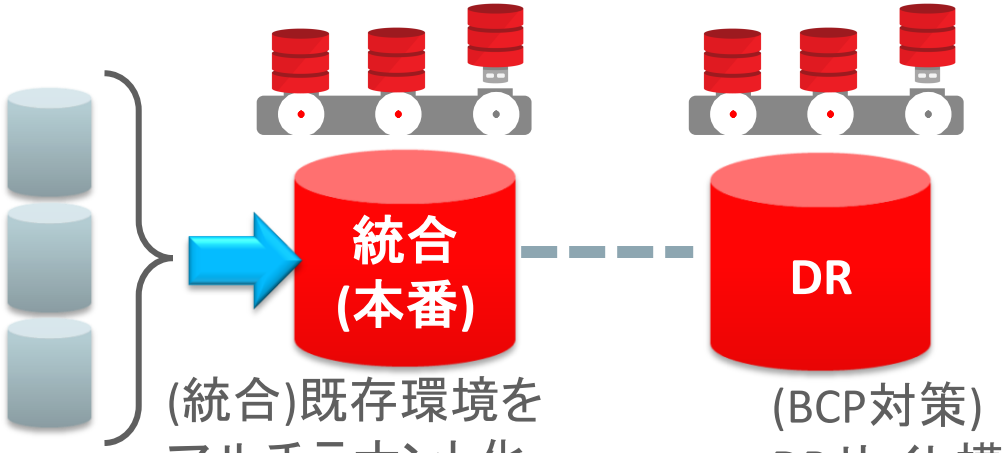
第2部

4. Oracle Multitenantによる統合で広がるユースケース

Oracle Multitenantで実現する新しいビジネス価値とお客様事例

マルチテナントの代表的なユースケース

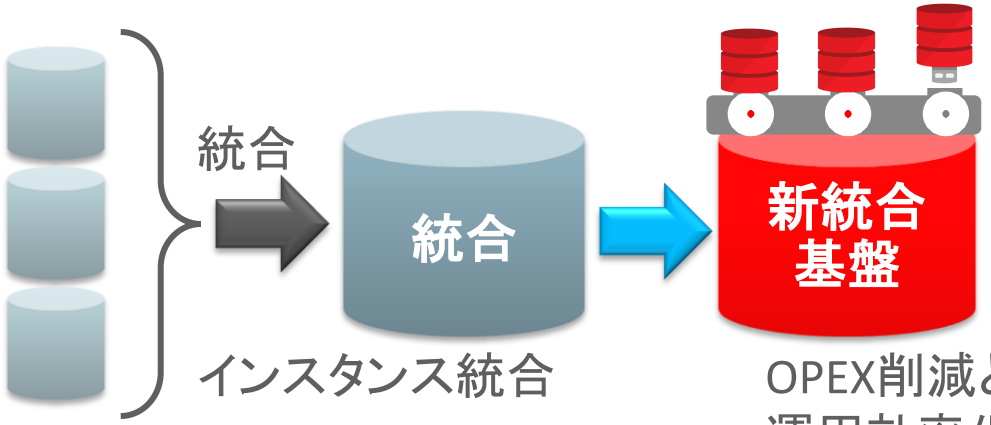
①既存システムの統合基盤(CAPEXとOPEXの削減)



(統合)既存環境を
マルチテナント化

(BCP対策)
DRサイト構築

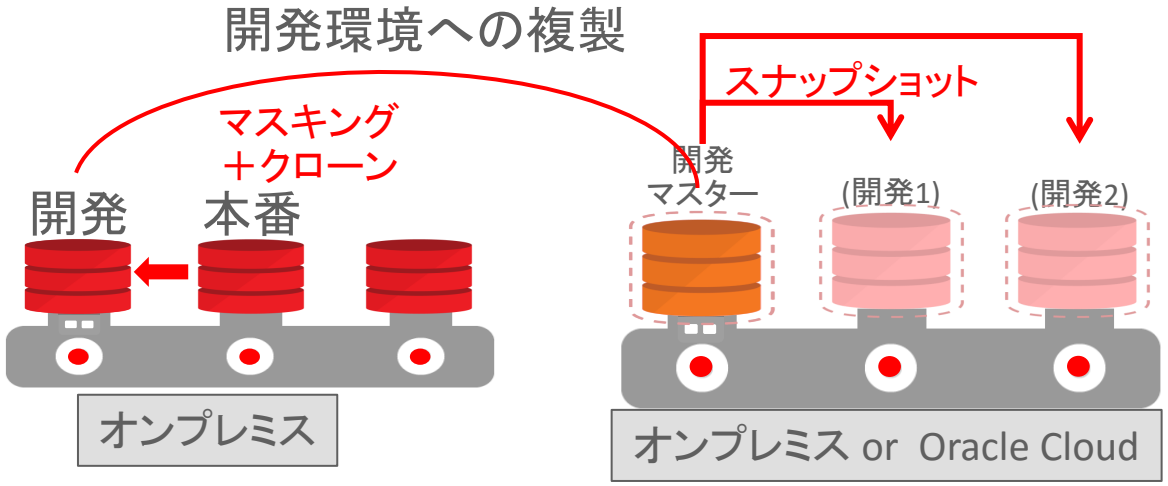
②統合システムの更改(12cにアップグレード)



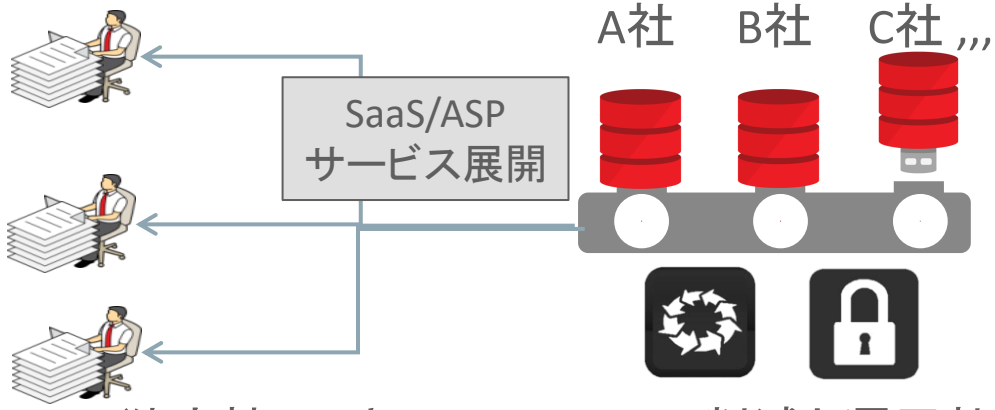
インスタンス統合

OPEX削減と
運用効率化

③開発・検証環境のアジリティ向上



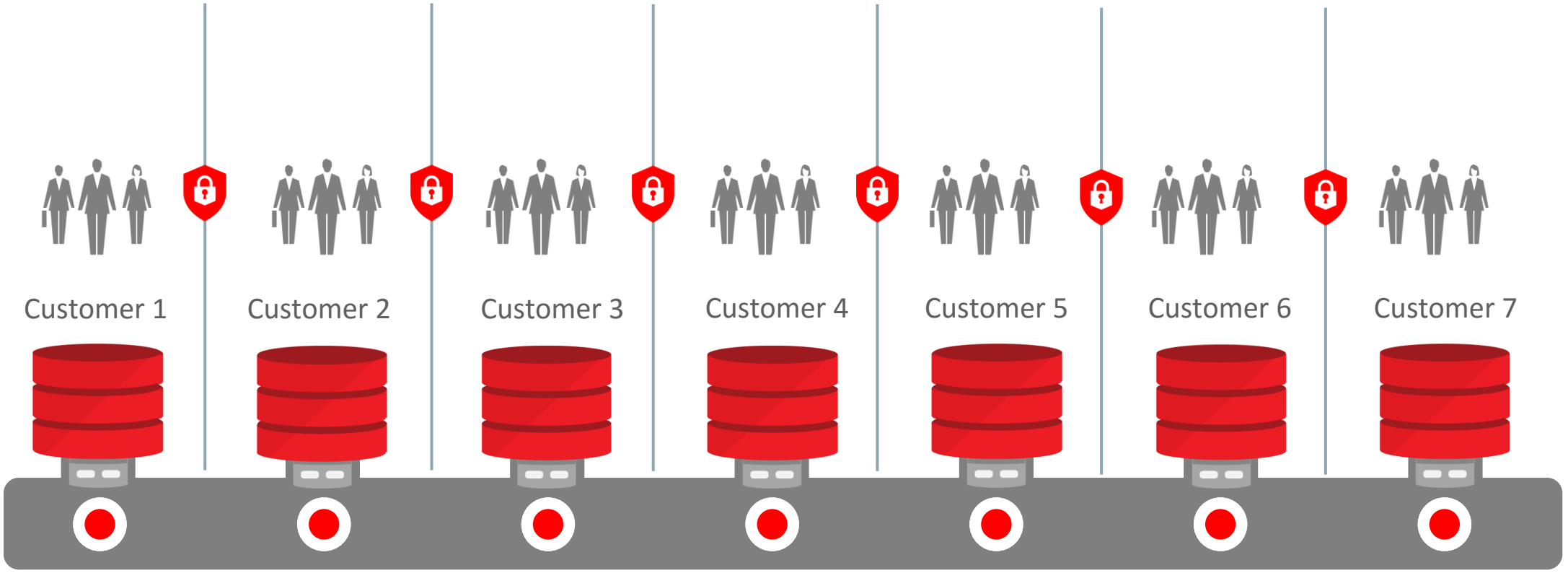
④ SaaS/ASPサービスの基盤としての活用



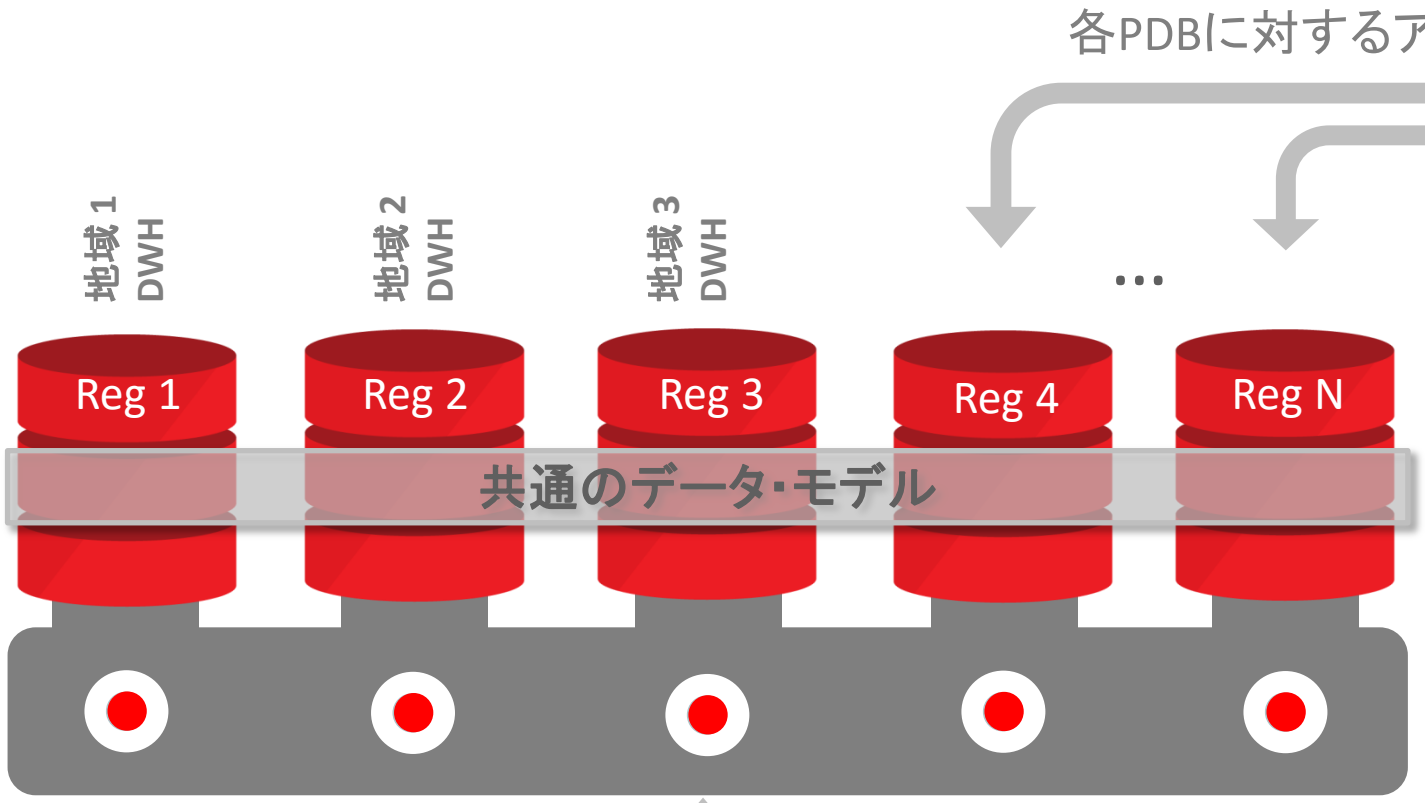
独立性、セキュリティ、OPEX削減と運用効率化

Oracle Multitenant for Software as a Service

マルチテナント対応をアプリケーションではなくデータベースで実装

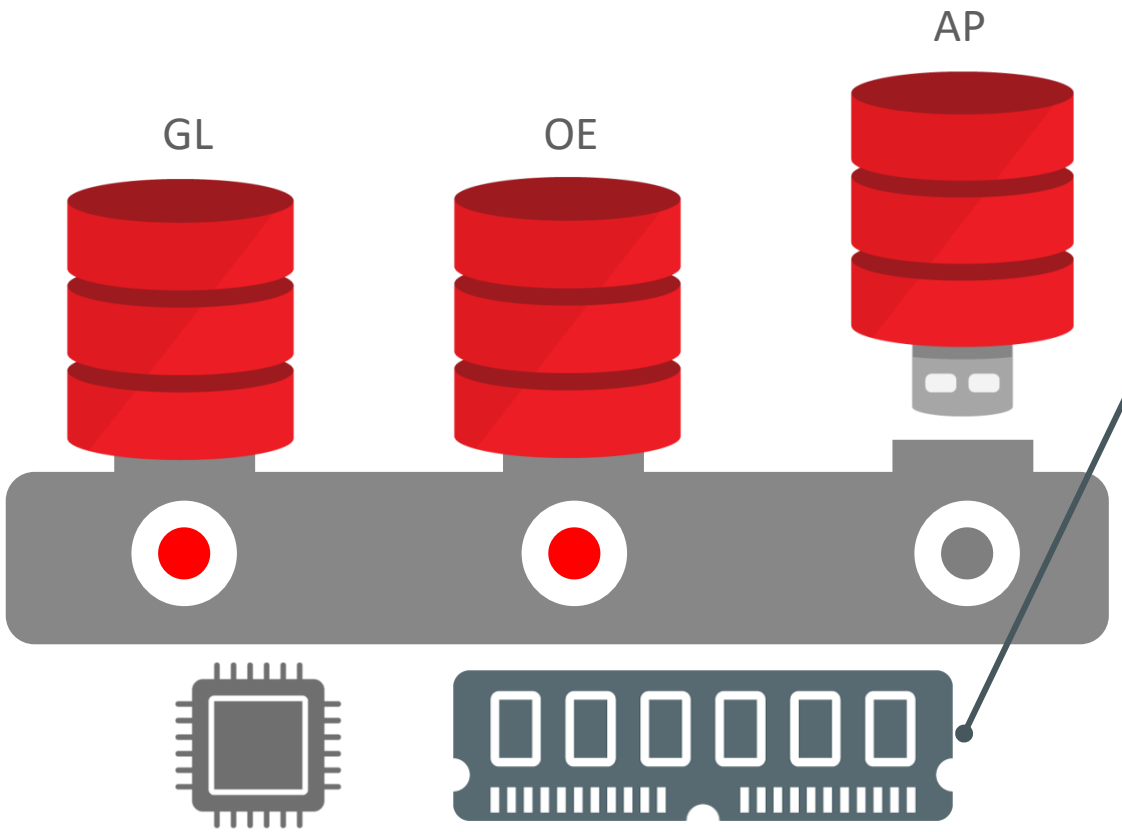


マルチテナント環境のデータ･ウェアハウス



- **containers**句を利用して横串検索
- SQL> select count(*)
  from ***containers***(sh.lineorder);
- SQL> select count(*)
  from ***containers***(sh.lineorder)
  where CON_ID in (45, 49);

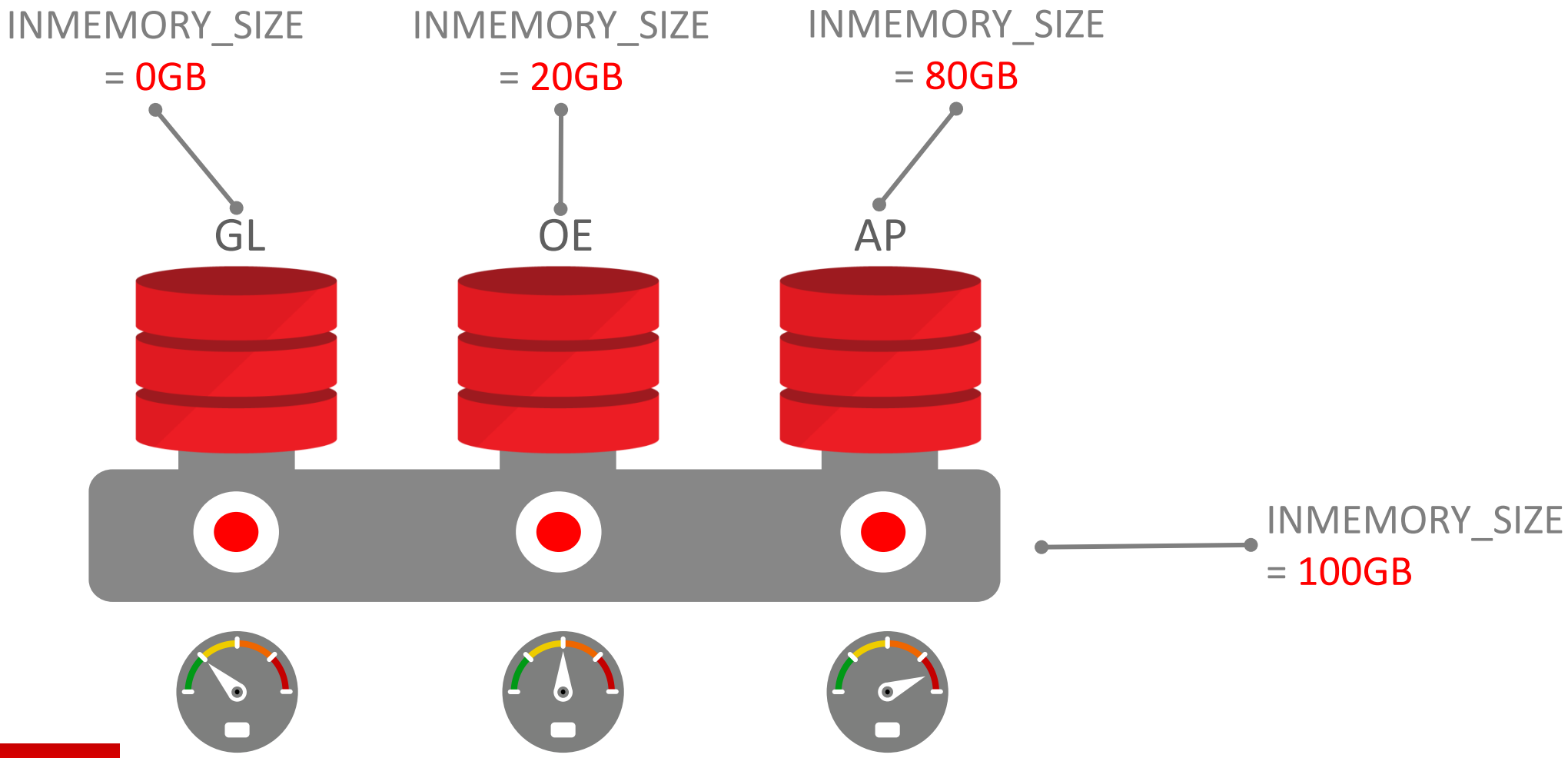
マルチテナント環境でのDatabase In-Memoryの利用



メモリーとバックグラウンド・プロセスの共有

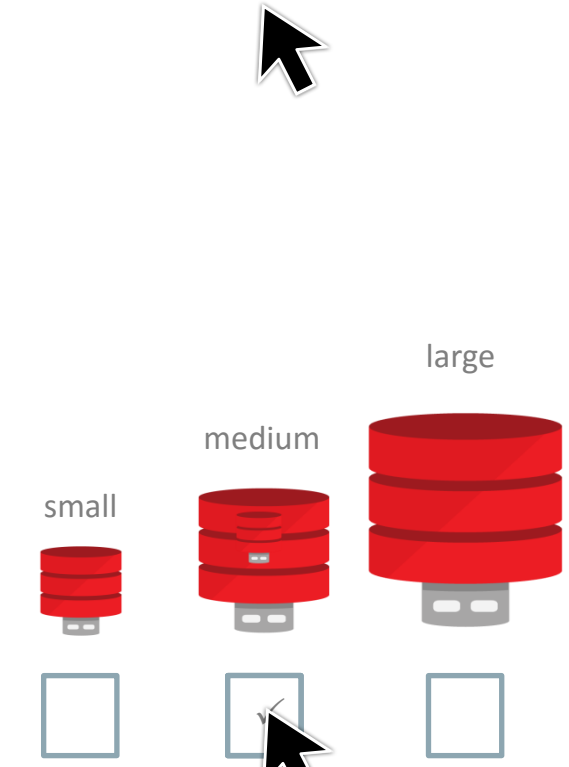
- In-Memory AreaはCDBレベルで設定
- INMEMORY_SIZE=100G
- 各PDBのINMEMORY_SIZEパラメータの値はデフォルトでは、CDBから引き継がれる

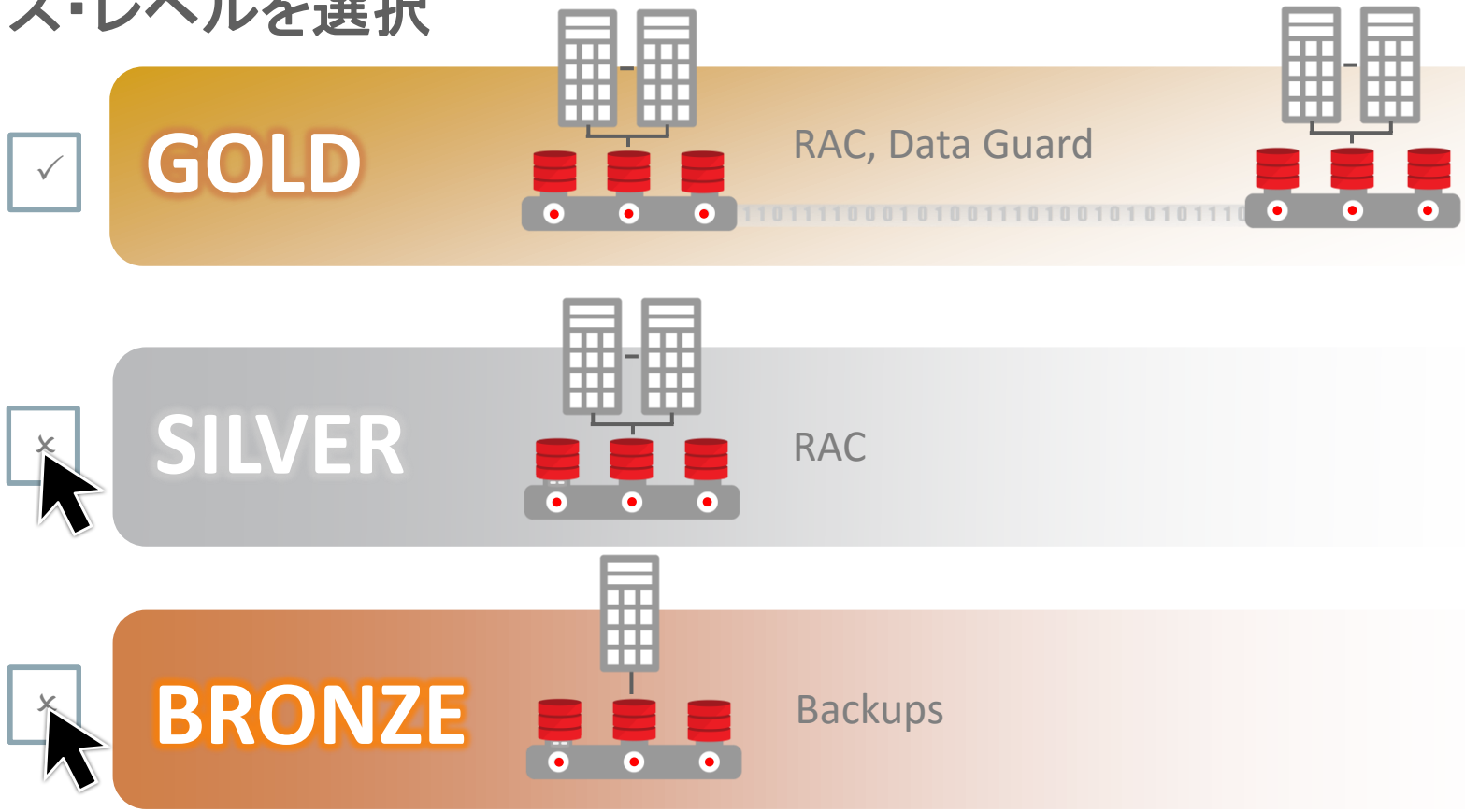
マルチテナント環境でのDatabase In-Memoryの利用



Oracle Multitenant によるDatabase as a Serviceの実現

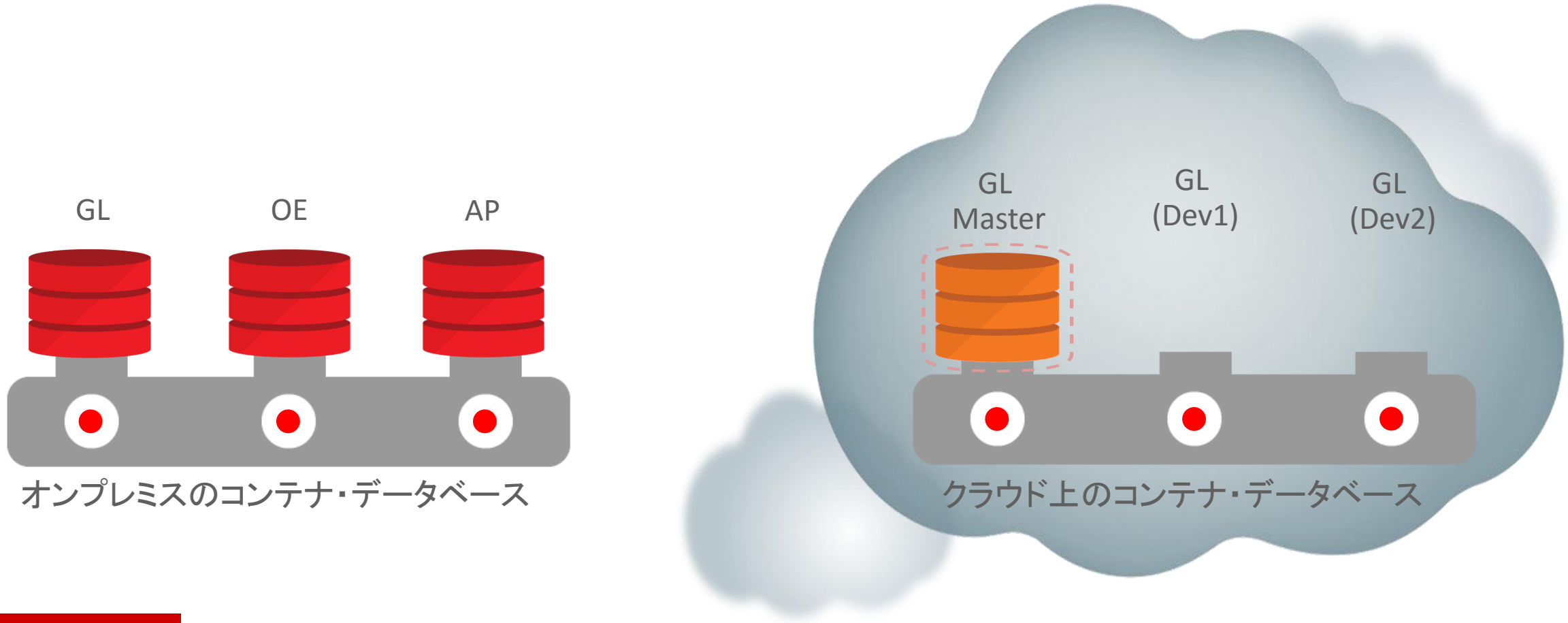
用途に合ったサイズとサービス・レベルを選択





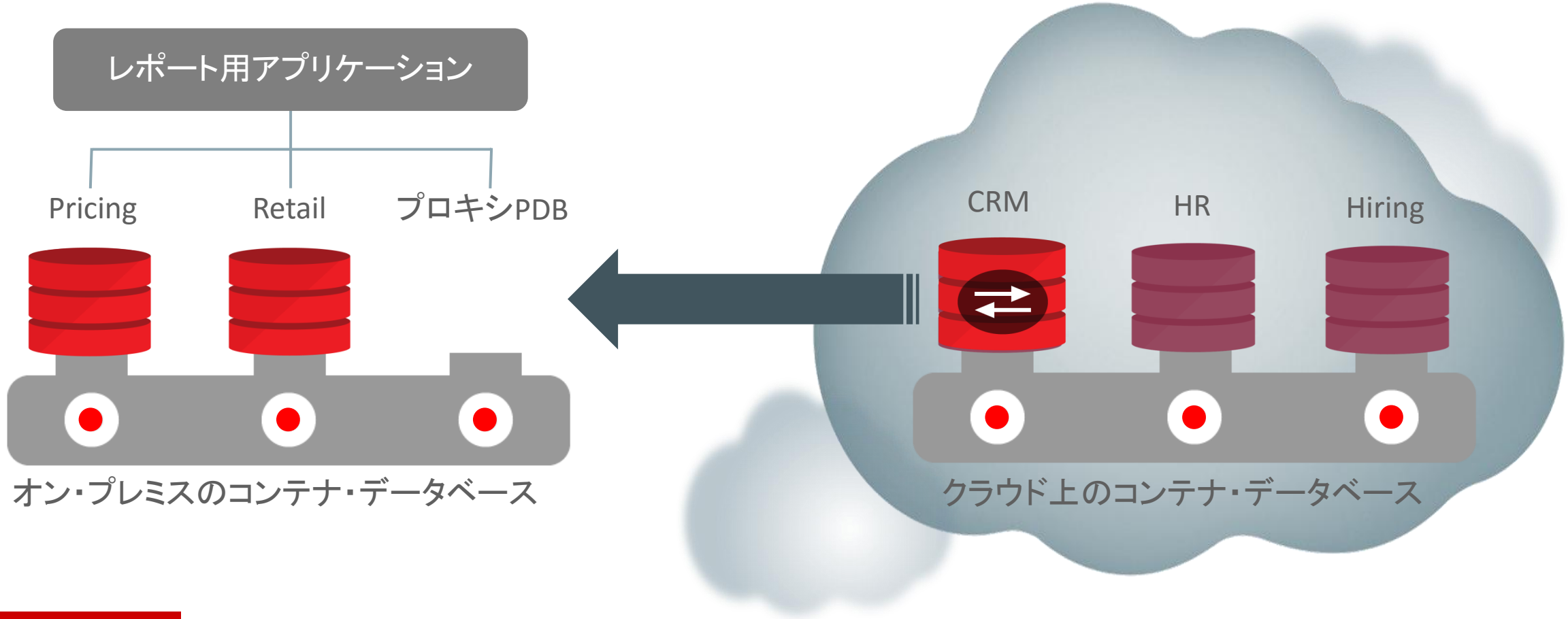
Oracle Multitenantによってクラウドへプラグイン

プラガブル・データベースはポータブル・データベース – クラウドへの移行も容易



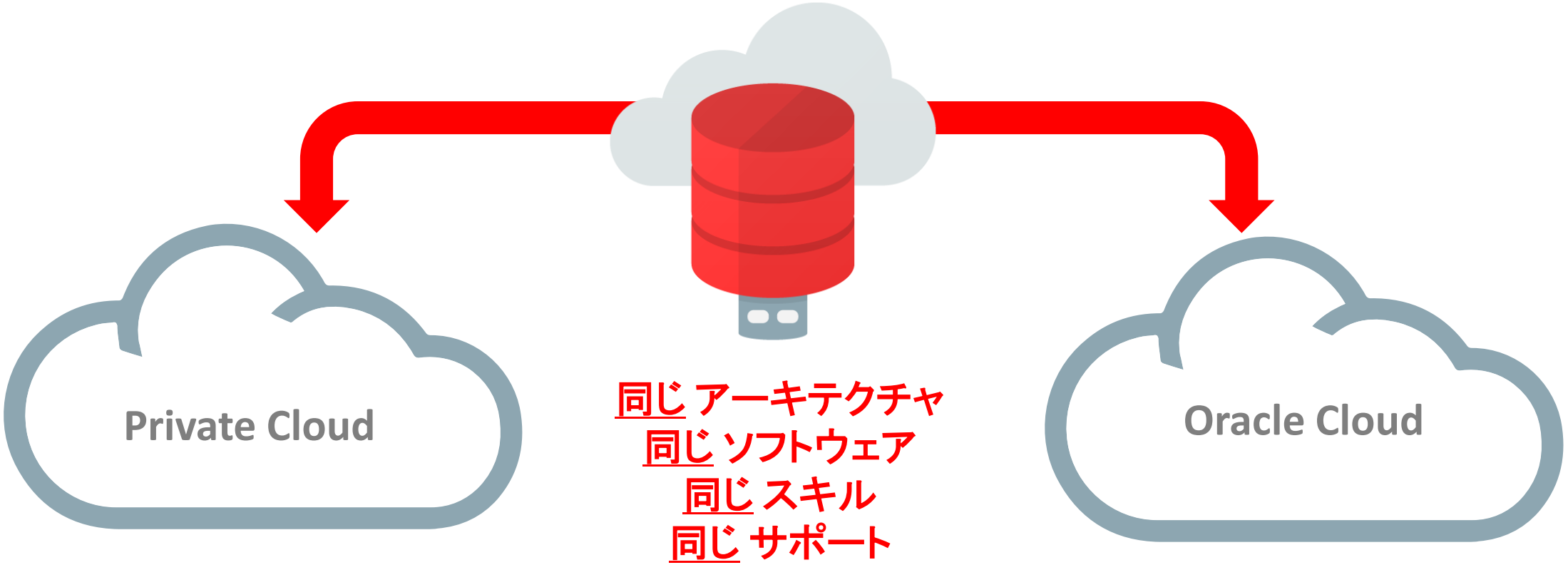
Cloud上のPDBをオンプレミスのPDBと同様に利用

プロキシPDBを使用によりデータの配置場所を柔軟に



完全にコンパチブルなハイブリット・クラウド

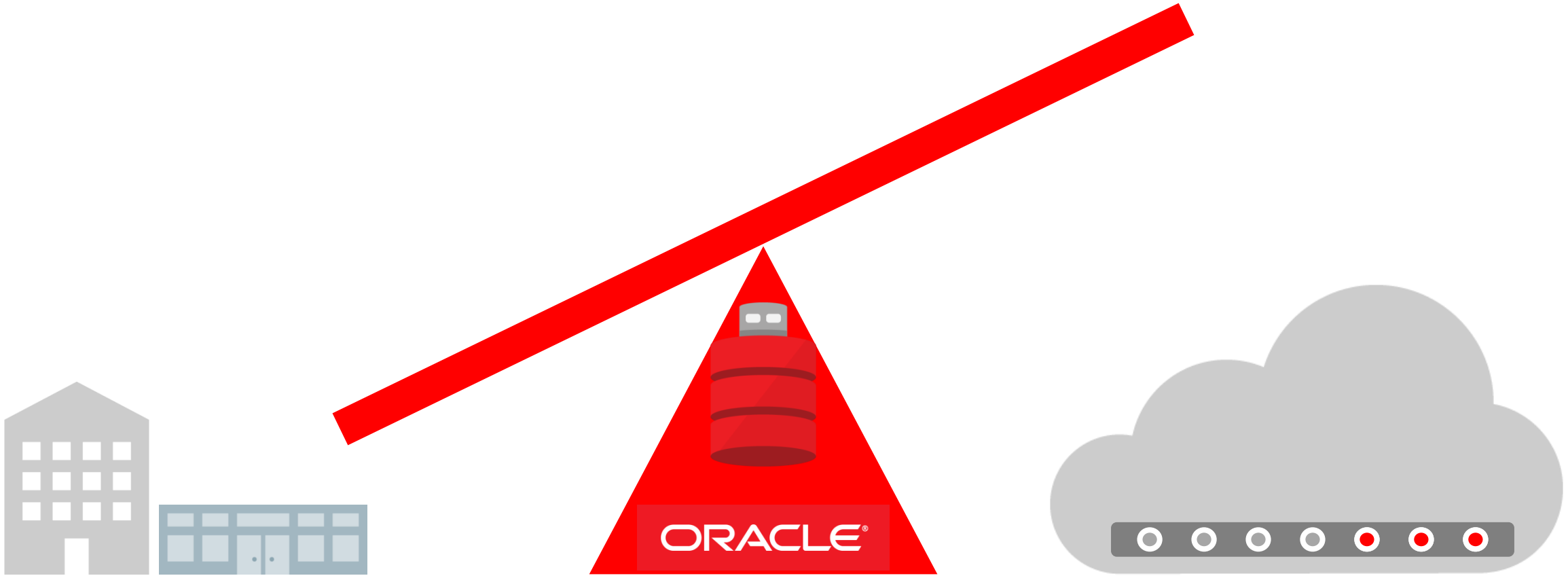
プライベート・クラウドとパブリック・クラウド間の共存と移行



オンプレミスとパブリック・クラウド間のアプリケーションとデータの自動化された移動

Multitenant:Cloudへのマイグレーションの支点

Cloudの基盤であり、Cloudへの移動手段でもある



Oracle Database Cloud Service
あらゆるサービスレベルに対応するラインナップ

Exadata Express

1 PDB

~50 GB

1 CPU

シンプルにデータ格納
DB運用不要

Database

1 DB

~12 TB

1~16 CPU

Oracle Database
そのまま使えるVM環境

Exadata

1 Exadata

~168 TB

16~272 CPU

Oracle Database
が超高速基盤で稼働

Exadata Express Cloud Service

Oracle Database(Pluggable Database) on Exadata を低価格で利用可能

ユーザーごとにPDBを提供

最新DB 最強基盤	Oracle Database 12c Release 2 (EE & Options) Exadata Machine
フルマネジド DB運用不要	Oracle がHWもDBも管理 DBAがいなくても利用可能
低価格で 利用可能	1ヶ月175ドル(20GB)からスタート可能

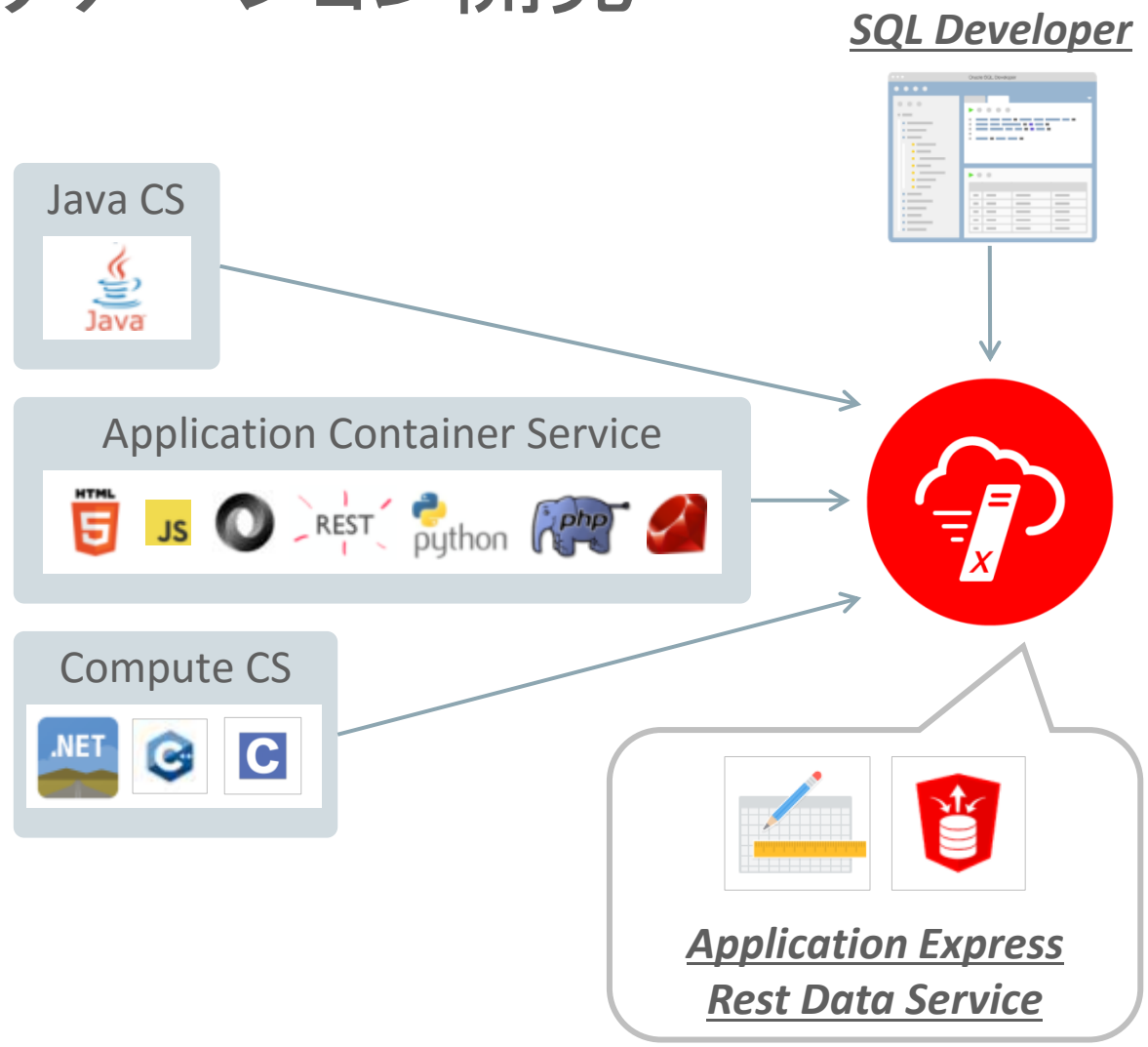
Exadata Express Cloud Service: 利用可能なリソース

Resource	X20	X50	X50IM
DB_PERFORMANCE_PROFILE	S20	S50	S50IM
Max CPU	1 core	1 core	1 core
CPU_COUNT (Hard CPU limit)	2	2	2
SESSIONS (Max sessions)	30	60	60
Storage	20 GB	50 GB	50 GB
PGA_AGGREGATE_LIMIT (Hard PGA limit)	3G	5G	5G
PGA_AGGREGATE_TARGET (PGA target, after which the PDB uses temp)	1.5G	2.5G	2.5G
SGA_TARGET (Hard SGA limit)	3G	5G	10G
Data Transfer	120GB/month	300GB/month	300GB/month
INMEMORY_SIZE	not available	not available	5G

** Oracle Database Exadata Express Cloud Service – Resource Restrictions http://bit.ly/2dEXDlA*

Exadata Express におけるアプリケーション開発

- 多様な開発言語をサポート
  - SQL*Net 接続可能
- 多様な技術をサポート
  - JSON
  - REST
- オラクル開発ツール群にも対応
  - Application Express
  - SQL Developer



お客様事例

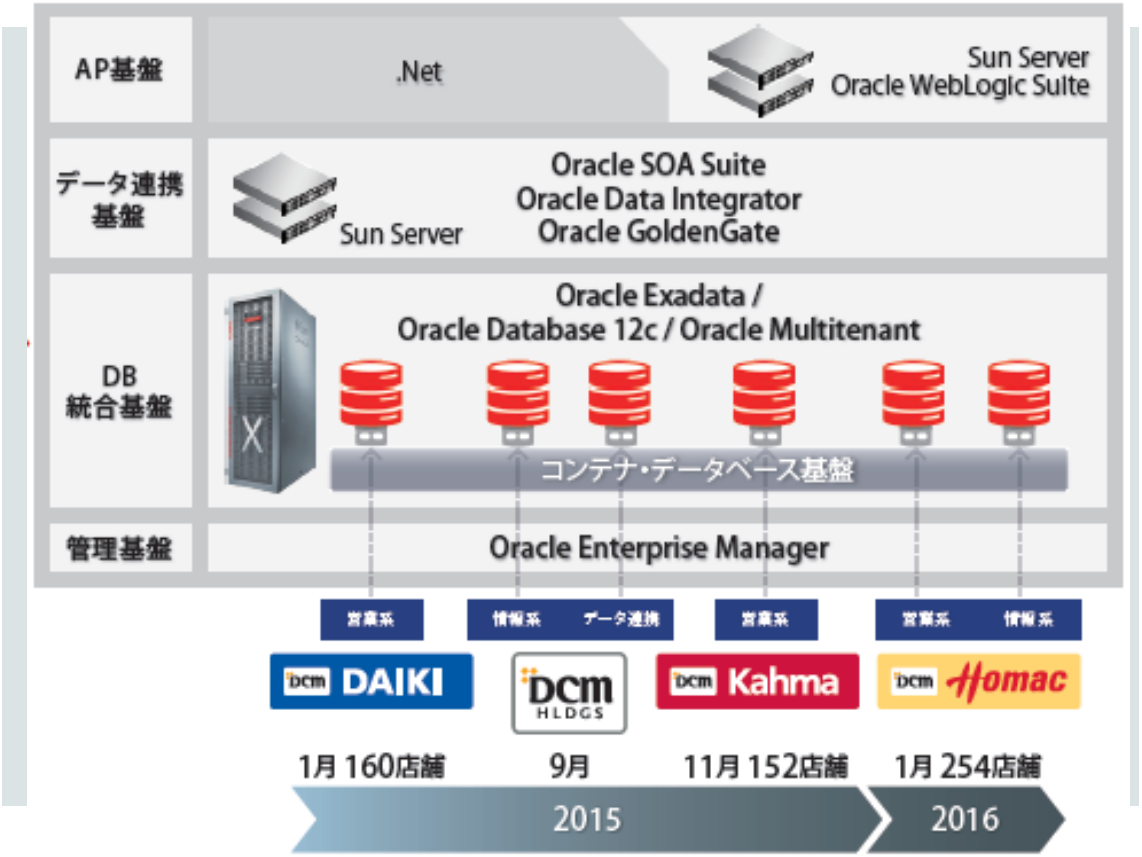
Oracle Database 12c マルチテナントがもたらすお客様価値
業務フローの統一に向けグループ各社のシステムを統合

DCMホールディングス 様

業務フローを統一し、一元化したデータの活用
「デマンドチェーンマネジメント」の実現

【導入のポイント】

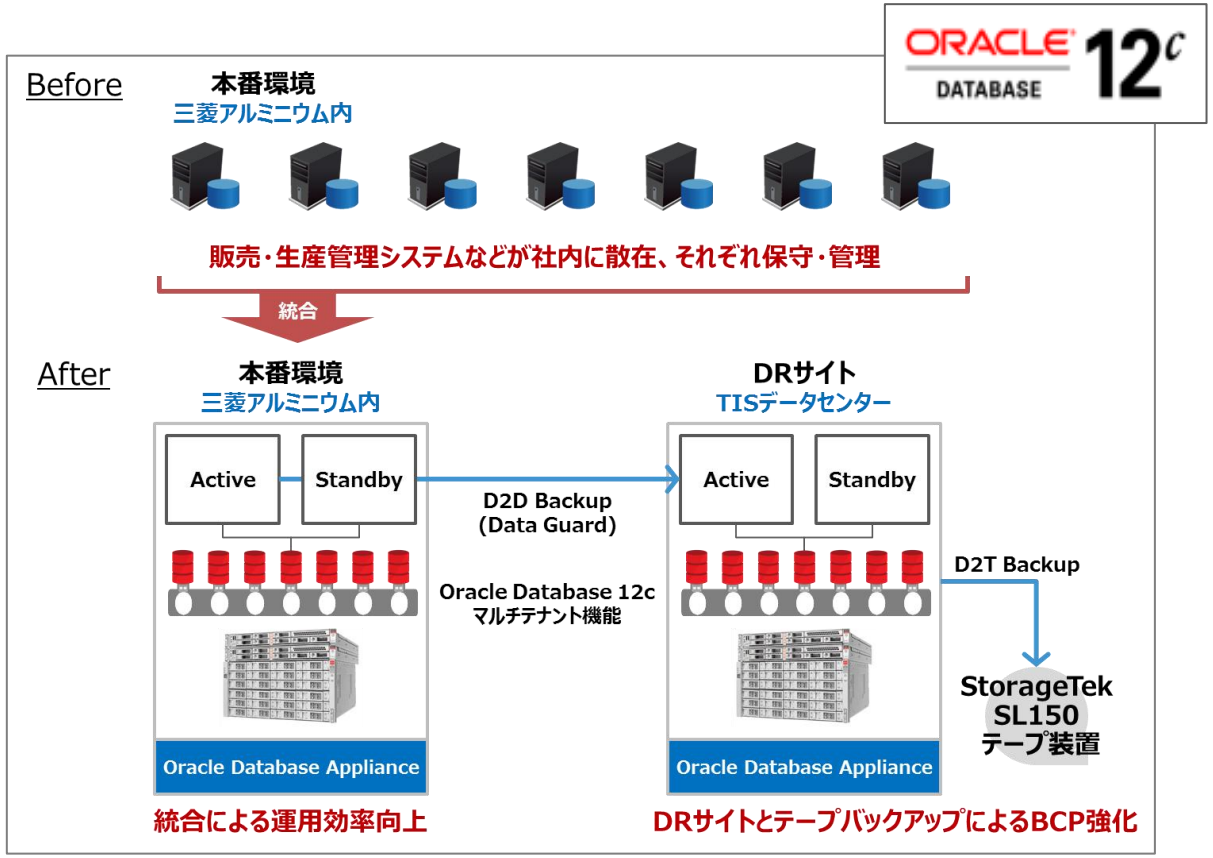
- マルチテナントを有効活用し、グループ各社が保有していた6DB環境を**独立性を維持しながら統合**
- **業務フローの統一に向けた基盤**が整備され、将来のM&Aや事業規模の拡大に柔軟に対応可能
- 初期導入費用を**最大で40%削減**（既存システム比）
  11本のサーバラックを4本へ縮小
- 夜間バッチ処理が**約2倍高速化**（約3時間短縮）
  商品改廃などの一連の業務処理の遅延を改善
- 店舗業務を**停止することなく移行**完了



Oracle Database 12c マルチテナントがもたらすお客様価値
販売・生産管理システムを統合し、事業継続計画の強化も実現

三菱アルミニウム様

- お客様のプロジェクトの目的
  - システム運用効率の向上と災害対策強化、システムレスポンス改善
- お客様の課題解決に向けた取り組み
  - 2015年10月より稼動、**12月までに販売管理や生産管理など5つのDBがマルチテナント機能を活用し統合。2017年4月までに合計7つの全てのDBが統合予定**
  - バックアップ、パッチ適用などの運用を一括実行することが可能となり、**従来の運用・保守費用（OPEX）を大幅に削減できることを期待**
  - **BCP（事業継続計画）強化:** 「Oracle Data Guard」を活用し、**自社とTISのデータセンターでDR（災害復旧）サイトを構築**
  - セキュリティ対策：「Advanced Security」による**バックアップ時の暗号化**



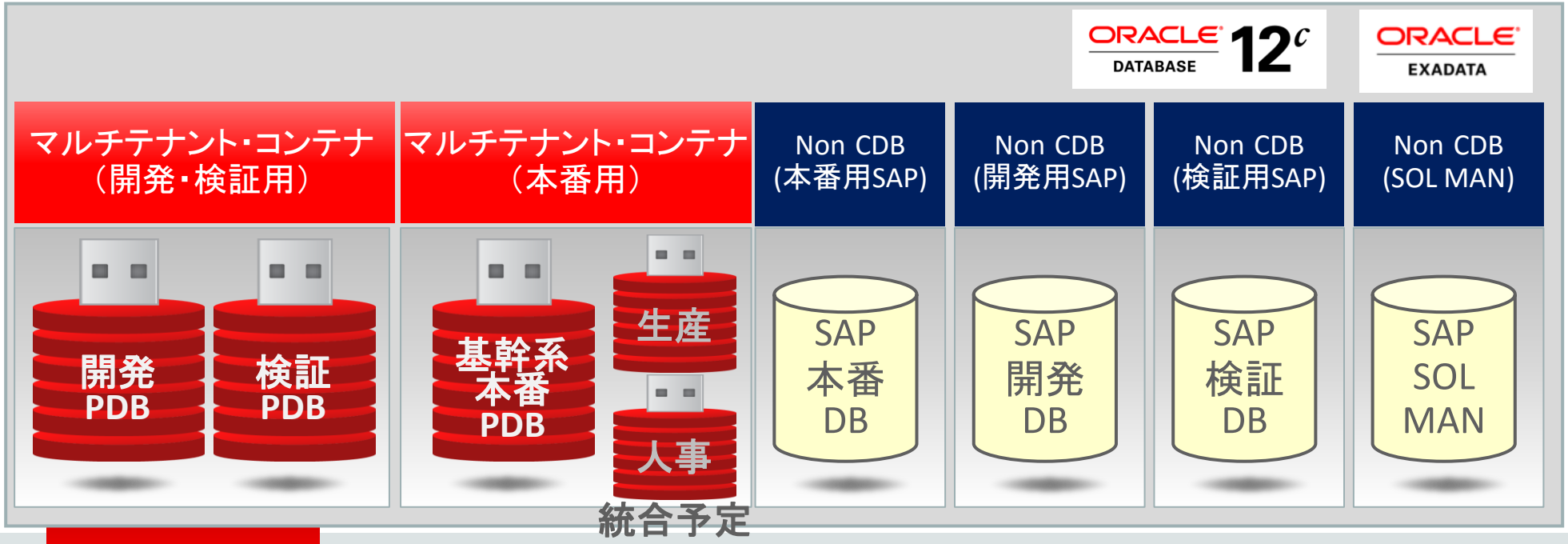
導入支援：TIS株式会社

Oracle Database 12c マルチテナントがもたらすお客様価値

システムを更改し、更なる性能向上と運用管理の強化

ライオン株式会社 様

- Exadata X5へ更改：Exadata V2上稼動していた環境（基幹業務やSAP用途のDB）
- オール・フラッシュ版のExadataによるさらなる性能向上
- Database 11gからDatabase 12cへのアップグレード・プロジェクト
- マルチテナント採用による運用の効率化（本番用、開発・検証用を稼働）



Oracle Database 12c マルチテナントがもたらすお客様価値
連結会社向けシステムをプライベート・クラウド環境で構築

パナソニックIS 様

システムの独立性を確保し、データベース統合による**高い運用効率とセキュリティを実現**

【 Before（統合前の課題）】

- 連結会社が活用：販売支援システム(16社)
- **運用の煩雑性**や複数DBの稼動による**リソース不足**

【 After（マルチテナントを活用した共通基盤）】

- **マルチテナントとExadataによる新クラウド基盤**
- **1コンテナ(CDB)に38個のPDBを統合＆稼働**
- 共通するマスタデータは、**共通化したPDBを活用**
- 運用コストの削減: **運用工数を約15分の1に**
- Exadataに統合:**平均150%、最大670%性能向上**
- **障害・データ破損から保護**「Data Guard」活用

第2部

5. まとめ

12c R2におけるOracle Multitenantの革新

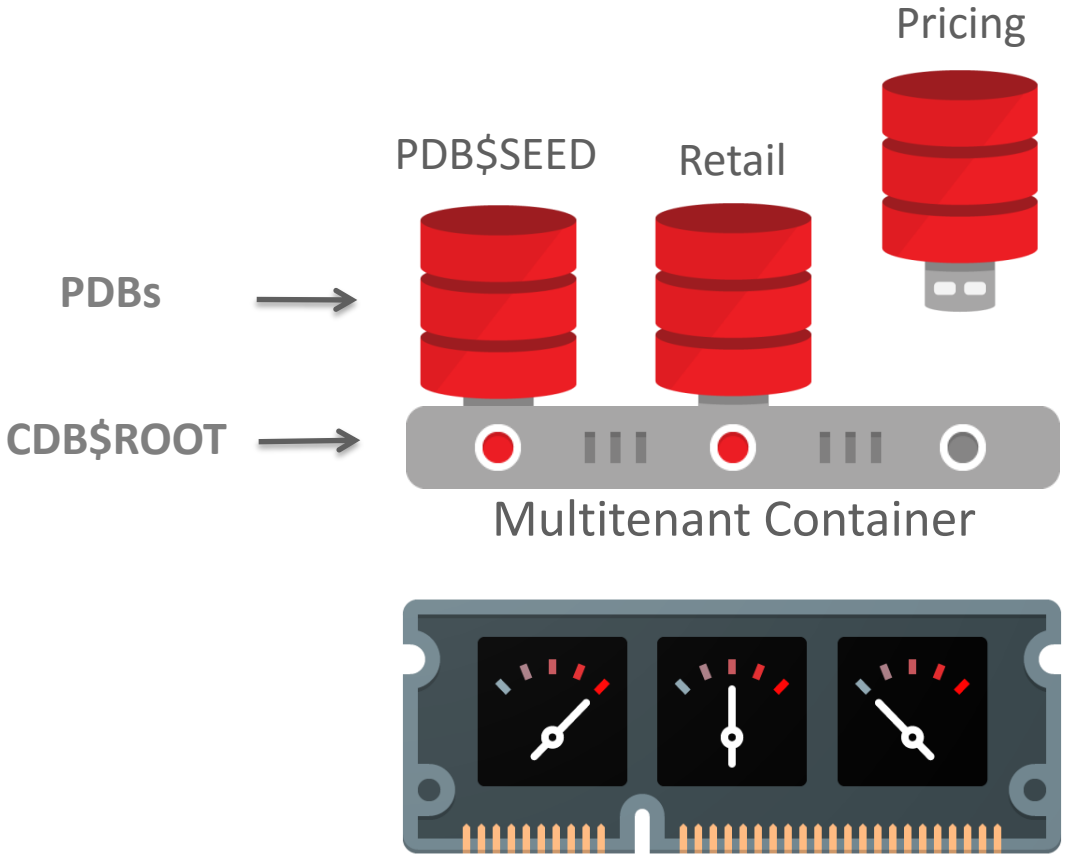
Oracle Multitenantを活用したデータベース統合 まとめ

- **オーバーヘッドの少ない統合、高い集約密度**
  - CPUやメモリーやバックグランド・プロセスを共有することでリソースを有効に活用でき、高いパフォーマンスと集約密度を実現
- **Manage many databases as one**
  - 標準化された構成を実現し、SQL文で管理、運用が行える
  - 共通する管理・運用業務をCDBレベルで実施
    - バックアップ、パッチ適用、アップグレード、高可用性構成 (RAC / DG)
- **容易な移行**
  - アプリケーションの変更は不要
    - Oracle Databaseの機能のほとんどがMultitenant Architectureで利用可能
  - システム間の高い独立性

合理化、標準化、自動化によるコスト削減

Oracle Multitenant

データベースの統合とシンプルな運用のための新しいアーキテクチャ



アプリケーションごとにPDBの提供

- ポータビリティ/可搬性 (プラガビリティ)
- 高速なプロビジョニング (クローン)
- アプリケーションの変更は不要

CDB$Rootで共通オペレーションの実施

- Manage many as one
  (アップグレード、バックアップ、HA構成)
- 細かい制御も可能

メモリーとバックグラウンド・プロセスの共有

- より多くのアプリケーションを稼働できる

Key Benefits

ベネフィット	有効な機能
CapExの最小化	• 1サーバー当たりのアプリケーション数
OpExの最小化	• Manage many as one (パッチ適用作業の削減) • 手順とサービス・レベルの標準化 • セルフ・サービスによるプロビジョニング
アジリティの最大化	• 開発・テスト環境用にスナップショット・クローニング • “プラガビリティ”による可搬性 • RACによるスケーラビリティ
容易	• 適用: アプリケーションの変更不要 • 利用: インターフェースはSQL

Appendix

Appendix

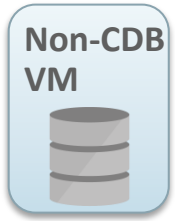
1. データベース統合手法ごとのパフォーマンス比較
   vs. 仮想マシンによる統合
   vs. DBの集積による統合(Non-CDB統合)
2. サーティファイ情報
3. 参考情報

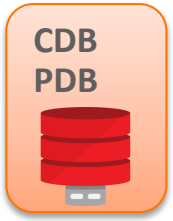
Appendix:
1. データベース統合手法ごとのパフォーマンス比較

データベース統合手法による比較検証

- **検証1:「仮想マシン」との比較**

Non-CDB VM
vs.
CDB PDB





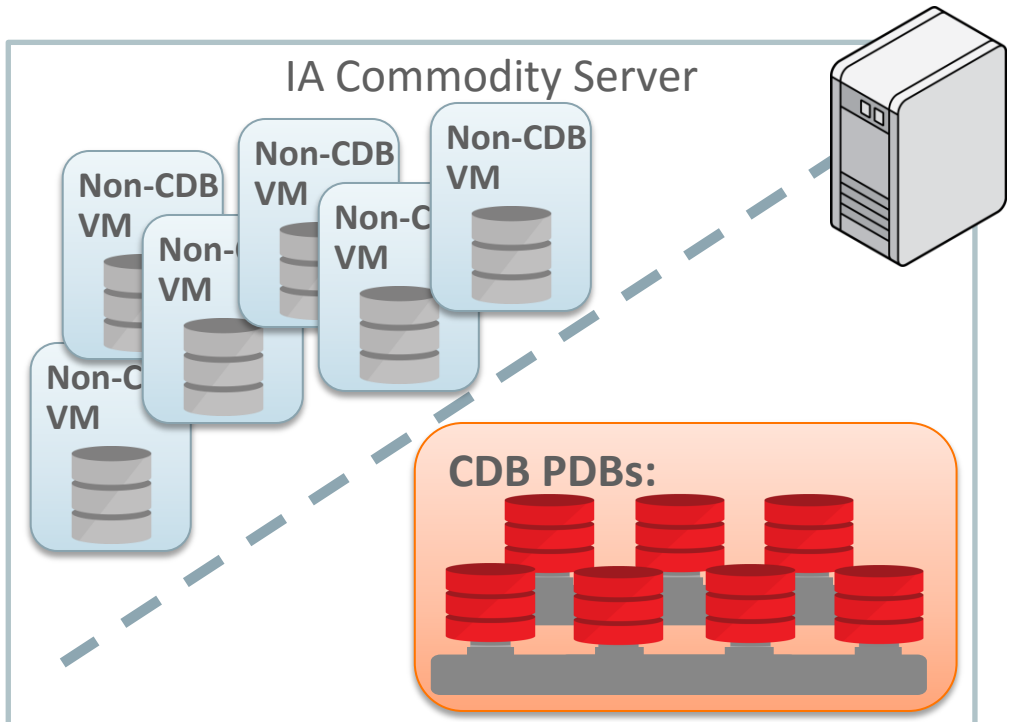
  - **環境**
    - Commodity IA Server
    - VM環境
  - **ワークロード**
    - Light weightな処理、低いTPS

- **検証2:「DBの集積」との比較**

Non-CDB
vs.
CDB PDB

  - Test 1: Cost 1 non-CDB vs. 1 PDB
  - Test 2: Efficiency 252 non-CDBs vs. 252 PDBs
  - Test 3: Density 168 non-CDBs vs. 252 PDBs
  - Test 4: Elasticity 8 non-CDBs vs. 8 PDBs
  - **環境**
    - Engineered System
      - **SuperCluster T5-8 / Exadata Storage Server**
  - **ワークロード**
    - OpenモデルのOLTP型の処理、高いTPS

検証1:「仮想マシン」との比較
Non-CDB VMによる統合 *vs.* CDB PDBによる統合 (on VM)



ワークロードの要件 :
- 平均 TPM <= 30 (1-2 TPS)
- 平均レスポンスタイム <= 500 ms
- 平均CPU使用率 75% まで
- 平均メモリー使用量 75% まで

Database Software :
- GI 12.1.0.2 Oracle Restart
- Oracle Database 12.1.0.2 Single Instance

System Configuration:
- 1 socket single 2.26 GHz core (Hyper Threadingは無効)
- 128GB Memory
- VMWare ESXi 5.1 (VM configuration)

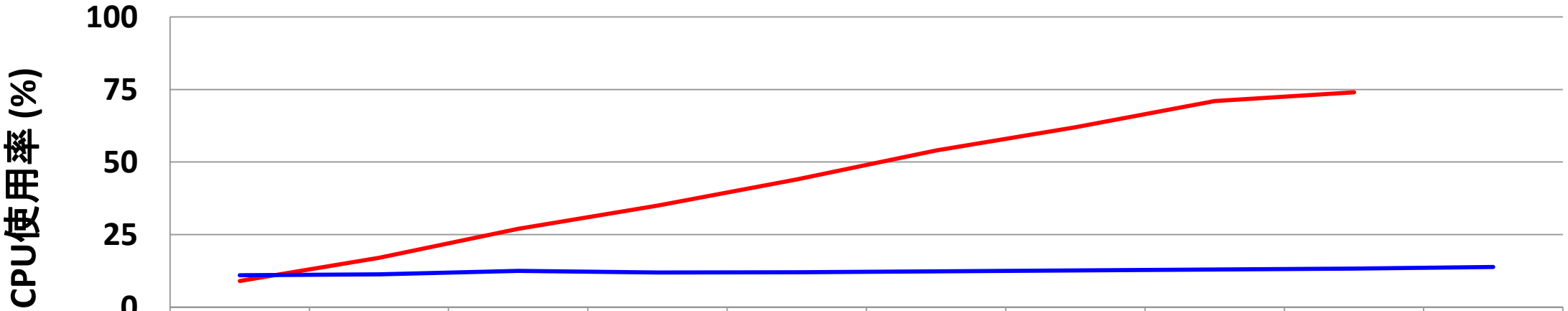
Non-CDB VM:
- 1 vcpu (2260MHz)
- 6GB memory per VM
- OS: Oracle Linux 6

CDB PDBs(on 1 VM):
- 1 vcpu (2260MHz)
- OS: Oracle Linux 6

Workload :
- Swingbench v. 2.3.0.422
- Order Entry (50% read / 50% write)
- 最大同時接続数 : 2

軽量なWorkloadを想定した検証

Non-CDB VM *vs.* CDB PDB : データベース数とCPU使用率



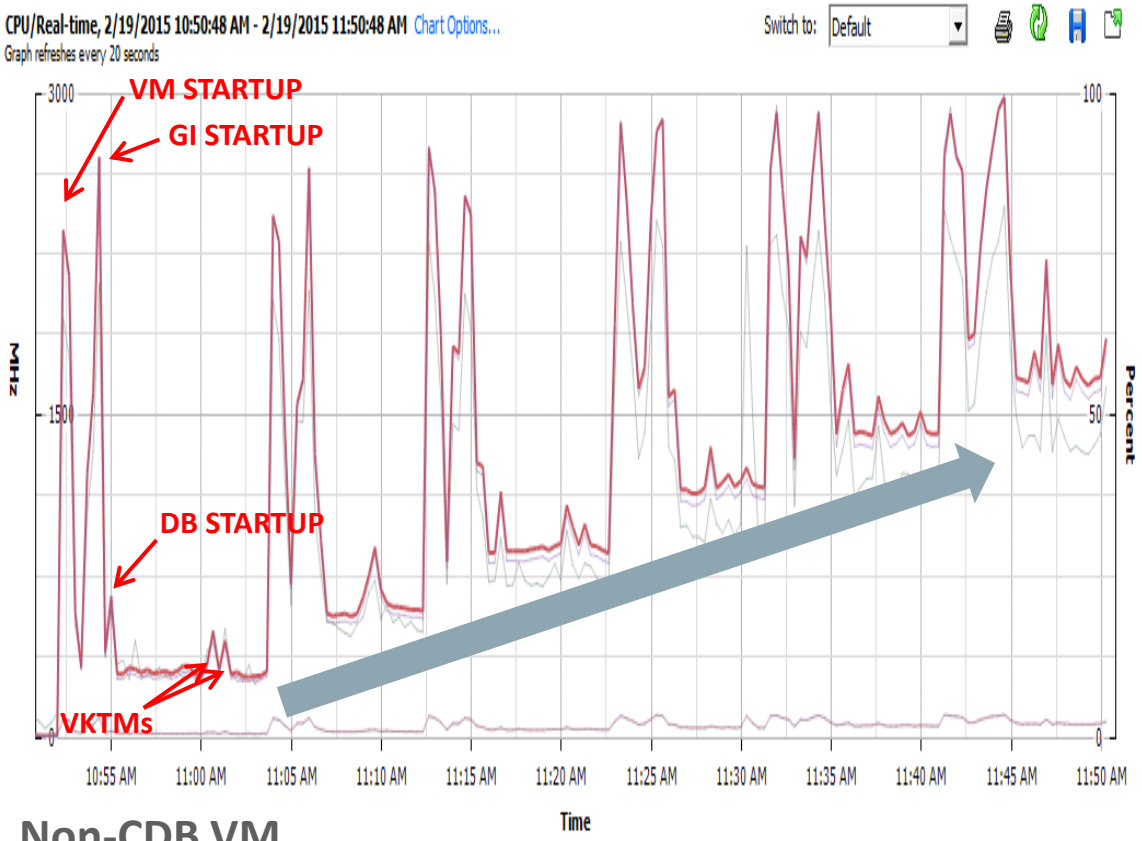
	1	2	3	4	5	6	7	8	9	10
non-CDB VM	9	17	27	35	44	54	62	71	74	
CDB PDBs	11	11.34	12.52	11.97	12	12.35	12.67	13	13.3	13.8

データベース数とCPU使用率

ワークロードの要件 :
- 平均 TPM <= 30 (1-2 TPS)
- 平均レスポンスタイム <= 500 ms
- 平均CPU使用率 75% まで
- 平均メモリー使用量 75% まで

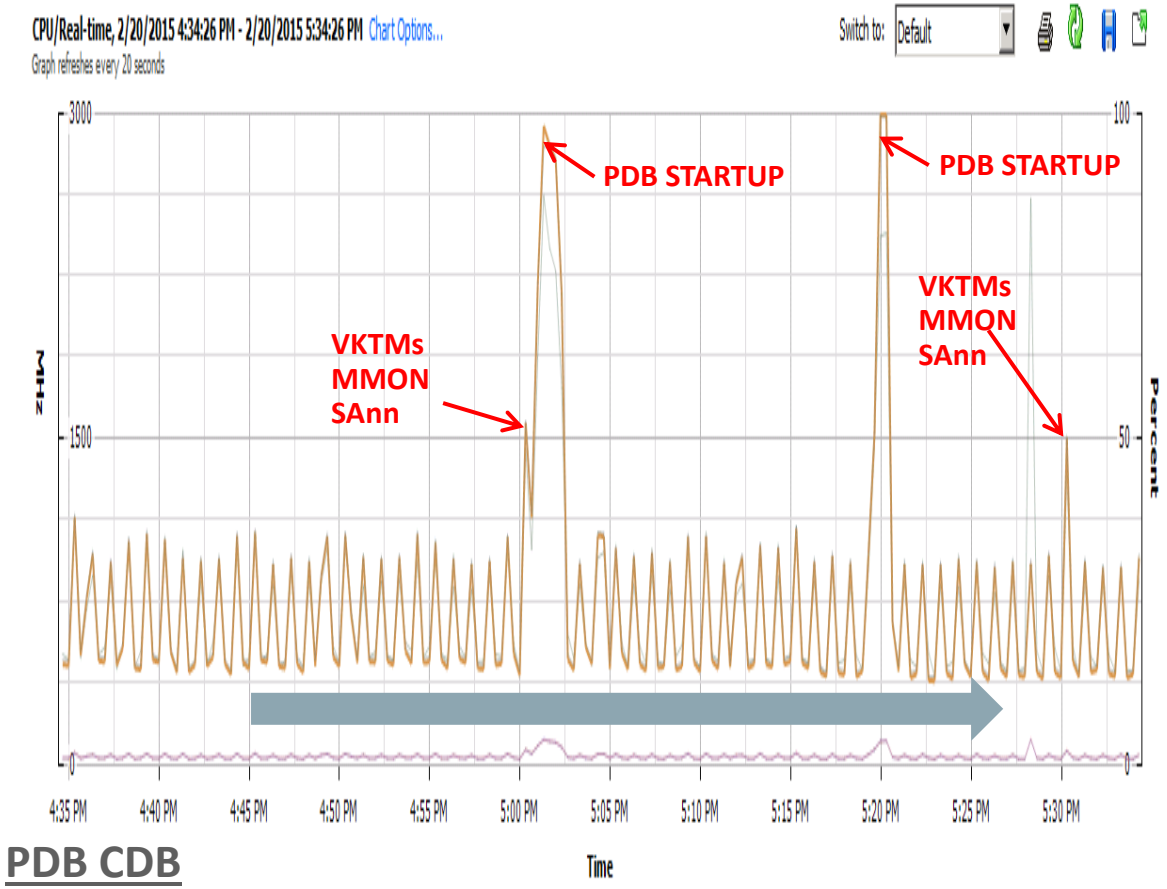
- Non-CDB VMは各VMを起動するごとに約9%のCPU使用率と6GBのメモリー使用量が増大
  9VMを起動した時点で、要件である CPU使用率75% に抵触
- CDB PDBsは、PDBの起動ごとのCPUとメモリー・リソースの消費はわずかに抑えられている

Non-CDB VM *vs.* CDB PDB : 起動時のCPU使用率の遷移



Non-CDB VM

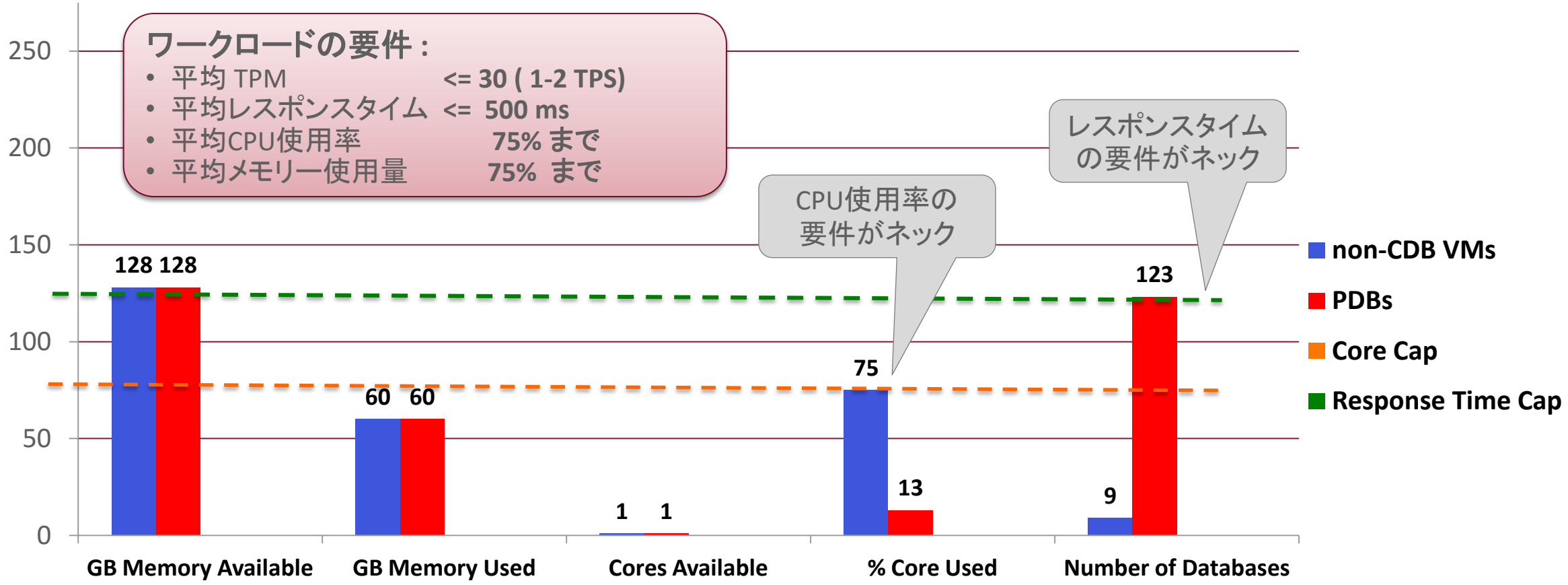
- VMが起動するたびにCPU使用率が高くなっている
- 6VMを起動した時点で50%以上を消費している



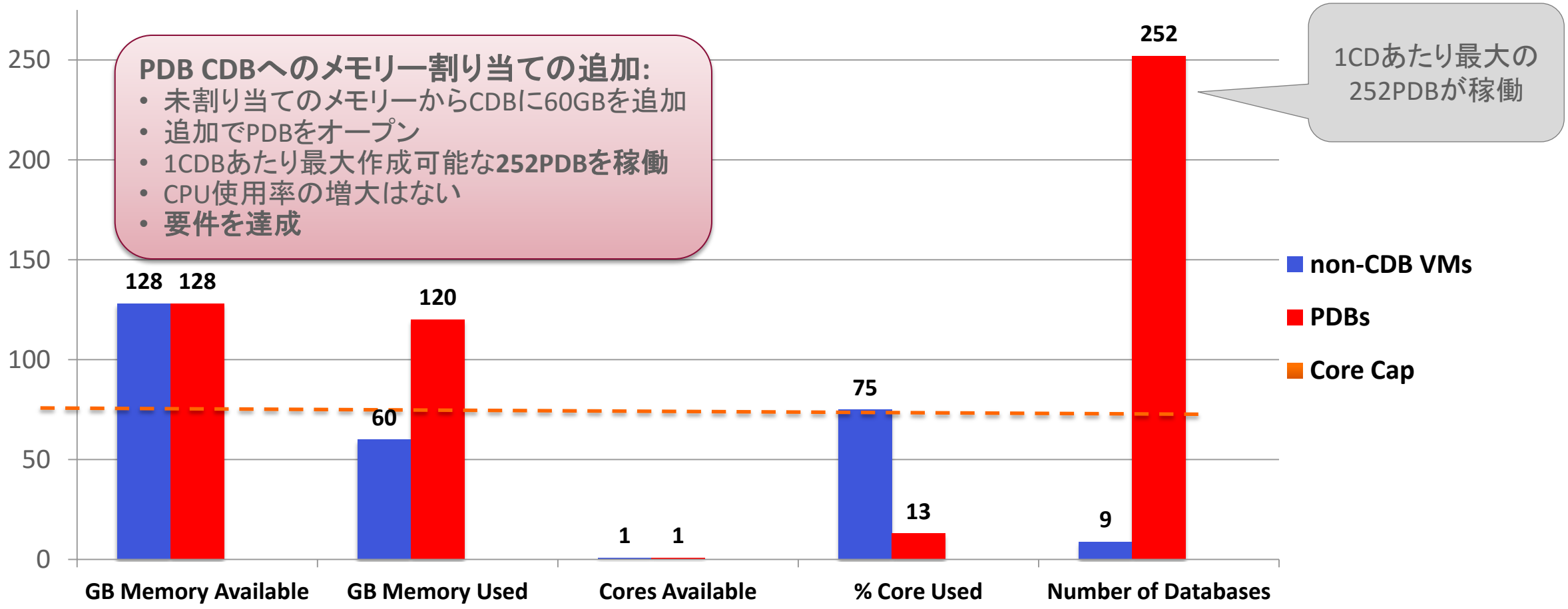
PDB CDB

- PDBの起動する前と後の平均CPU使用率に大きな差はない
- 10 PDB以上起動させた場合でもCPU使用率は低く保てている

Non-CDB VM *vs.* CDB PDB : 統合できるデータベース数



Non-CDB VM *vs.* CDB PDB : 統合できるデータベース数



VM(仮想マシン)による統合と比べた Oracle Multitenantによる統合の優位性

- **より多くのデータベースを統合可能**
  - オーバーヘッドの少ない統合
    - Hyper Visorを介さず、OSの数も増えない
      - Hyper VisorをOSによるリソースの消費がない、管理対象も増えない
    - CPU、メモリーなどのリソースを有効に活用できる
      - 余剰リソースを減らすことができ、必要なPDB間で共有することが可能

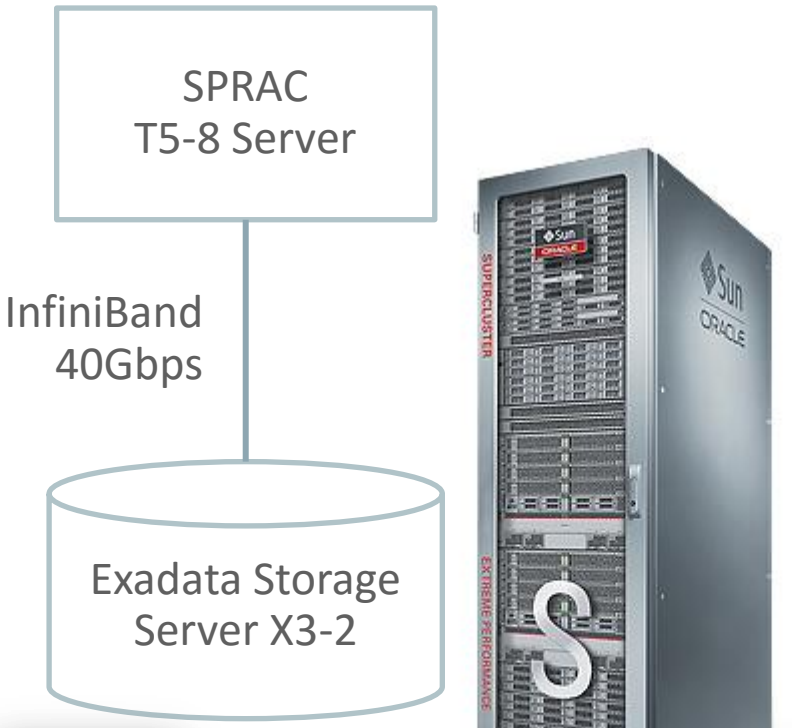
高いコスト・パフォーマンス
(H/W、S/Wライセンス・コストの節約)

- **データベースのアーキテクチャ、仕組みをベースにした統合**
  - 共通する操作はCDB$Rootで行える (Manage Many as One)
  - Hyper Visorレイヤーを含まないため、シンプルなシステム構成
  - 可用性: 少ないリソースで、より確実で、高い可用性を実現
    - リソース再配置 / フェイルオーバー/ DR構成

運用コストの削減

VM環境でMultitenant構成を利用することも可能
データベース・レイヤーの統合はOracle Multitenantを利用し、高い集約効率を実現

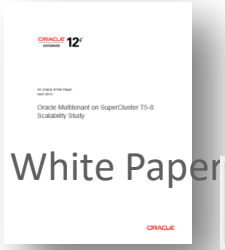
検証2:「DBの集積」との比較
Non-CDBによる統合 *vs.* CDB PDBによる統合(on Engineered System)



Database Server :
- SPRAC T5-8 Server * 1
- 8 sockets, 16-cores/socket, 3.6 GHz SPARC T5 processor (128 cores)
- 2 TB Memory
- Solaris 11 Update 1 SRU 16.
- Oracle 12.1.0.2.0 (release candidate)
- Automatic Storage Management (ASM) with normal redundancy
- Shared ORACLE_HOME for non-CDBs

Storage Server :
- X3-2 Exadata Storage Server * 8
- Software version OSS 11.2.3.3.0

Workload :
- 72% write (insert/update/delete) transactions
- 28% read-only transactions (light-weight selects, mid-weight scans)



White Paper

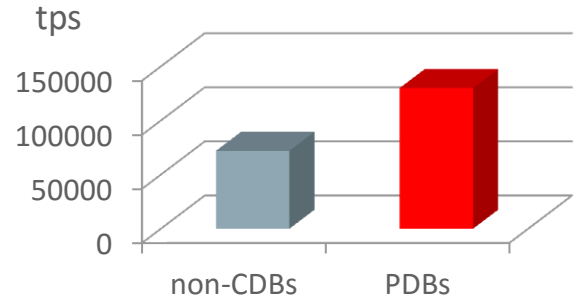
oracle multitenant スケーラビリティ

出典: White Paper: Oracle SuperCluster T5-8 Oracle Multitenant データベース統合の効率に関する事例
http://www.oracle.com/technetwork/jp/database/multitenant/learn-more/oraclemultitenantt5-8-final-2185108-ja.pdf

サマリー: データベース統合テスト

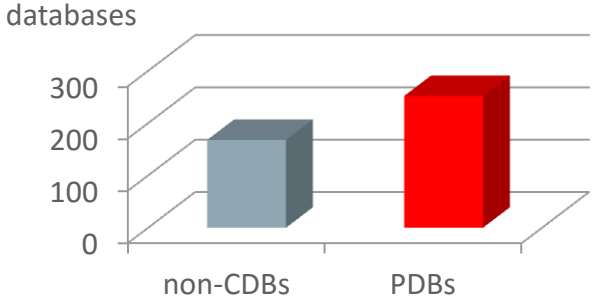
Oracle Multitenantは少ないシステムリソース消費で高いパフォーマンスを実現

スループット
252 non-CDBs vs. 252 PDBs



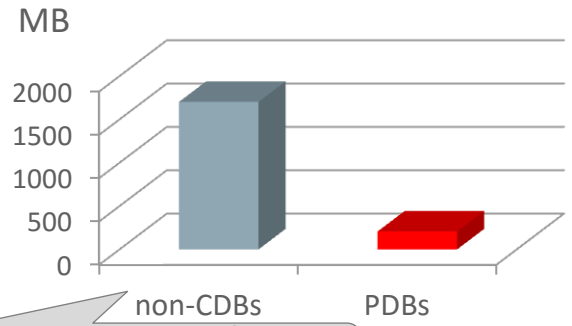
80%高い
スループット

データベース数
(1DBあたり同じスループット)



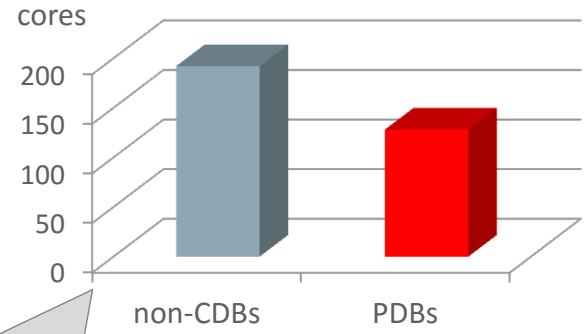
1.5倍の数の
データベース
を統合可能

データベースあたりの
メモリー・フットプリント
(バッファ・キャッシュは除く)



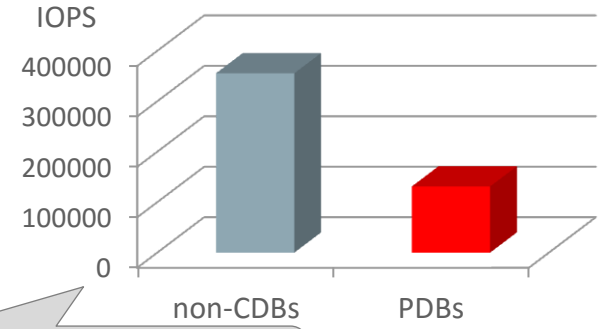
メモリーのフットプリント
を1/8に削減

252 DB/PDBを稼働させるために
必要なコア数



64コア少なく実現

252 DB/PDBを稼働させるために
必要なストレージIOPS



ストレージIOPSを1/3
に削減

252 non-CDBs *vs.* 252 PDBs リソース消費とパフォーマンスの分析

Oracle Multitenant: より少ないリソース消費でより高いパフォーマンスを達成

システム	トランザクション		CPU使用率	ストレージIOPS		
	スループット	レスポンスタイム	合計	物理読み込み	物理書き込み	Redo ログ書き込み
252 non-CDBs	72,600 tps	7 ms	68%	36,900 r/s	133,100 w/s	101,400 w/s
252 PDBs	130,300 tps	10 ms	68%	25,400 r/s	100,400 w/s	5,400 w/s
	+80%	log file sync	same	30%	25%	19x

プロセス	フォアグランド・プロセス			バックグラウンド・プロセス		
	プロセス数	CPU使用率	PGAサイズ	プロセス数	CPU使用率	PGAサイズ
252 non-CDBs	2688	35%	50 GB	8387	26%	149 GB
252 PDBs	2688	60%	50 GB	91	3%	2 GB
	+74%			92x	9x	75x

メモリー	SGA		PGA	合計	データベースごとのメモリー・フットプリント
	バッファ・キャッシュ	その他のプール	合計PGAサイズ	合計メモリーサイズ	バッファ・キャッシュとフォアグランドを除く
252 non-CDBs	964 GB	270 GB	199 GB	1433 GB	1702 MB
252 PDBs	1250 GB	49 GB	52 GB	1351 GB	208 MB
	+30%	6x	4x		8x

non-CDBs *vs.* PDBs
パフォーマンス要件と集約できるテナント数

Oracle Multitenant: より多くのテナントを集約し、パフォーマンス要件を満たすことが可能

- **マルチテナント・アーキテクチャでは、1.5倍多くのデータベースを統合可能**

	テナントごとのスループット要件			目標とするCPU使用率要件	稼働させられたテナント数
	Small	Medium	Large		
non-CDBs	97 tps	485 tps	970 tps	<= 70%	168
PDBs	97 tps	485 tps	970 tps	<= 70%	252

+50%

- **Non-CDB上で252テナントを稼働させるには、より多くのリソースが必要**
  - 追加で64コア (→ 追加でもう1台物理サーバーが必要)
  - 3倍のストレージIOPS

	252テナントを稼働させるためのハードウェア要件			
	コア数		ストレージIOPS	
non-CDBs	192 cores		355,500 IOPS	
PDB	128 cores		131,200 IOPS	

33%

3x

DBの集積による統合と比べた Oracle Multitenantによる統合の優位性

高い集約率

- リソースの効率的な利用
  - CDB構成は**バックグラウンド・プロセスの数**を抑えられる
    - サーバー・プロセスがSQL処理に使えるCPUリソースが増える
    - コンテキスト・スイッチの発生回数が低減
  - 共有プールおよびバックグラウンド・プロセスが使う**PGAの合計サイズ**が抑えられる
    - 共有プールを複数のPDBで利用
      - SQLカーソルの有効利用
    - メモリーのフットプリントが軽くなり、余剰をバッファ・キャッシュに割り当てられる
      - バッファ・キャッシュ・ヒット率の向上
  - バッファ・キャッシュをPDB間で共有するため、**特定のPDBの急なワークロード増加に強い**

高いパフォーマンス

- パフォーマンス特性
  - 同じ数(252個)のDB(non-CDB構成)/PDB(CDB構成)の比較では、CDB構成が**より少ないリソース消費でより高いパフォーマンス**を実現
  - REDOログやダーティ・バッファの書き込みが**複数のPDB分**をまとめて効率よく行われる
    - I/Oサイズが大きくなり、ストレージIOPSが下がる
    - scalable log writer architecture
      - 複数のLog Writer Slave(LG*nn*)による並列書き込み

Appendix:
2. サーティファイ情報

Oracle Multitenant構成をサーティファイ済みの Oracle Applications

Oracle Applications

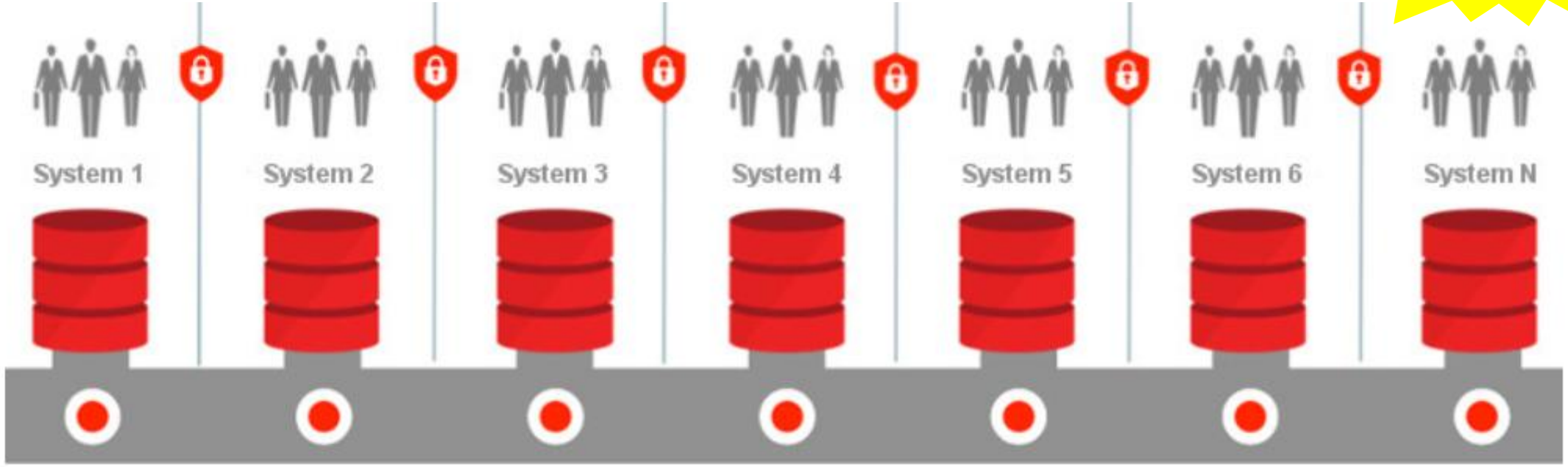
- Fusion Applications
- Applications Unlimited
  - Siebel
  - PeopleSoft
  - JD Edwards
- Oracle Cloud Applications
  - Taleo Business Edition
  - RightNow

Oracle Applications (continued…)

- Oracle Argus Analytics
- Oracle Utilities Mobile Workforce Management
- Oracle FLEXCUBE Direct
- Oracle Argus Mart
- Supply Chain
- Oracle Utilities Customer Care and Billing
- Oracle Argus Insight
- Oracle Utilities Work and Asset Management
- Retail Analytics
- Oracle Argus Safety
- Planning Non-RPAS
- Oracle Utilities Smart Grid Gateway
- Merch Suite
- Oracle Utilities Meter Data Management System
- Stores Suite
- …

Oracle Multitenant Option for SAP Customers

2016年
7月



Oracle Multitenant for SAP in Early Customer Shipment Status

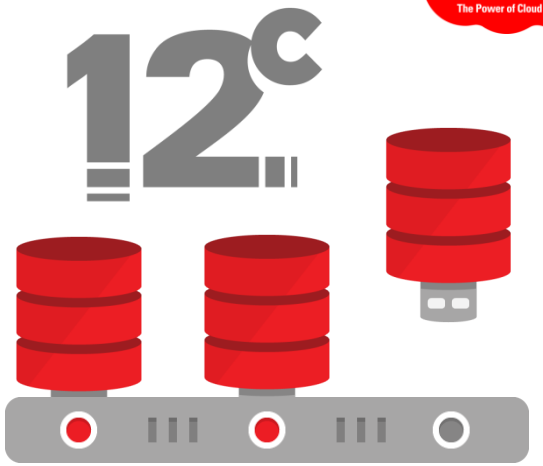
Oracle Multitenant, the last major Oracle Database 12*c* option to be certified for SAP environments, is now available with early customer shipment status.

Appendix:
3. 参考情報

Learn Oracle from Oracle

POCO
The Power of Cloud by Oracle

マルチテナント・アーキテクチャを勉強しよう。



オラクルユニバーシティでは、マルチテナント・アーキテクチャの概念からマルチテナント環境におけるデータベース管理方法まで 、DBA に必要とされるスキルを幅広く習得することができる研修コースを提供しています。注目のマルチテナント機能をわかりやすい講義と実機演習を通して
じっくり・しっかり学習することができます。

Oracle Database 12c: マルチテナント・アーキテクチャ

コース概要	このコースでは、Oracle Database 12c の新機能であるマルチテナント・アーキテクチャのすべての機能について学習します。マルチテナント・コンテナ・データベースおよび関連付けられたプラガブル・データベースのコンポーネントに関する詳細な情報から、ビジネス・アプリケーションに適した記憶域構造を持つマルチテナント・コンテナ・データベースおよび関連付けられたプラガブル・データベースを作成および管理する方法を、実践的な演習に加えて通して習得することができます。また、共通ユーザーとローカル・ユーザーを作成して、ビジネス要件に合わせてデータベース・セキュリティを管理する方法についても説明します。
コース内容	■はじめに ■コンテナおよびプラガブル・データベースのアーキテクチャ ■CDBおよびPDBの作成 ■CDBおよびPDBの管理 ■CDBおよびPDBの記憶域の管理 ■CDBおよびPDBのセキュリティの管理 ■可用性の管理 ■パフォーマンスの管理 ■その他
対象者	• データベース管理者
前提条件	• 「Oracle Database 12c: 管理ワークショップ」コース受講相当の知識 • 「Oracle Database 12c: バックアップ・リカバリ」コース受講相当の知識
日程 (2016年12月現在)	2日間

【オラクルユニバーシティ研修の特長】

わかりやすい
経験豊富なインストラクターがわかりやすくマルチテナントを解説します。

じっくり
実機を演習でマルチテナントの機能をじっくり確認できます。

選べる受講形態
集合研修に加えて Oracle トレーニング・オンデマンドでも受講可能。多忙なエンジニアの皆さまも効率的にスキル・アップできます。

オラクルユニバーシティ お問合わせ窓口
ORACLE UNIVERSITY
TEL 0120-155-092
URL http://www.oracle.com/jp/education/

ORACLE

オラクル データベース インサイダー

データベース関連製品の導入検討に役立つ情報サイト

PR

ORACLE

@IT Special

8システム／5人の運用体制が170システム／7人に：

運用工数を15分の1に削減！ パナソニックISが実践するOracle Exadataとマルチテナントを活用した大規模DB統合のアプローチ

パナソニックグループのIT中核会社として同グループのIT企画／運用を主導するパナソニック インフォメーションシステムズは、Oracle Exadataを利用してグループ内のデータベース環境を統合。

基幹DBのBCP対策強化、管理性と性能の大幅向上を実現：

三菱アルミニウムがマルチテナント機能で販売／生産管理データベースを一挙集約し、事業継続性も強化。その選択の理由とは？

三菱アルミニウムは2015年12月、中核生産拠点となる富士製作所で稼働する基幹データベース群を、Oracle Database 12cのマルチテナント機能を用いてOracle Database Appliance上に集約。

SAPのさらなる高速化にIn-Memoryの活用も検討：

ライオンがSAP ERPや基幹系のDB基盤をOracle Exadata X5 & Oracle Database 12cに更改　性能が4～20倍向上

ライオンは先頃、Oracle Exadata V2で運用してきたSAP ERPや基幹系のデータベース基盤をX5に移行。併せてRDBMSもマルチテナントを活用したOracle Database 12cにアップグレードし、処理性能が4～20倍向上した。同社は今後、Oracle Database In-Memoryの活用も検討しているという。[プライベートクラ

出典：　www.atmarkit.co.jp/ait/special/at140606/oracledatabaseinsider.html

ORACLE

リファレンス
マニュアル・ドキュメント

- Oracle Database Concepts, 12*c* Release 2 (12.2)
  - Part VI Multitenant Architecture
    https://docs.oracle.com/database/122/CNCPT/multitenant-architecture.htm
- Oracle Database Administrator's Guide, 12*c* Release 2 (12.2)
  - 全般
    https://docs.oracle.com/database/122/ADMIN/toc.htm
- Oracle Database Security Guide, 12c Release 2 (12.2)
  - Configuring Privilege and Role Authorization
    https://docs.oracle.com/database/122/DBSEG/configuring-privilege-and-role-authorization.htm#DBSEG004

追加の情報源

ソーシャル・メディア等

https://twitter.com/OraclePDB

https://blogs.oracle.com/multitenant/

https://www.facebook.com/OracleDatabase

otn http://www.oracle.com/goto/multitenant
http://www.oracle.com/technetwork/jp/database/multitenant/overview/index.html(日本)

ホワイト・ペーパー

- Oracle Multitenant White Paper in 12.2
- ホワイト・ペーパー：Oracle Multitenant in 12.1(2013)
- White Paper: Oracle SuperCluster T5-8 Oracle Multitenant データベース統合の効率に関する事例
- White Paper: Database Live Migration with Oracle Multitenant
- White Paper: Security Concepts in Oracle Multitenant
- MAA ベスト・プラクティス：データベース統合のための高可用性ベスト・プラクティス

12c R2 CoreTech Seminar

- セミナー資料

ビデオ

- Oracle Database 12c Multitenant Architecture Overview
- Consolidation with Oracle Database 12c
- Oracle Database 12c: Introduction to a Multitenant Environment with Tom Kyte

MOS

- **[マスターノート] Oracle マルチテナント オプション (Doc ID 2004241.1)**

ORACLE

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®