
ORACLE®

Oracle Database 12*c* Release 2 CoreTech Seminar

12.2.0.1
Multitenant

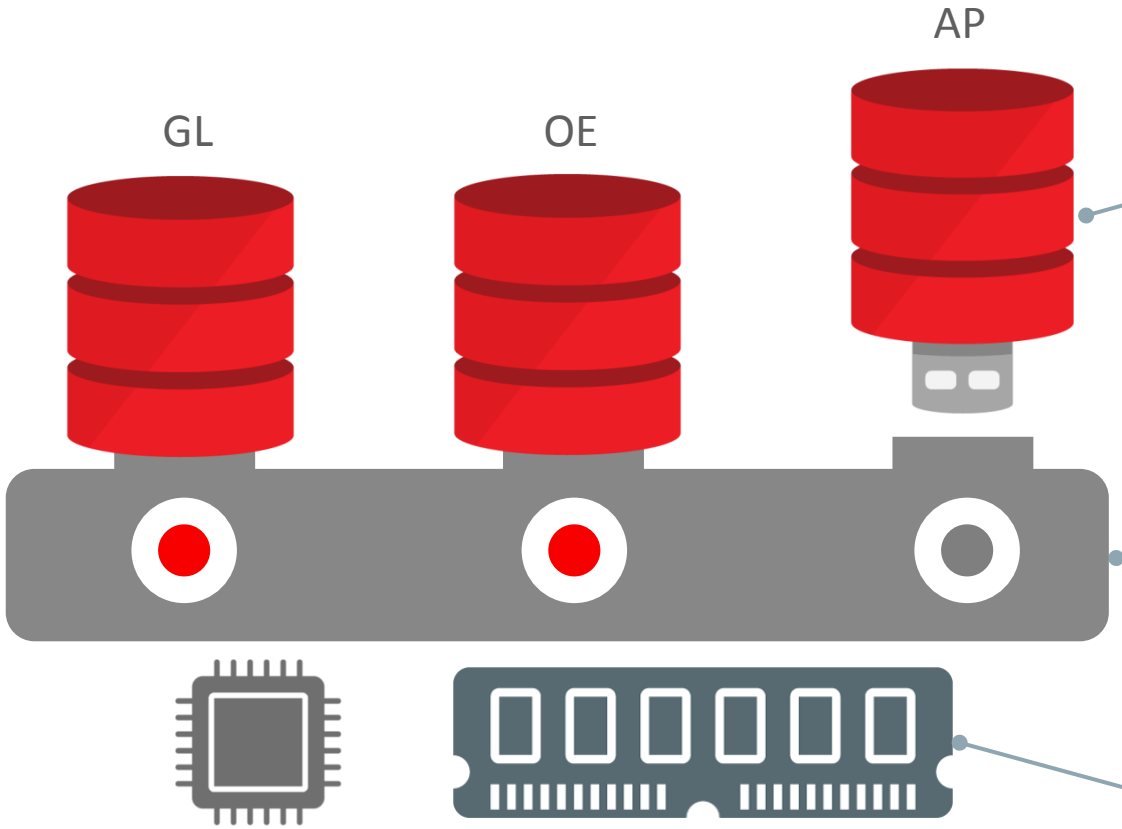
日本オラクル株式会社
クラウド・テクノロジー事業統括
Database & Exadataプロダクトマネジメント本部
伊藤 勝一
2016/10

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.

マルチテナント・アーキテクチャ

CapExとOpExを削減、アジリティの向上、導入・利用が容易



VMに比べて優れたリソース効率

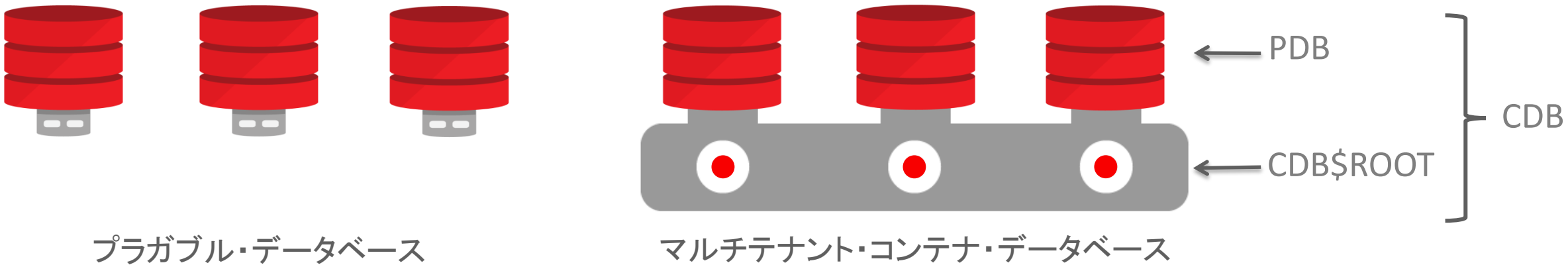
アプリケーションごとに自己完結したPDB
- アプリケーションの変更は不要
- 迅速なプロビジョニング (クローン)
- ポータビリティ (プラグ/アンプラグ)

CDB単位の共通オペレーション
- 一括管理
  (パッチ、アップグレード、HA構成、バックアップ)
- 個別の操作も可能

共有メモリーとバックグラウンド・プロセス
- サーバーあたりの集約密度の向上

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

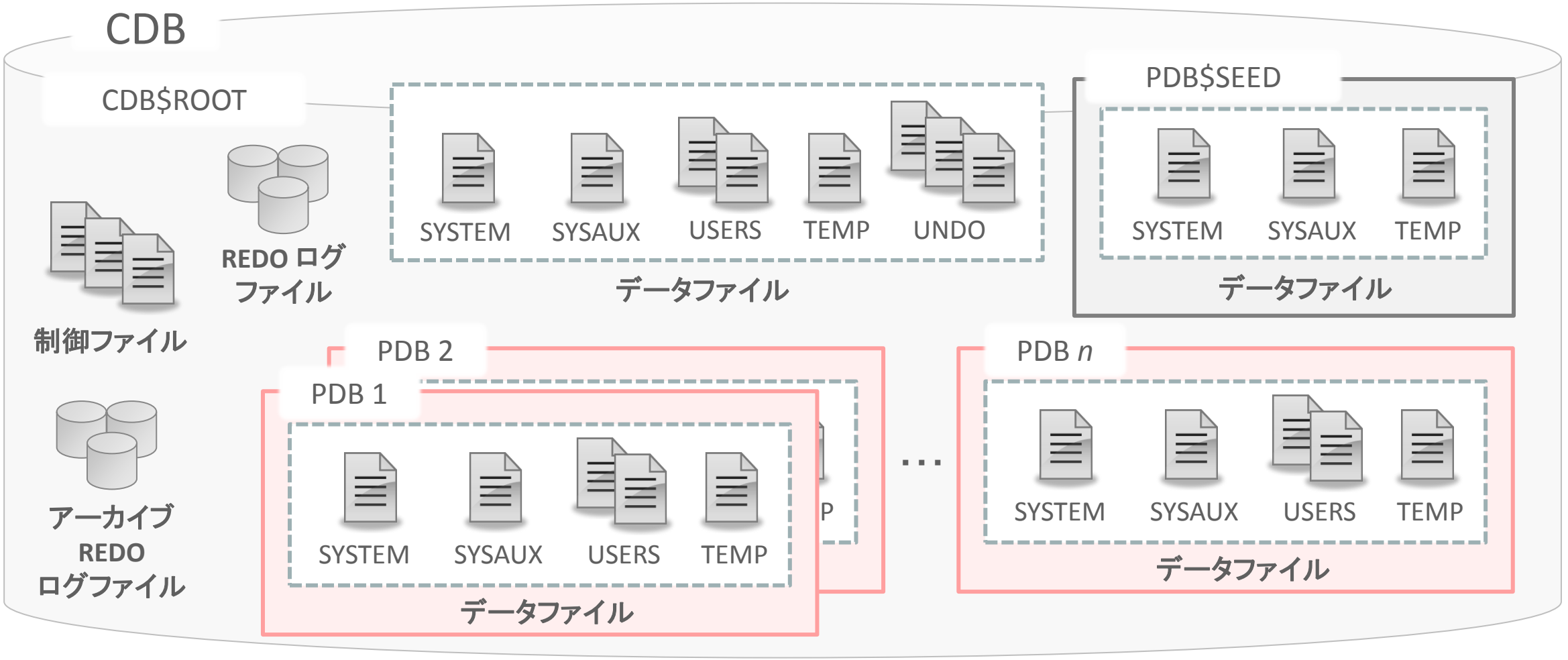


プラガブル・データベース

マルチテナント・コンテナ・データベース

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



Oracle Multitenantの主な利点

Benefit	Capability Enabled
CapEx(設備投資)の最小化	• サーバーあたりの集約密度の向上
OpEx(運用コスト)の最小化	• 一括管理 (パッチ適用回数減) • 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	• Dev & Testでのスナップショット・クローン • プラグ/アンプラグによる可搬性 • RACによるスケーラビィティ
容易	• 導入: アプリケーションの変更は不要 • 利用: SQLによる操作

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

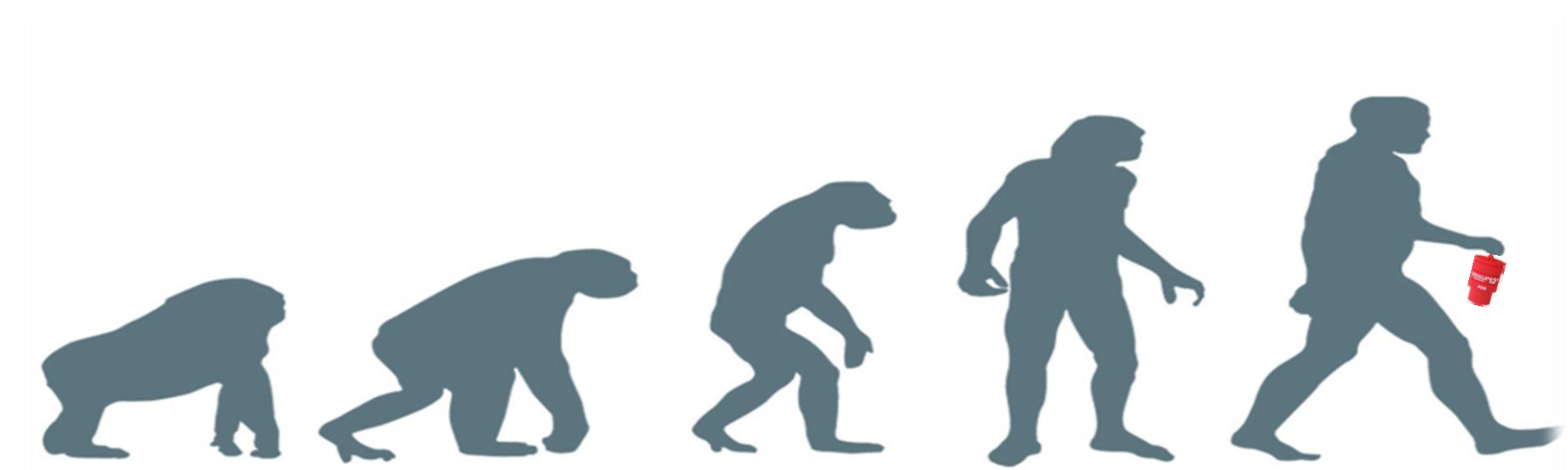
Agenda

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. アプリケーション・コンテナ
4. 位置透過性を実現する機能
5. まとめ

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



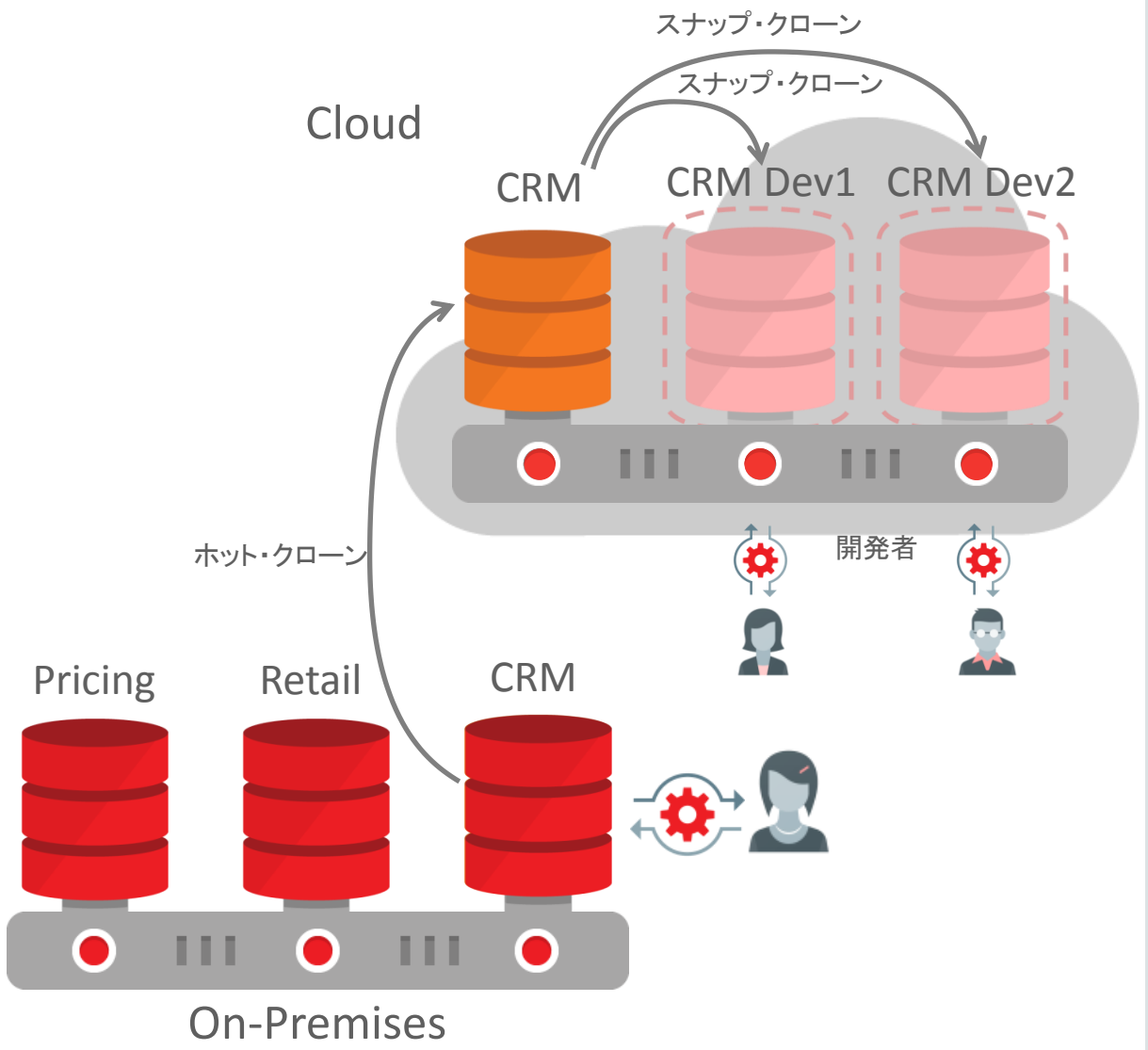
クローン元PDBが
読取り専用 –
コールド・クローン/
リモート・クローン

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

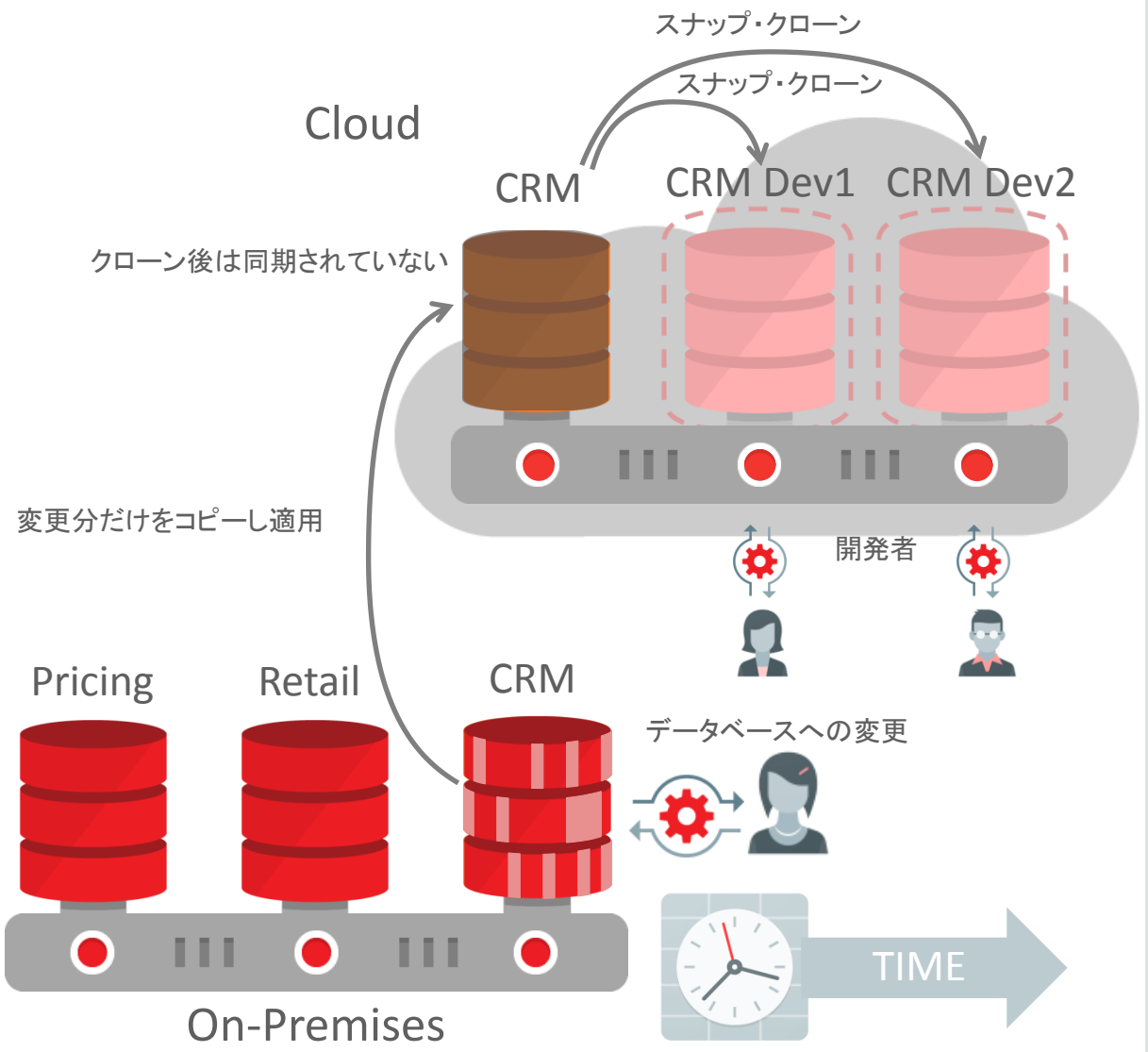
PDBホット・クローン

- PDBホット・クローン
  - オンラインでテスト・マスターを作成



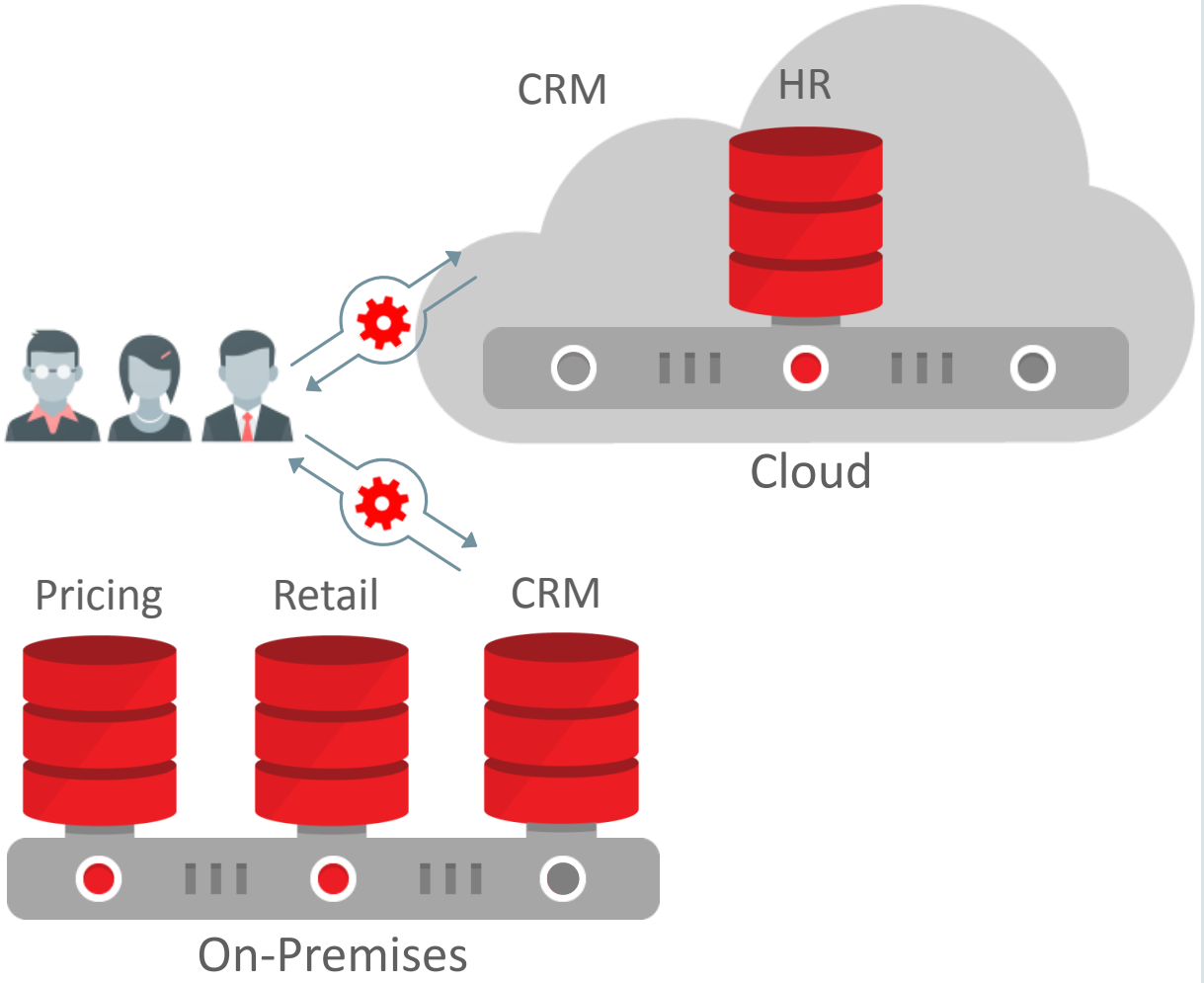
PDBリフレッシュ

- 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



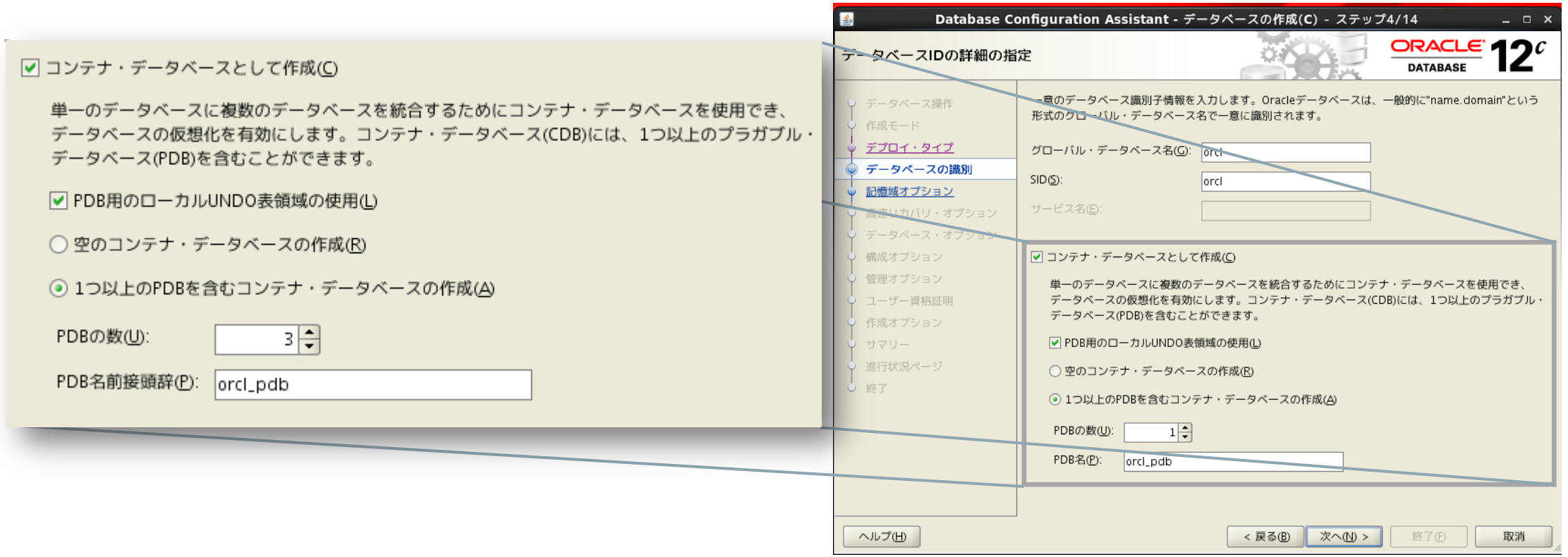
ローカルUNDO管理

マルチテナント環境でのUNDOモード

- 共有UNDO
  - CDB$ROOTのUNDO表領域が、プラグされているすべてのPDBで共用
  - 12.1での構成、12.2でも設定可能
- ローカルUNDO NEW IN 12.2
  - 12.2より追加されたモード
  - ホット・クローンなどオンライン・オペレーションを行う場合に必須
  - ローカルUNDOの構成はすべてのPDBに適用
    - 一部のPDBのみに適用することはできない
    - CDB$ROOTで構成され、CDB$ROOTの属性(Property)
    - PROPERTY_NAME = LOCAL_UNDO_ENABLED
  - UNDO表領域管理
    - Non-CDBと同様の複数のUNDO表領域、切り替え、オフライン化が可能

CDB作成時のUNDOモードの指定

- DBCAでCDBを作成時にUNDOモードを選択可能
  - 「**PDB用のローカルUNDO表領域の使用**」のオプションがデフォルトでチェック済み



CDBレベルでのUNDOモード設定変更

- 共有UNDOモードへの設定変更
  - CDB$ROOTにSYSユーザーで接続し、UNDOモードの切り替え

```
SQL> startup upgrade
SQL> alter database local undo off;
```

  - 上記を実行後、再起動
- ローカルUNDO
  - CDB$ROOTにSYSユーザーで接続し、UNDOモードの切り替え

```
SQL> startup upgrade
SQL> alter database local undo on;
```

  - 上記を実行後、再起動

UNDOモードの移行

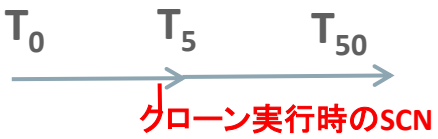
- 共有UNDOからローカルUNDOへの移行
  - UNDO表領域の自動作成
    - ローカルUNDOを持たないPDBを、ローカルUNDOとして構成されているCDBにプラグインした場合や、リモートPDBとしてクローンした時にUNDO表領域が自動で作成される
- ローカルUNDOから共有UNDOへの移行
  - ローカルUNDOモードで稼働していたPDBは、共有UNDOモードのCDBにプラグインされると共有UNDOが使用されるため、ローカルUNDO表領域は削除可能
  - ローカルUNDOで稼働していたかの確認
    - アンプラグ時に生成するXMLメタデータ・ファイルのエントリから確認
      ローカルUNDOの場合、<localundo>1</localundo>

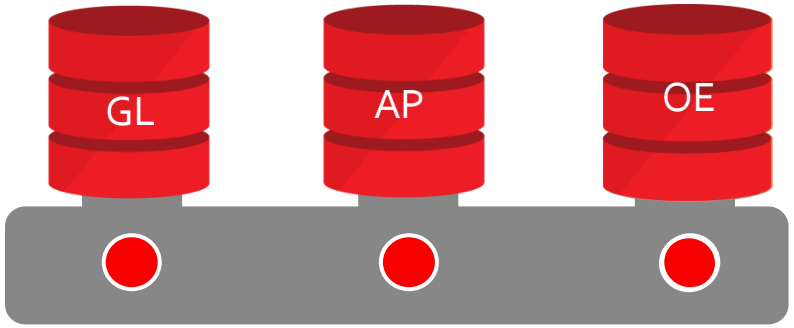
```
$ grep localundo cdb122.xml
   <localundo>1</localundo>
```

ホット・クローン

PDBコールド・クローン

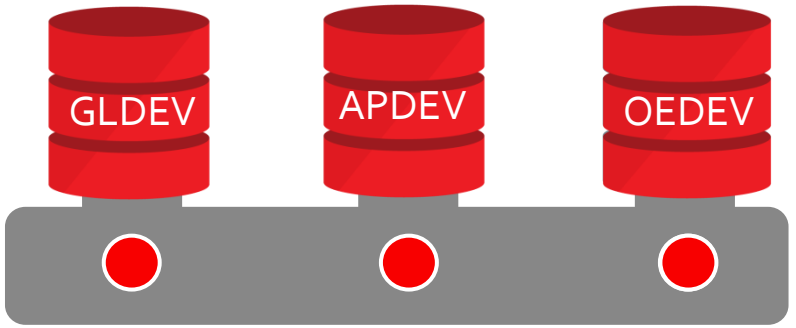
PRODUCTION





T_5 1. *alter pluggable database oe close;*
2. *alter pluggable database oe open read only;*

DEVELOPMENT



3. *create pluggable database oedev from oe@dblink;*

データファイルのコピー

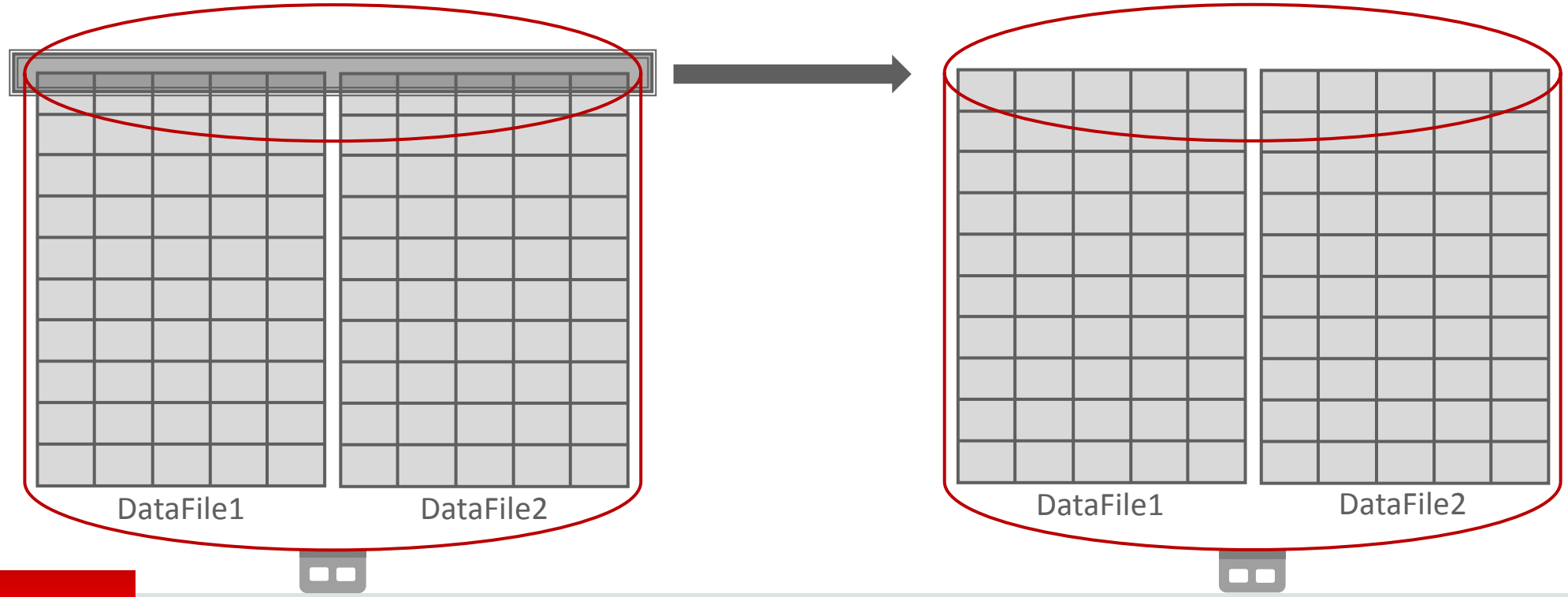
T_5 4. *alter pluggable database oe open read write force;*

T_5 4. *alter pluggable database oedev open;*

PDBコールド・クローン

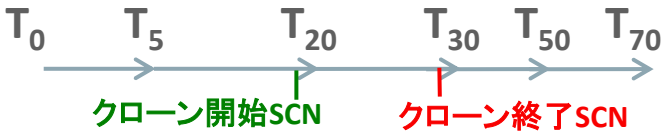
クローン元のPDBを読み取り専用に変更 (12c R1での実装)

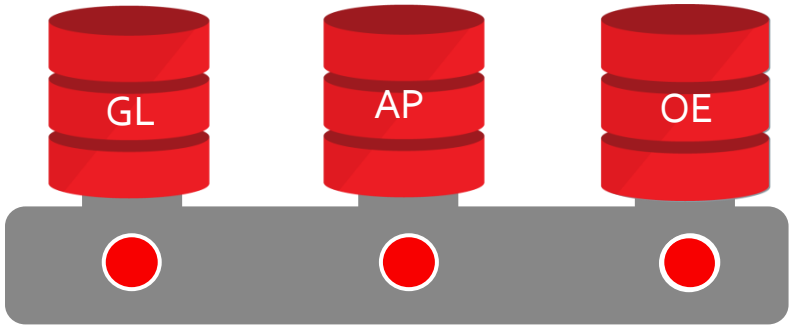
- クローン元となるPDBは読取り専用(Read Only)に変更
- リードとコピーは並列実行
- クローン完了後にクローン元のPDBを読取り/書込み(Read Write)でオープン



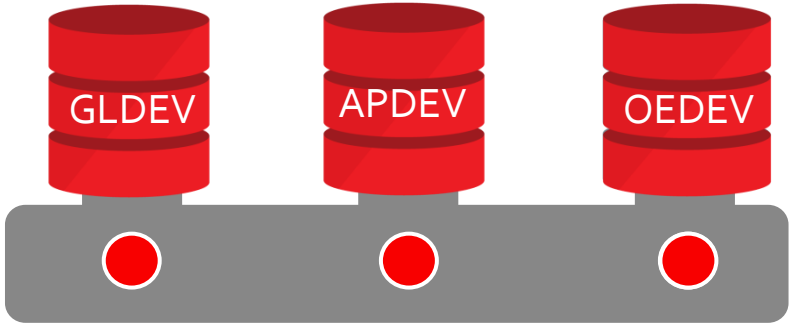
PDBホット・クローン

PRODUCTION





DEVELOPMENT

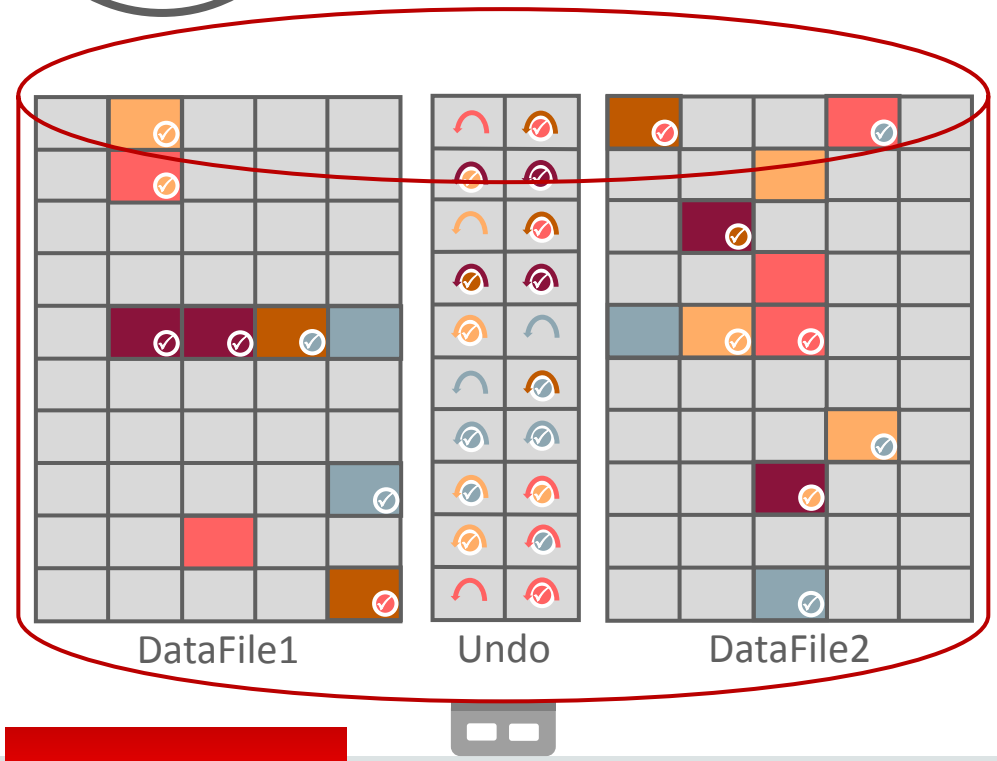


T_{20} 1. *create pluggable database oedev from oe@dblink;*

REDOの最終コピーとロールバック T_{30}

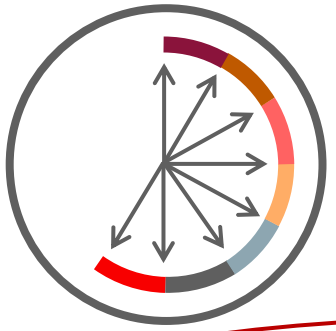
T_{30} 2. *alter pluggable database oedev open;*

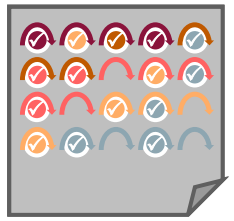
時間の経過とデータベースへの変更のモデル化



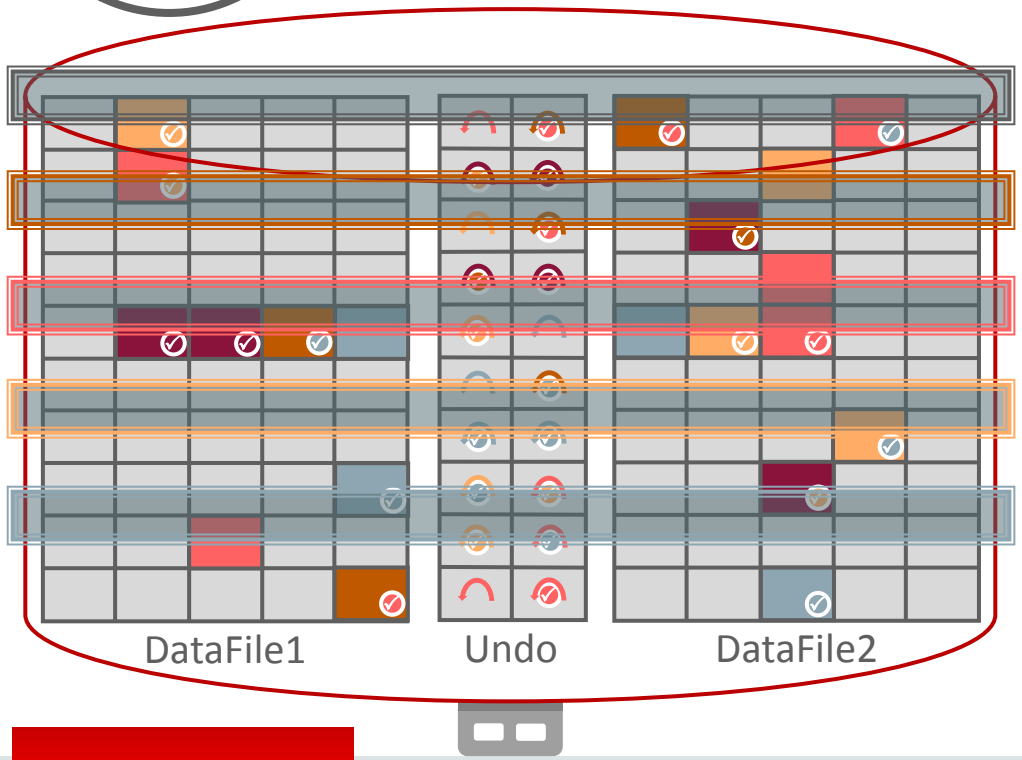
	凡例
	インターバル中に変更されて 未コミットのブロック
	インターバル中に変更されて コミットされたブロック
	未コミットのREDO
	インターバル中に書き込まれ、 コミットされたUNDO

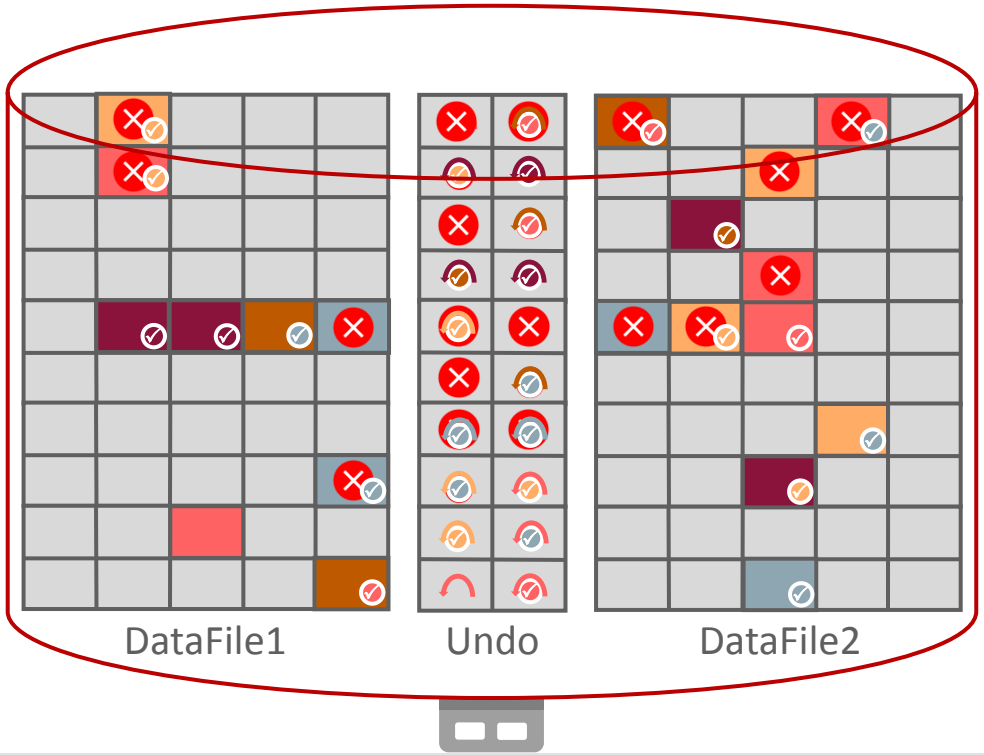
PDBホット・クローン





- クローン元PDBは読取り/書込み(Read Write) のまま
- リードとコピーは並列実行
- 実行中の操作は“Dirty Read”となる
- いくつかのデータ変更は、最初のファイル・コピーには含まれない
- クローン元に追従するためREDOの転送と適用を実施
- UNDOの適用、未コミットのトランザクションのロールバック





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

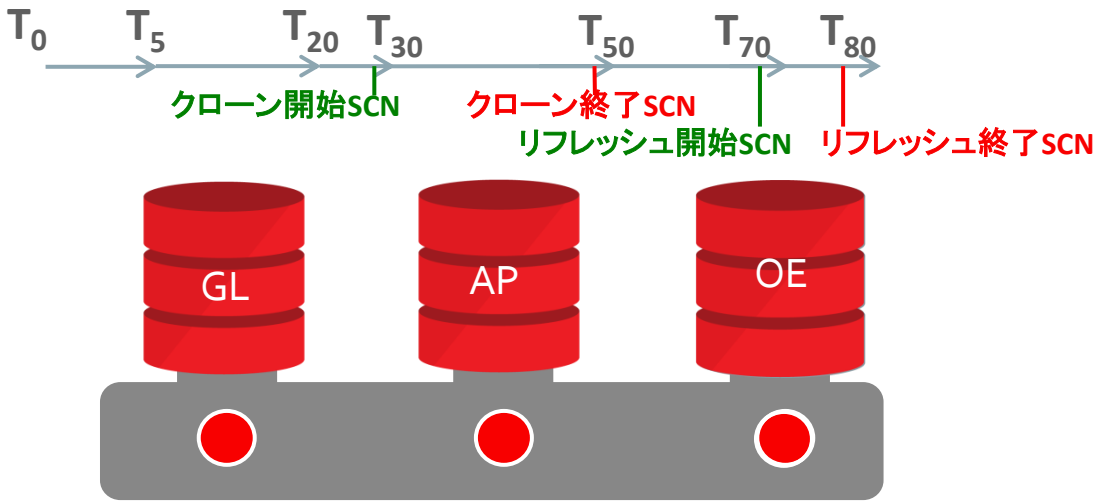
- 構成
  - 同じCDB上でもPDBホット・クローンが可能
  - 異なるCDBへクローンを行う場合、データベース・リンクを使用
    - 新しくPDBが作成されるターゲット側のCDBから、クローン元であるソース側のCDBまたはクローン対象のPDBに対してデータベース・リンクを作成
  - 同じエンディアンであれば、異なるプラットフォームでもPDBホット・クローン可能
    - Windows (x86_64) => Linux (x86_64)

    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

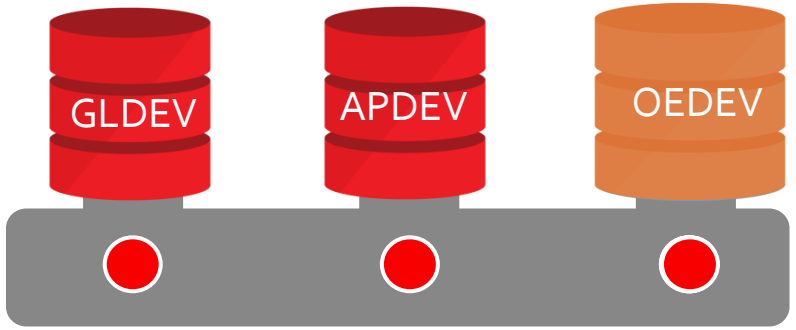
PDBリフレッシュ

PDBリフレッシュ – 手動モード

PRODUCTION



DEVELOPMENT



1. *create pluggable database oedev from oe@dblink* ***refresh mode manual****;*

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

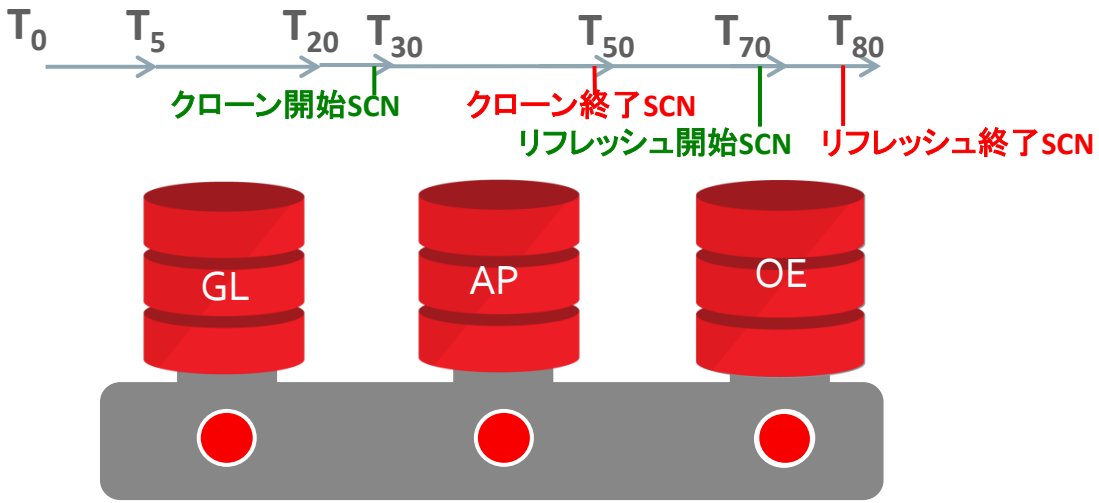
T_{50} 2. *alter pluggable database oedev open read only;*

3. *alter pluggable database oedev refresh;*

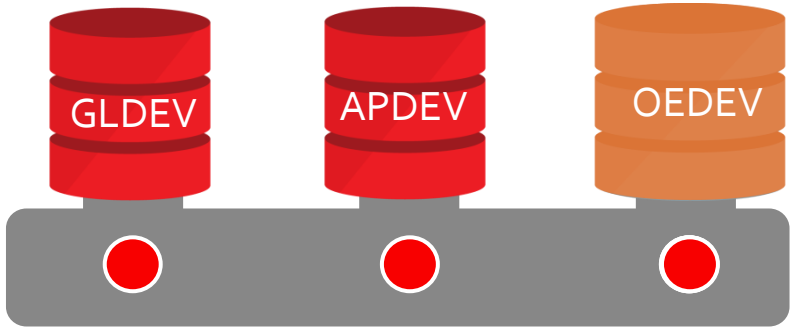
T_{80} 4. *alter pluggable database oedev open read only;*

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

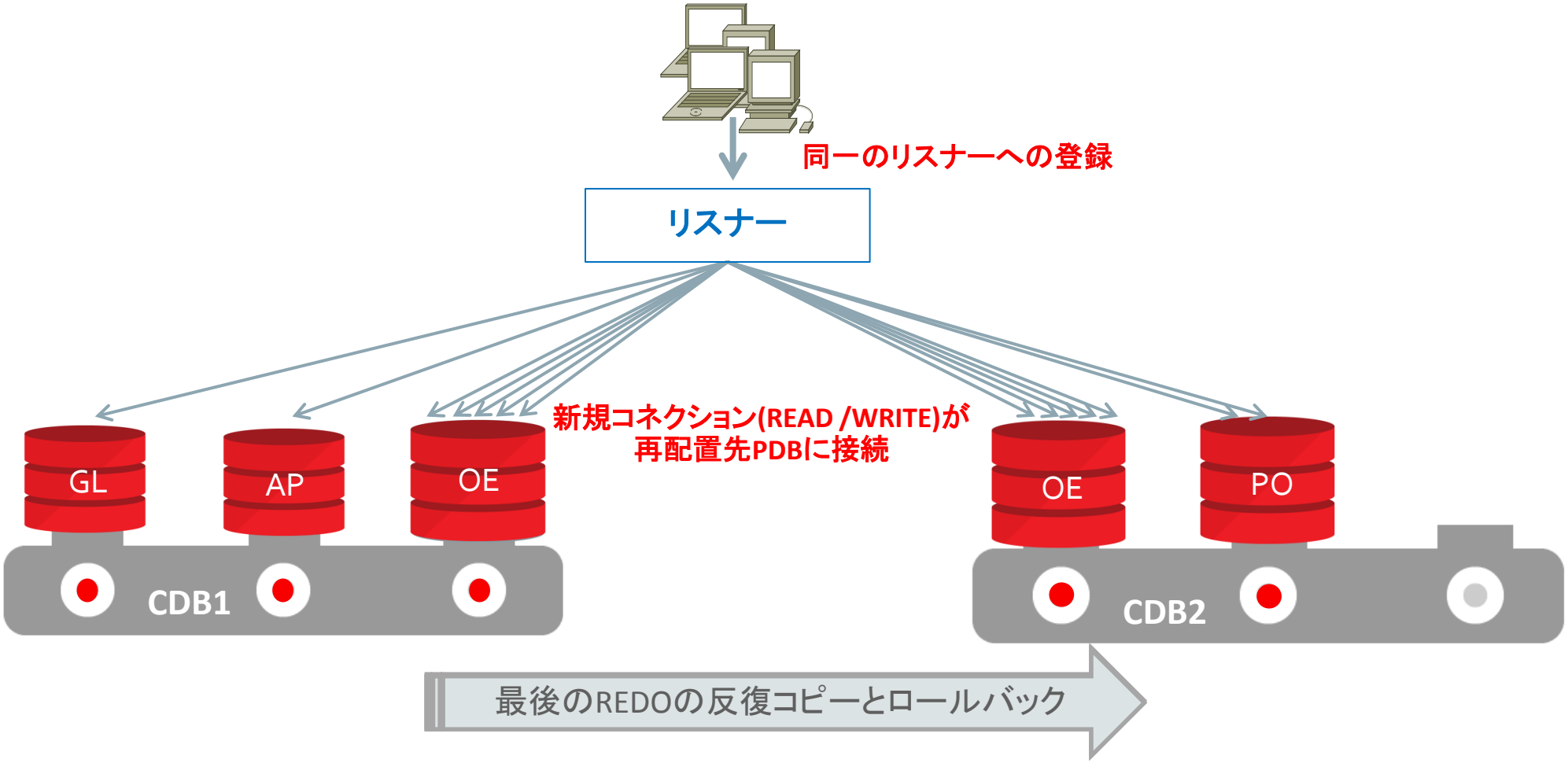
- 優位性
  - 継続的にクローン元のPDBでのアプリケーションの稼働を可能とする
  - クローン元データベースへの影響を最小化
  - RDBMSに統合
    - 3rdパーティのソフトウェアは不要
  - アプリケーション開発とリリースまでの時間を短縮
  - データベースのプロビジョニング・コストを削減
  - クローン元PDBとの差分リフレッシュによる軽い処理
  - 時間粒度の細かいクローニング
    - 2つのモード - 手動と自動
    - Oracleのスケジューラー・ジョブとして事前定義

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)

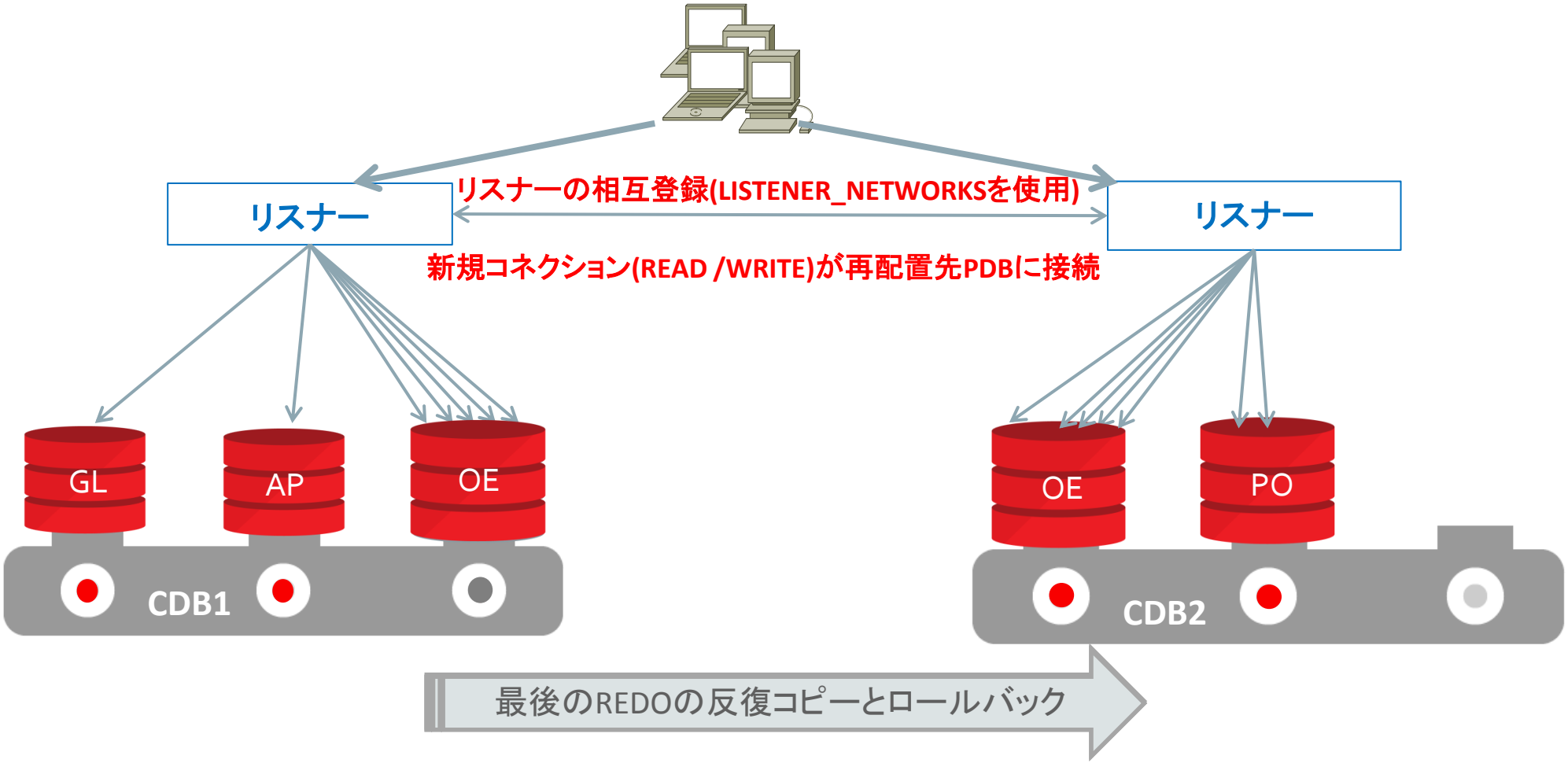
PDB再配置:
リスナーを共有するケース



create pluggable database OE from OE@CDB1_dblink relocate;

alter pluggable database OE open;

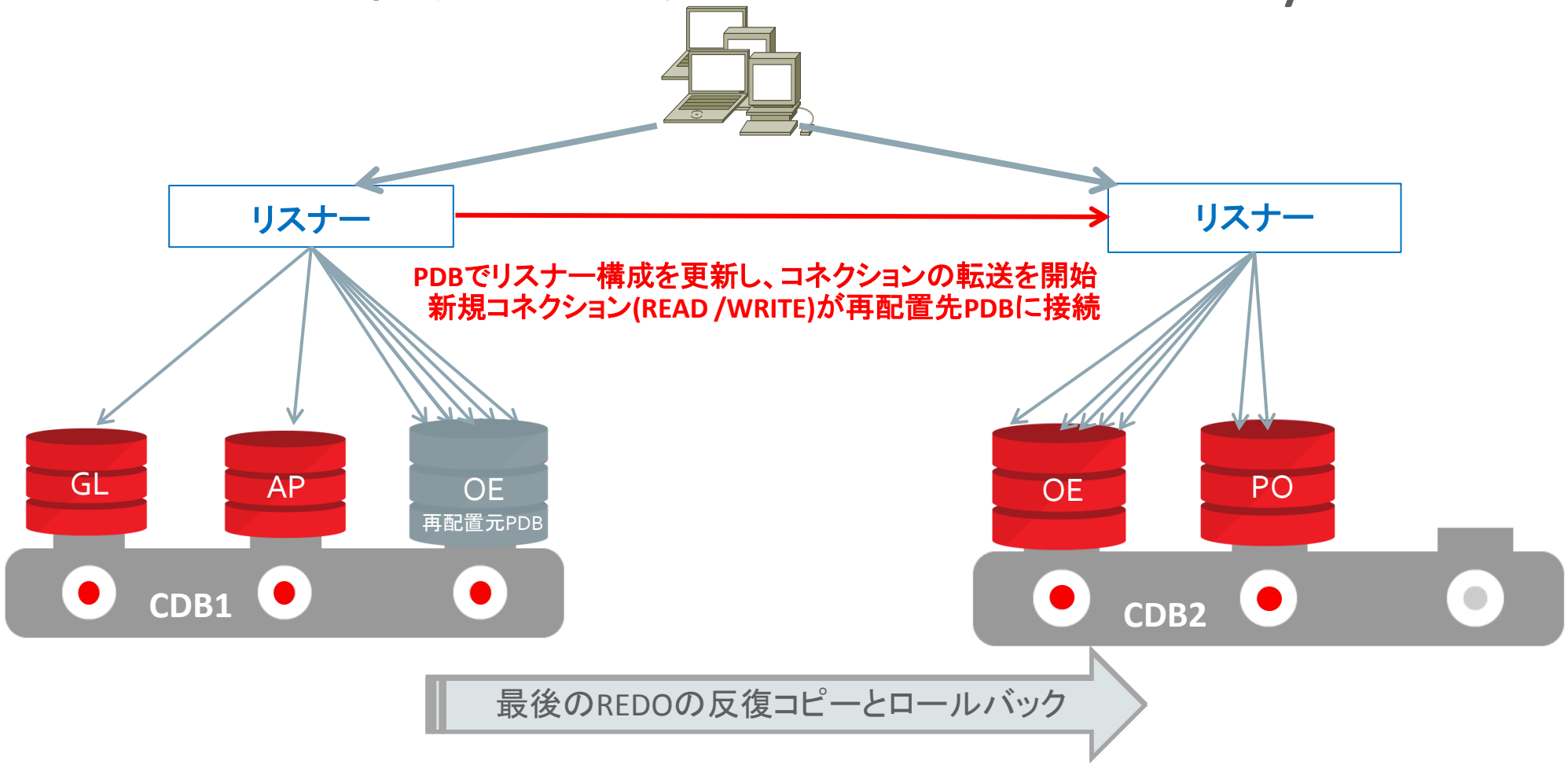
PDB再配置:
相互のリスナーに登録しているケース



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再配置の基盤となるテクノロジ

ホット・クローン
- ローカルUNDO、アーカイブ・ログ・モードでの運用

共通ユーザー
- リモートPDBのクローニング用のデータベース・リンク

増分REDO適用

再配置対象のPDBの静止
- クライアント・セッションのはり替え、切断

リスナーによる接続要求の転送
- SCANテクノロジーの拡張

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再配置時のコネクションのオンライン転送

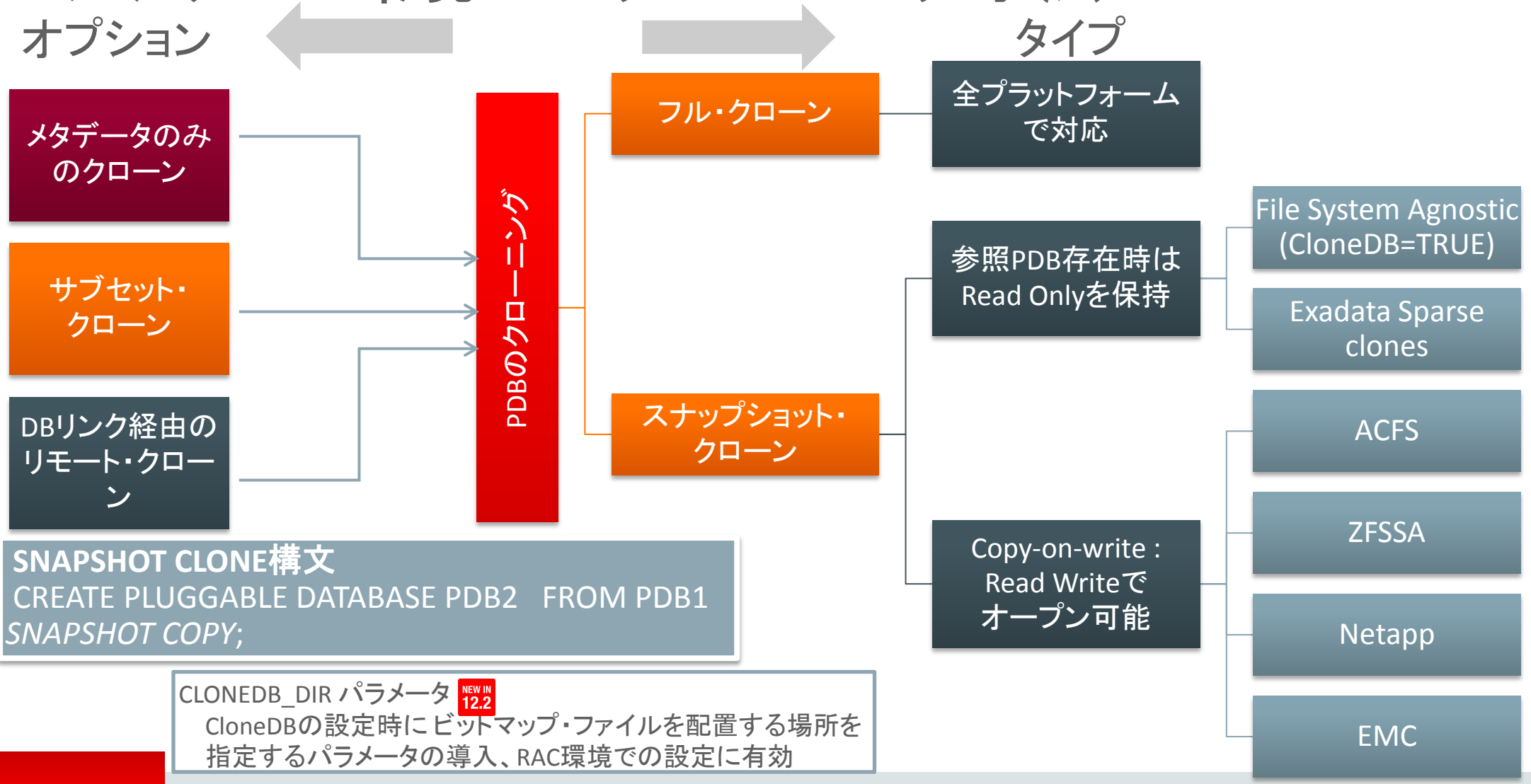
- 通常時のリスナーの状態 (lsnrctl serviceの結果出力)

```
サービス"soe"には、1件のインスタンスがあります。
 インスタンス"cdb0011"、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:15 拒否:0 状態:ready
     LOCAL SERVER
```

- PDB再配置実行時(Availability Maxを指定)

```
サービス"soe"には、1件のインスタンスがあります。
  インスタンス"cdb0011"、状態READYには、このサービスに対する2件のハンドラがあります...
    ハンドラ:
      "D000" 確立:0 拒否:0 現行:0 最大:1022 状態:ready
         DISPATCHER <machine: dbserver0011.jp.oracle.com, pid: 32292>
         (ADDRESS=(PROTOCOL=tcp)(HOST=dbserver0011.jp.oracle.com)(PORT=34309))
      "COMMON" 確立:0 拒否:0 状態:ready
         FORWARD SERVER
         (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dbserver0012.jp.oracle.com)(PORT=1521)))
```

マルチテナント環境でのクローニング手法



SNAPSHOT CLONE構文

CREATE PLUGGABLE DATABASE PDB2 FROM PDB1 *SNAPSHOT COPY*;

CLONEDB_DIR パラメータ NEW IN 12.2

CloneDBの設定時にビットマップ・ファイルを配置する場所を指定するパラメータの導入、RAC環境での設定に有効

PDB再配置 + アプリケーション・コンティニュイティ

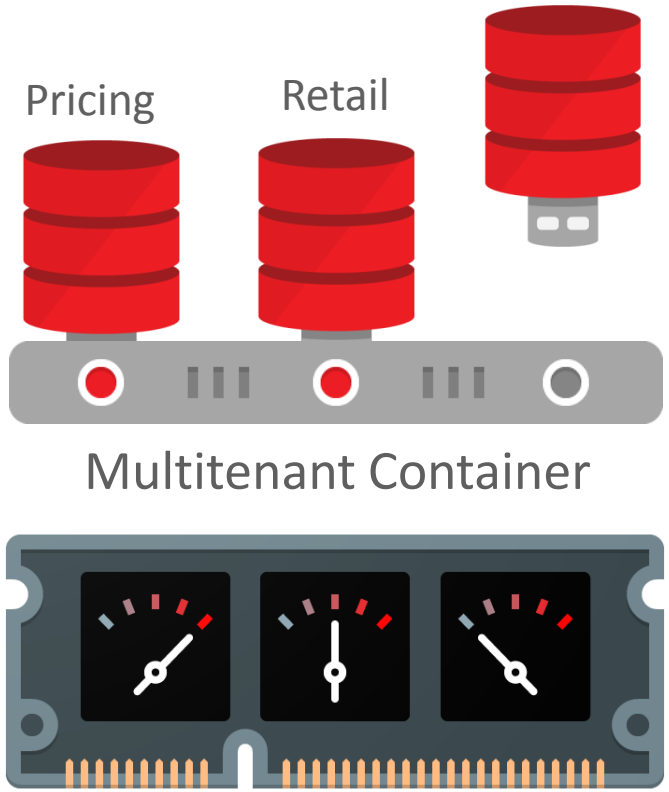
ゼロ・ダウンタイムでPDBを移動

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



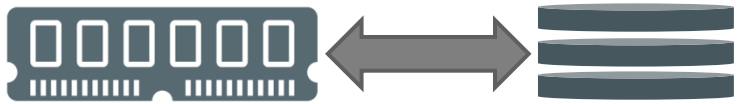
リソース制御

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

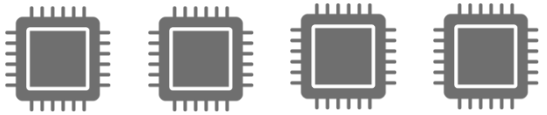
メモリー管理

- 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

PDB単位のCPUリソース管理

制限の強制

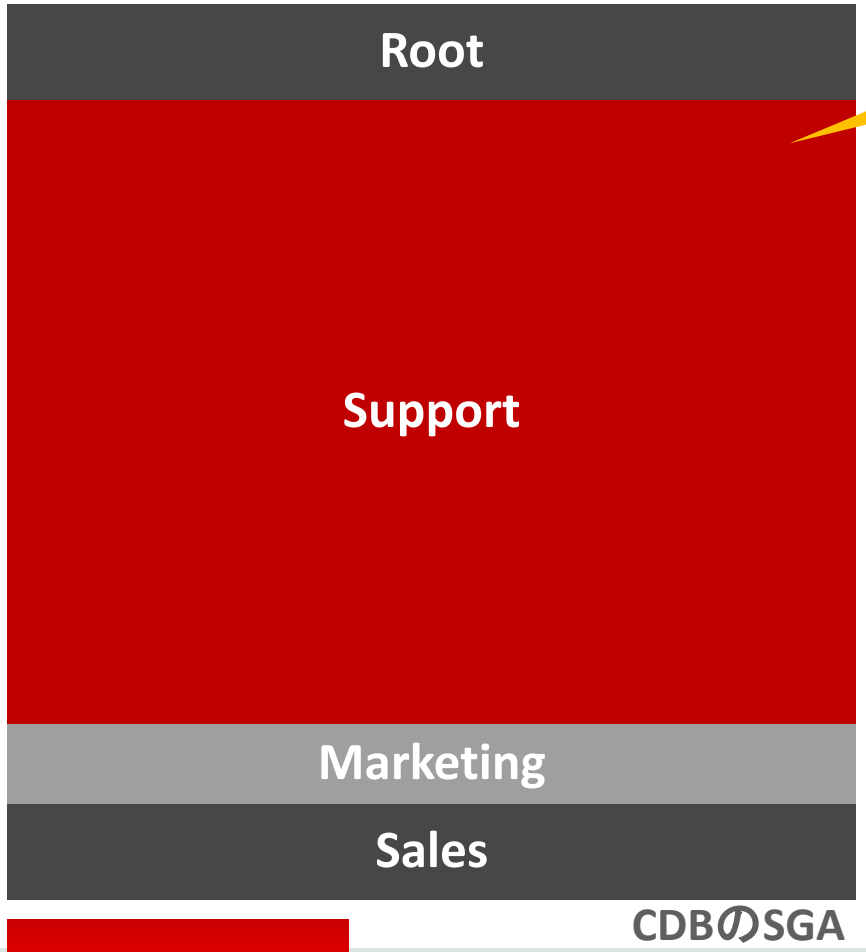
PDBごとにCPU_COUNTパラメータを設定

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

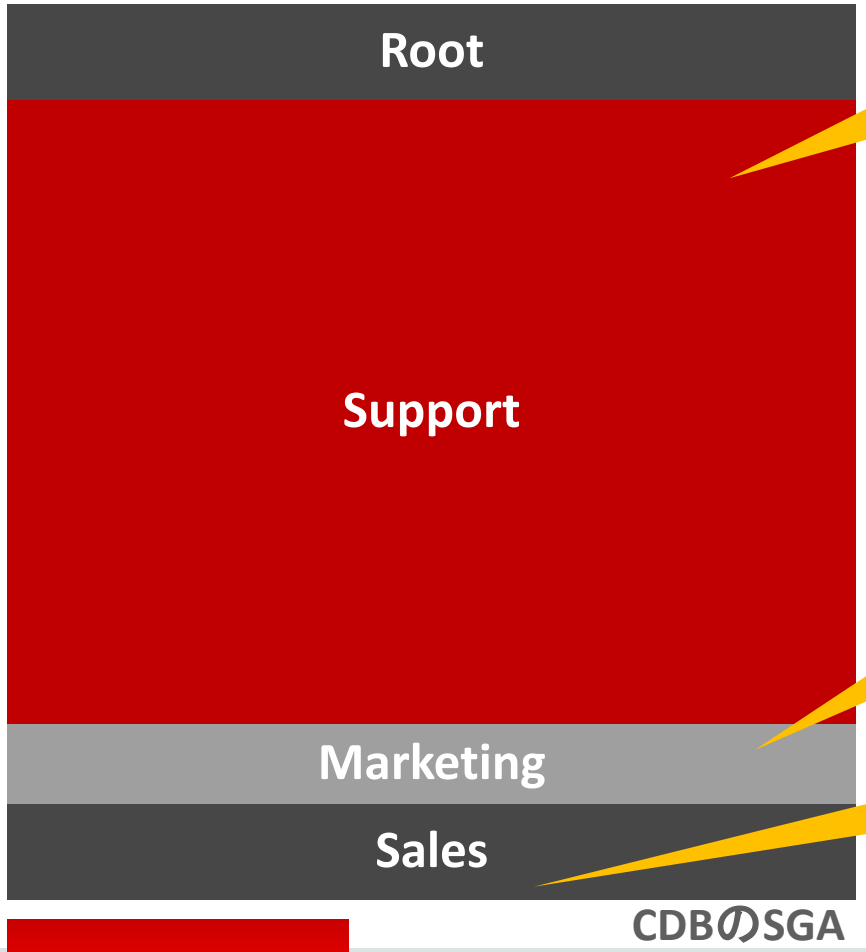
PDBのSGA管理



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

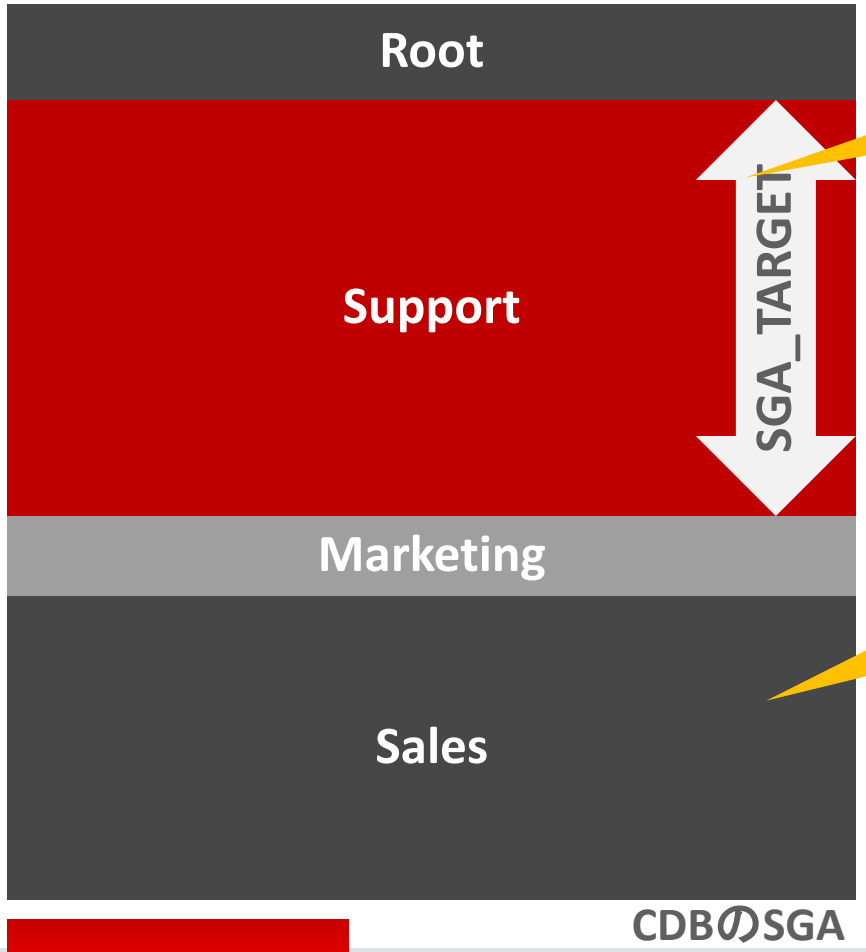


Support PDBはメモリーを良く使うワークロードを実行しており、SGAの大半を占有

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

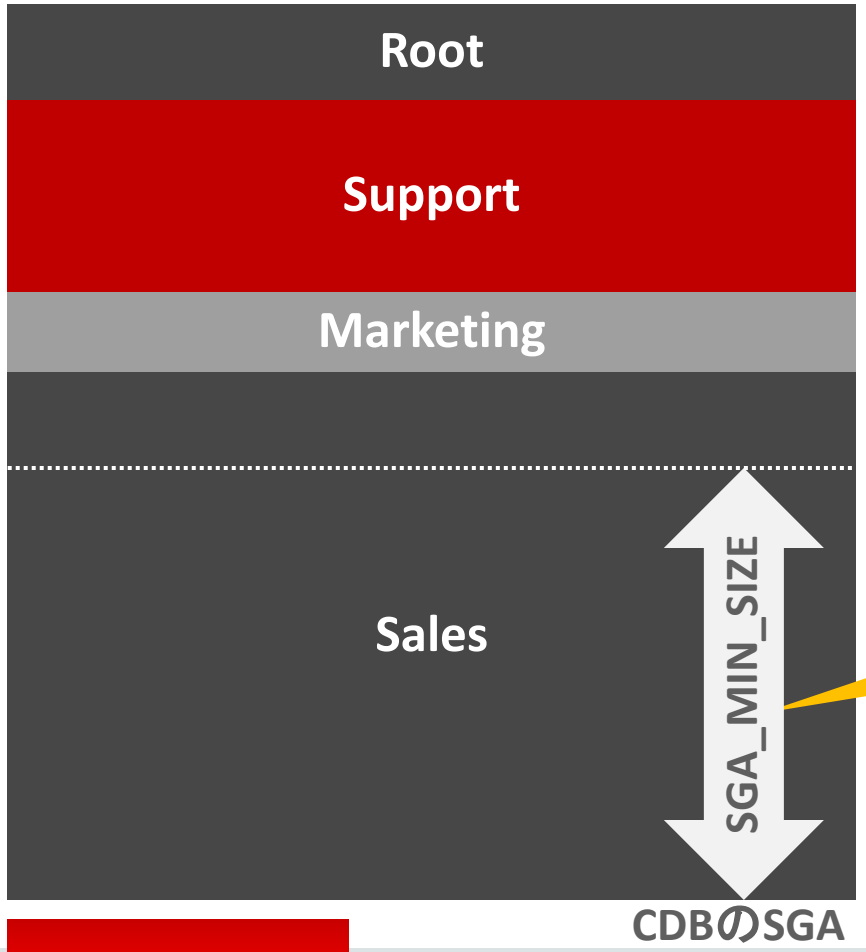
PDBのSGA管理



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

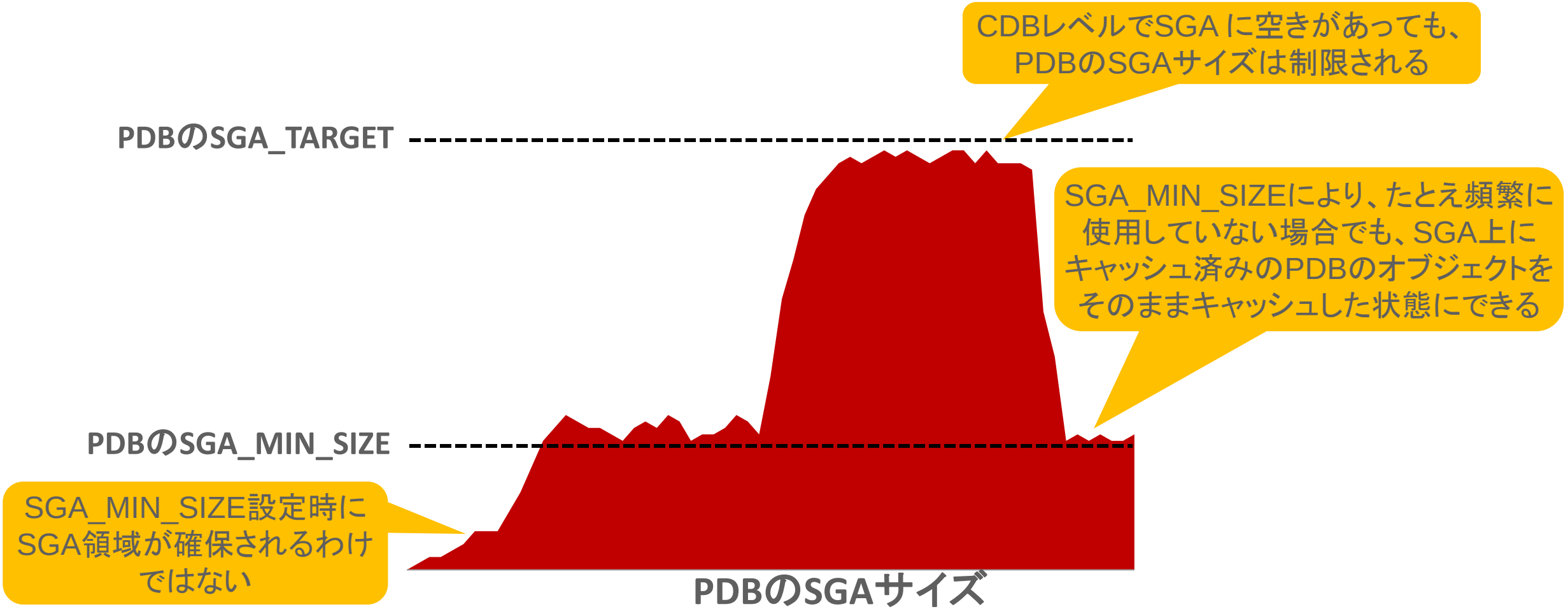
特定のPDBのSGAを制限すること
で、他のPDBによりSGAを提供!

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**

PDBのSGA管理



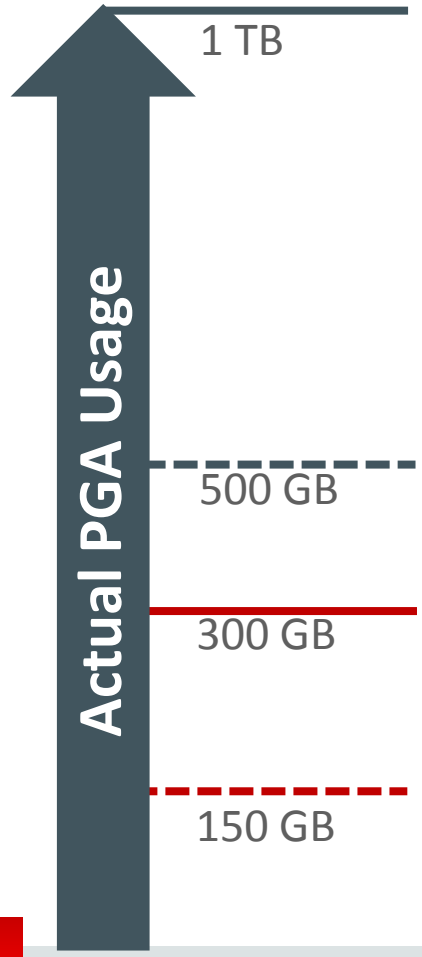
PDB単位のメモリー管理

PDBごとにSGAのサイズを制御

Parameter	Type	Description
SGA_MIN_SIZE	最低限確保するサイズ	PDBに割り当てが保証されるSGAのサイズ
DB_CACHE_SIZE	最低限確保するサイズ	PDBに割り当てが保証されるバッファ・キャッシュのサイズ
SHARED_POOL_SIZE	最低限確保するサイズ	PDBに割り当てが保証される共有プールのサイズ
SGA_TARGET	利用可能な上限のサイズ	PDBに割り当てられる最大のSGAサイズ

PDB単位のメモリー管理のパラメータは指定したサイズの確保と制限が行える
開発環境とクラウド統合環境の両方で重要

PDBのPGA管理



CDBのPGA_AGGREGATE_LIMIT
これ以上のPGA領域は割り当てられない
PGA領域を一番使用しているコールもしくはセッションが終了

CDBのPGA_AGGREGATE_TARGET
ここからすべてのセッションがPGAの代わりに一時表領域を使用する

PDB のPGA_AGGREGATE_LIMIT
PDBレベルでPGA_AGGREGATE_LIMITが設定

PDB のPGA_AGGREGATE_TARGET
PDBレベルでPGA_AGGREGATE_TARGETが設定

PDB単位のメモリー管理

PDBごとにPGAのサイズを制御

Parameter	Description
PGA_AGGREGATE_LIMIT	PDBが利用できるPGAの最大サイズ
PGA_AGGREGATE_TARGET	PDBが利用するPGAサイズの目標サイズ

- 部門レベルの統合環境
  - PDBがPGAを不均衡に利用している場合に設定を検討
- クラウド統合環境
  - 両方のPGAパラメータの設定を推奨
  - PGA_AGGREGATE_LIMIT = PGA_AGGREGATE_TARGET x 2 (デフォルト設定)

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)
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

PDBごとのメモリー管理 パラメータ設定時の留意点(1)

- MEMORY_TARGETをCDBで設定しない
- PDBで次のパラメータ指定時はCDBでSGA_TARGETの指定が必要
  - SGA_TARGET
  - SGA_MIN_SIZE
- SGA_MIN_SIZEにはSGA_TARGET の設定値の**50%以下**の値を設定
- PDBで次のパラメータ指定時はCDBでの設定値の**50%以下**の値を設定
  - SGA_MIN_SIZE
  - DB_CACHE_SIZE
  - SHARED_POOL_SIZE
    - CDBでSGA_TARGETを指定時、その値の50%をPDBで指定するDB_CACHE_SIZE+SHARED_POOL_SIZEの値が超えてはいけない

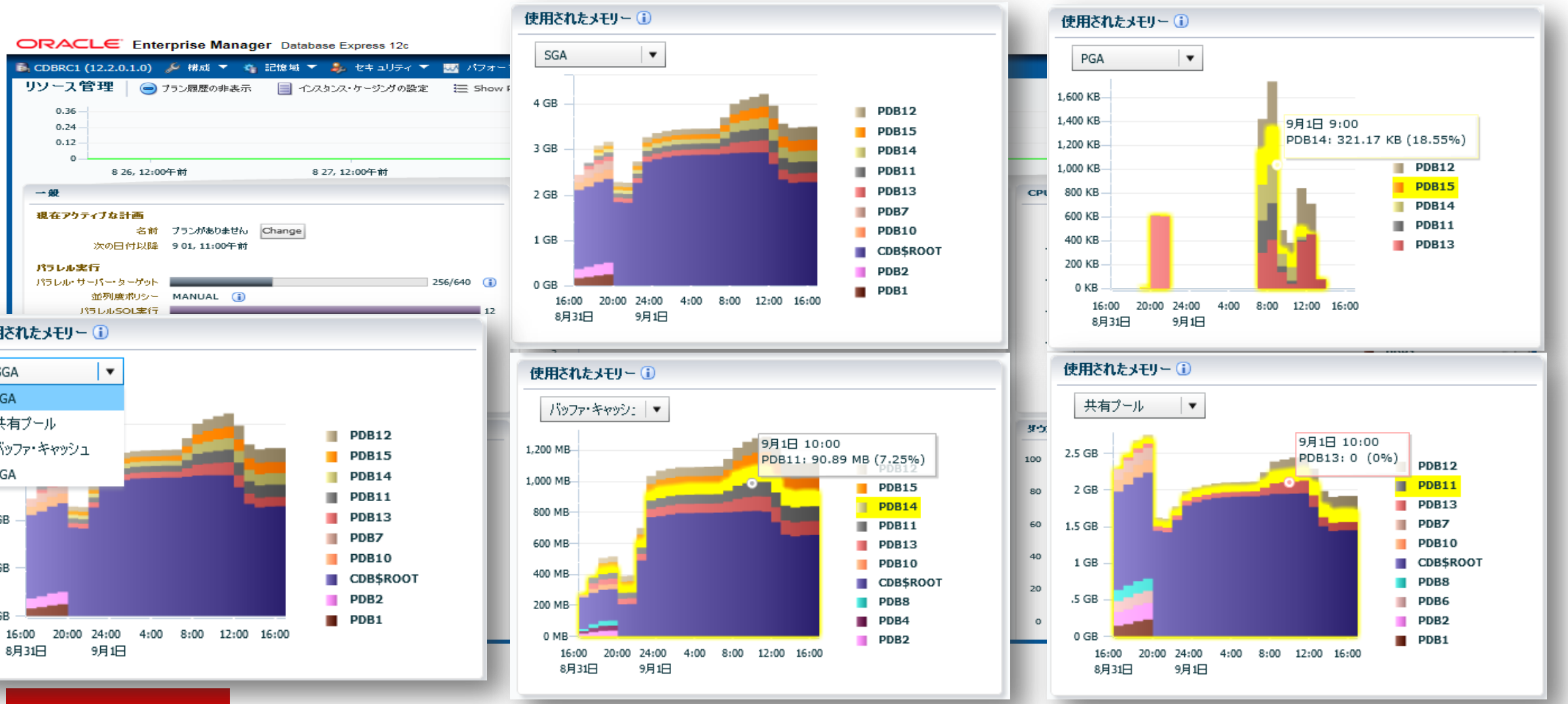
```
SQL> alter system set sga_min_size=800M;
alter system set sga_min_size=800M
*
行1でエラーが発生しました。:
ORA-32017: SPFILEの更新中に障害が発生しました
ORA-56746:
無効な値838860800 (パラメータsga_min_size); 50%
(パラメータsga_target)よりも小さくする必要があります
```

PDBごとのメモリー管理 パラメータ設定時の留意点(2)

- 次のパラメータは**全PDBの設定値の合計値**がCDBのSGA_TARGETの設定値の**50%以下**の値を設定
  - **SGA_MIN_SIZE**の全PDBの設定値の合計
    - 例: PDBが10個、CDBでSGA_TARGETを10GBに指定した場合
      - 全てのPDBでSGA_MIN_SIZEは500MBまでは設定可能
      - あるPDBで3GB、他のPDBで2GB、他は指定なし
  - **DB_CACHE_SIZE+SHARED_POOL_SIZE**の全PDBの設定値の合計
- 上記を超えて設定した場合、内部的に全PDBの設定値が調整される
  - アラートにメッセージが出力

```
PDB6(8):ALTER SYSTEM SET sga_min_size=1G SCOPE=BOTH PDB='PDB6';
2016-09-07T19:49:48.748003+09:00
Reducing SGA_MIN_SIZE across all PDBs because the sum (6144MB) is too large a percentage of CDB's SGA_TARGET
```

Enterprise Manager Express メモリー管理



PDB I/Oレート制限

目的

- Exadata以外のシステムにIOリソース・マネージャ(IORM)の代替となるものを提供
  - ストレージ・ソフトウェアとの緊密な統合なしにIORMをデータベースに実装できない
- ひとつの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

- Exadata IORMほど万能ではない

PDB I/Oレート制限

機能

- 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

PDB I/Oレート制限

機能

- ほとんどのPDBによるI/Oが制御される
  - バッファ・キャッシュへの読み込み (バッファ・キャッシュからのReadは制御しない)
  - ダイレクト・リードおよびダイレクト・ライト
  - 一時表領域のリード・ライト
- PDBによるI/Oでレートとして計測に含むが、制御しない
  - DBWR書込み
  - コントロール・ファイルとパスワード・ファイルのI/O
- PDBによるI/Oでレートとして計測せず、また制御もしない
  - LGWR I/Os
  - Root I/Os (ルート・コンテナによるI/O)
- 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

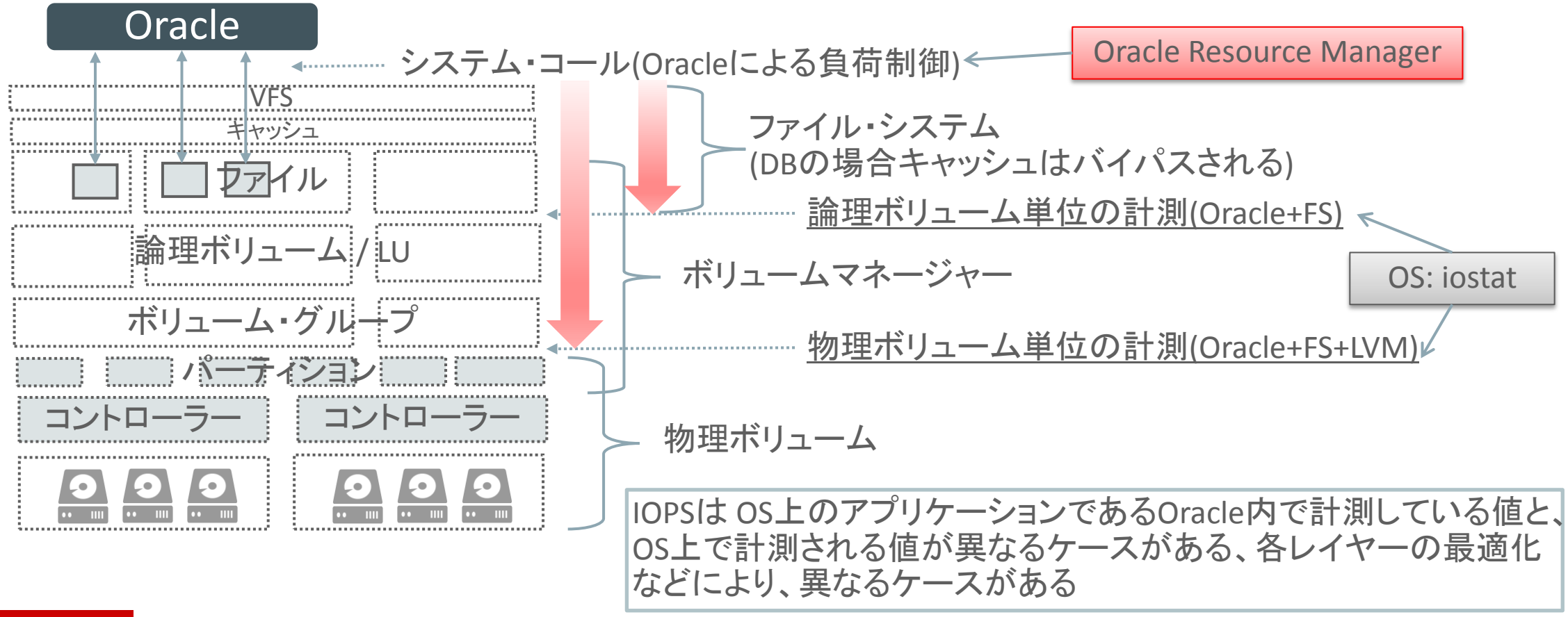
PDBレベルのリソース使用状況の確認

- 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

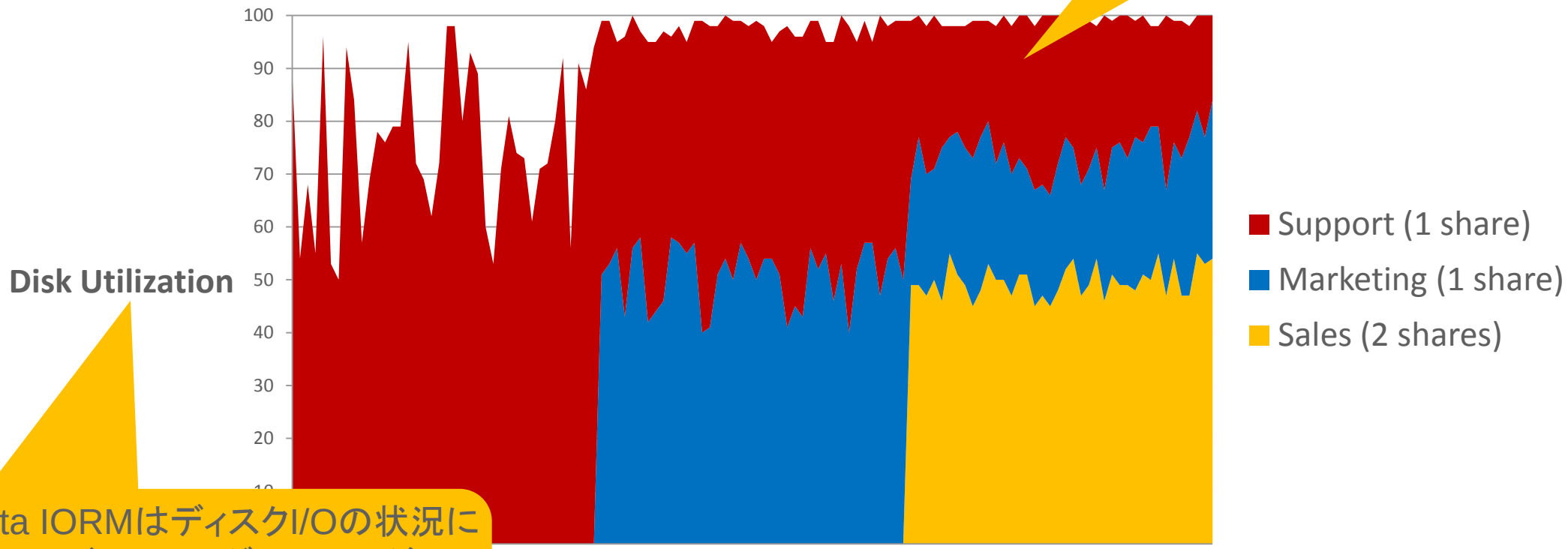
参考: IOPSの計測



参考: 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

セッション数の管理

PDB利用時のベスト・プラクティス

SESSIONSパラメータはCDBとPDBの両方で設定が可能

	CDB	PDB
SESSIONS	セッション数の最大値 全てのPDBで共有される	PDBで利用可能なセッション数の最大値
再帰セッション	含む	含まない
設定値の目安	SESSIONS = 接続数 x 1.1	SESSIONS = 接続数

クラウド環境では、アイドルセッションが多数となることが想定される
MAX_IDLE_TIMEパラメータの設定によりアイドル・セッションの自動切断が可能

パフォーマンス・プロファイル

実装背景

- 大規模なデータベース統合を行う場合、何百ものPDBのエントリを1つのCDBプランで管理することは困難
- 一般的にPDBを多数持つCDBでは、より少数の”プロファイル”としてPDBをタイプ付けできる
  - Gold、Silver、Bronze
  - Large、Medium、Small
- DB_PERFORMANCE_PROFILEパラメータの導入

CDB Resource Plan		
Database	**Shares**	**Utilization Limit**
GoldDB1	2	
GoldDB2	2	
…	2	
GoldDB100	2	
SilverDB1	1	75%
…	1	75%
SilverDB200	1	75%

パフォーマンス・プロファイル

使用方法

- ステップ1: CDBプランの中に指示子として、作成する:

CDB Resource Plan		
Profile	**Shares**	**Utilization Limit**
Gold	2	
Silver	1	75%

```
SQL> alter session set container = cdb$root;

SQL> begin
       dbms_resource_manager.create_pending_area;
       dbms_resource_manager.create_cdb_plan('daytime_plan');
       dbms_resource_manager.create_cdb_profile_directive(
          'daytime_plan', profile => 'gold', shares => 2);
       dbms_resource_manager.create_cdb_profile_directive(
          'daytime_plan', profile => 'silver', shares => 1, utilization_limit => 75);
       dbms_resource_manager.submit_pending_area;
       end;
       /
```

パフォーマンス・プロファイル
使用方法

CDB Resource Plan		
Profile	**Shares**	**Utilization Limit**
Gold	2	
Silver	1	75%

- ステップ2: PDBのパラメータファイルに“db_performance_profile”としてPDBプロファイル名を指定

```
SQL> alter session set container = pdb_1;

SQL> alter system set db_performance_profile = 'gold' scope=spfile;
```

- ステップ3: プロファイルが使用されているか確認

```
SQL> alter session set container = cdb$root;

SQL> select p.name, shares, utilization_limit, profile
       from v$rsrc_plan r, v$pdbs p where r.con_id = p.con_id;
```

PDB Parameter file

sga_target = 16G
pga_aggregate_target = 8G
db_performance_profile = gold

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
Shares	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

共有による潜在的リスク

Oracle Multitenant

- マルチテナントによる規模の経済性は、鍵となるインフラストラクチャおよびメモリーの共有によって生み出される
- テナント (PDB) はホストだけでなく、OS、ネットワーク、共通オブジェクトも共有する

共通アクセスの潜在的脆弱性への対応

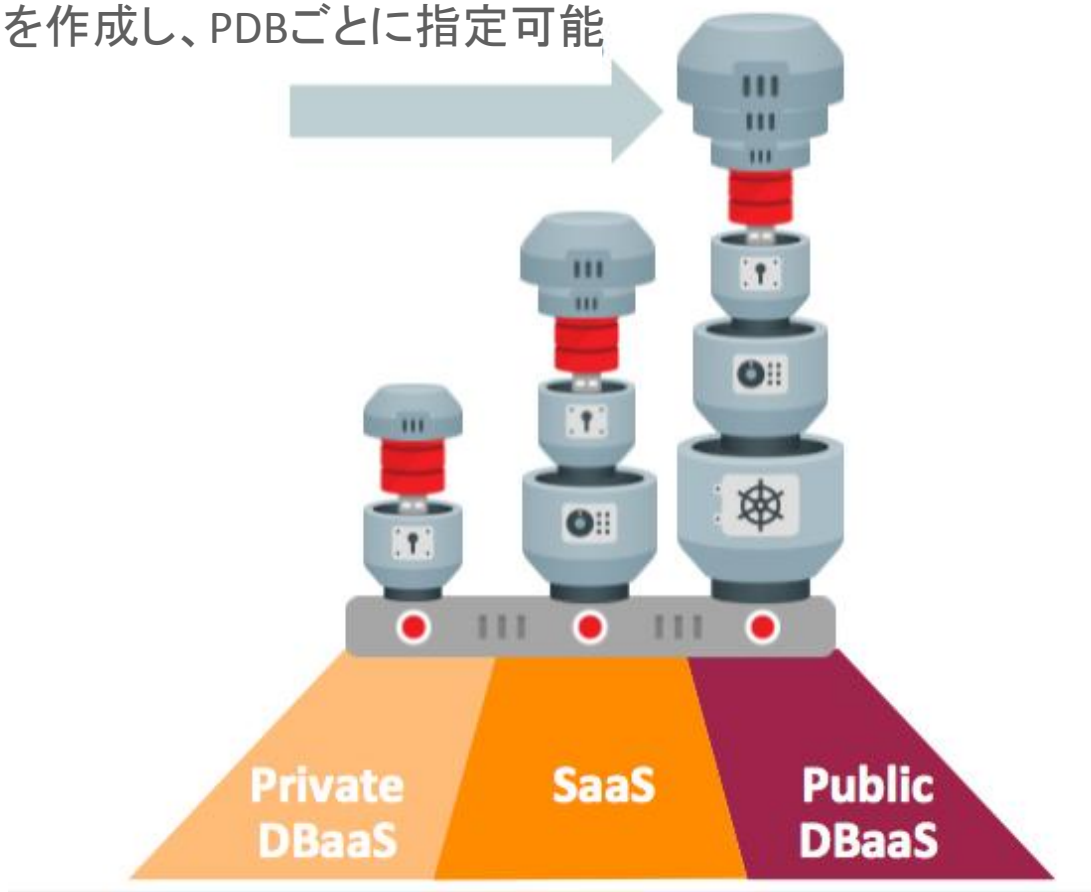
クラウドなどCDBを共有する環境でPDBごとの詳細なアクセス制御を実現

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



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;
```

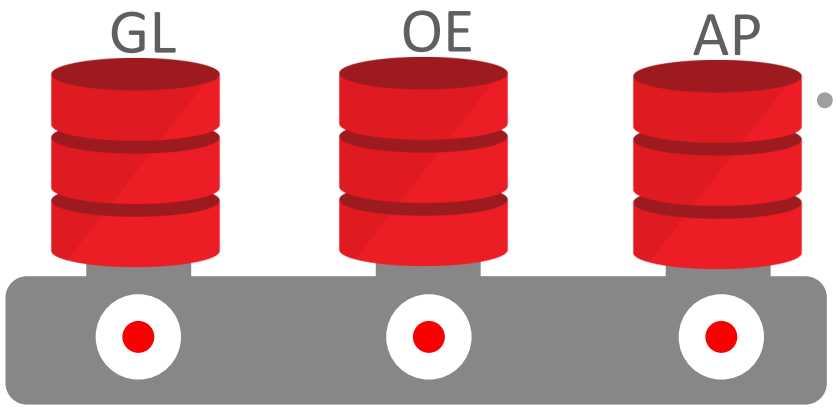
```
SQL> alter lockdown
     profile p1 disable
     statement=
       ('ALTER SYSTEM')
     clause=('SET')
     option= ALL EXCEPT
     ('cursor_sharing',
      'optimizer_mode');
```

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

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

PDBロックダウン・プロファイル

設定例

1. **ロックダウン・プロファイルの作成 (CDBで実施)**
   SQL> create lockdown profile prof1;

2. **PDB_LOCKDOWNパラメータの設定 (CDBで実施)**
   SQL> alter system set pdb_lockdown = prof1;

3. **Lockdown Profileの設定 (CDBで実施)**

   A) **機能制限**
   SQL> alter lockdown profile prof1 disable feature=('UTL_SMTP', 'UTL_HTTP'); # パッケージごとに指定する場合
   SQL> alter lockdown profile prof1 disable feature=('NETWORK_ACCESS'); # 機能バンドルを指定する場合

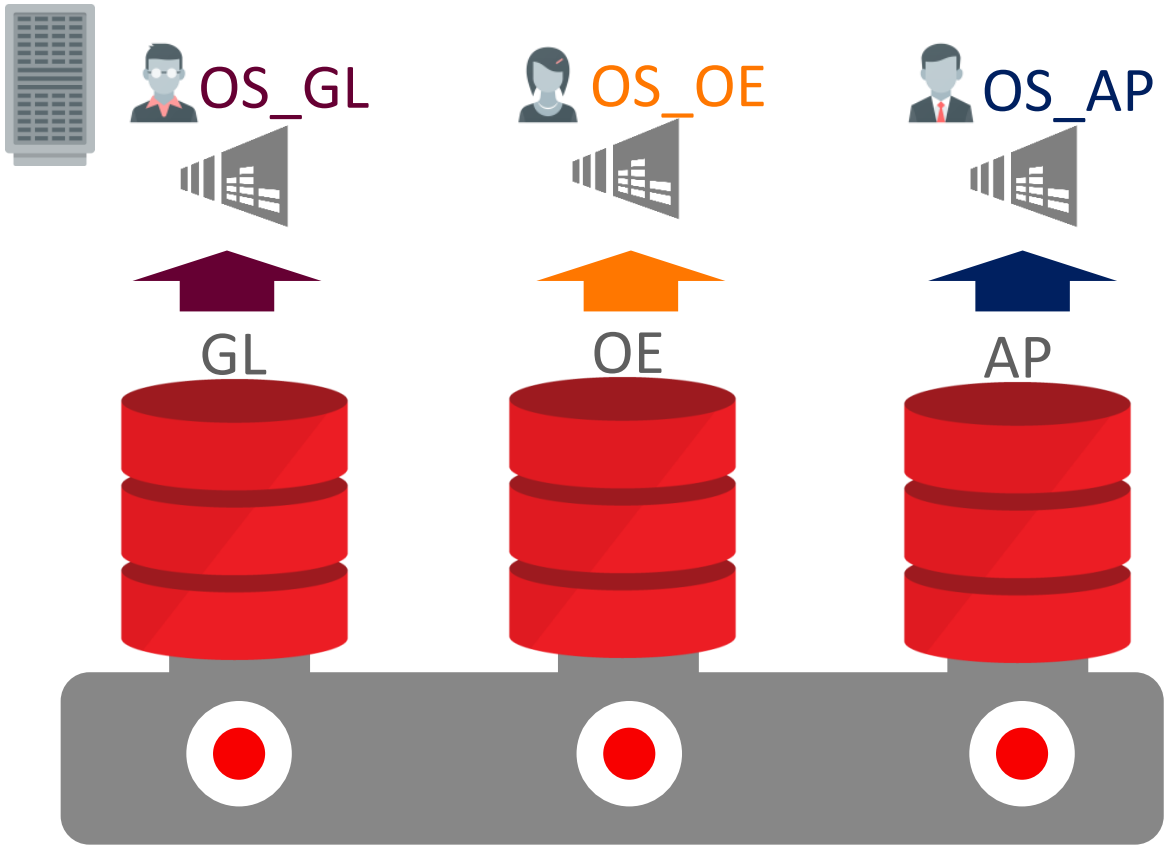
   B) **オプション利用制限**
   SQL> alter lockdown profile prof1 disable option=('PARTITIONING');

   C) **SQL文実行制限**
   SQL> alter lockdown profile prof1
   disable statement = ('alter system')
   clause = ('set') option = ('cpu_count') maxvalue = '4';

例えば、ALTER SYSTEMの実行制限を行う時には、ALTER SYSTEMの中のサブコマンド、オプションや変数の値まで詳細に指定して制限可能

PDB OSクレデンシャル

PDBごとにOS操作を行う際のOSユーザーを個別に指定



- 強力な権限を持つオラクル実行ユーザーではなく、PDBごとに個別のOSユーザーの権限でOS操作を実施。他PDBとのOSレベルでも権限を分割
- pdb_os_credentialパラメータに設定したOSユーザー権限でextprocエージェントを介して外部プロシージャを実行

設定例:

1. **OSクレデンシャルの作成(CDBで実施)**

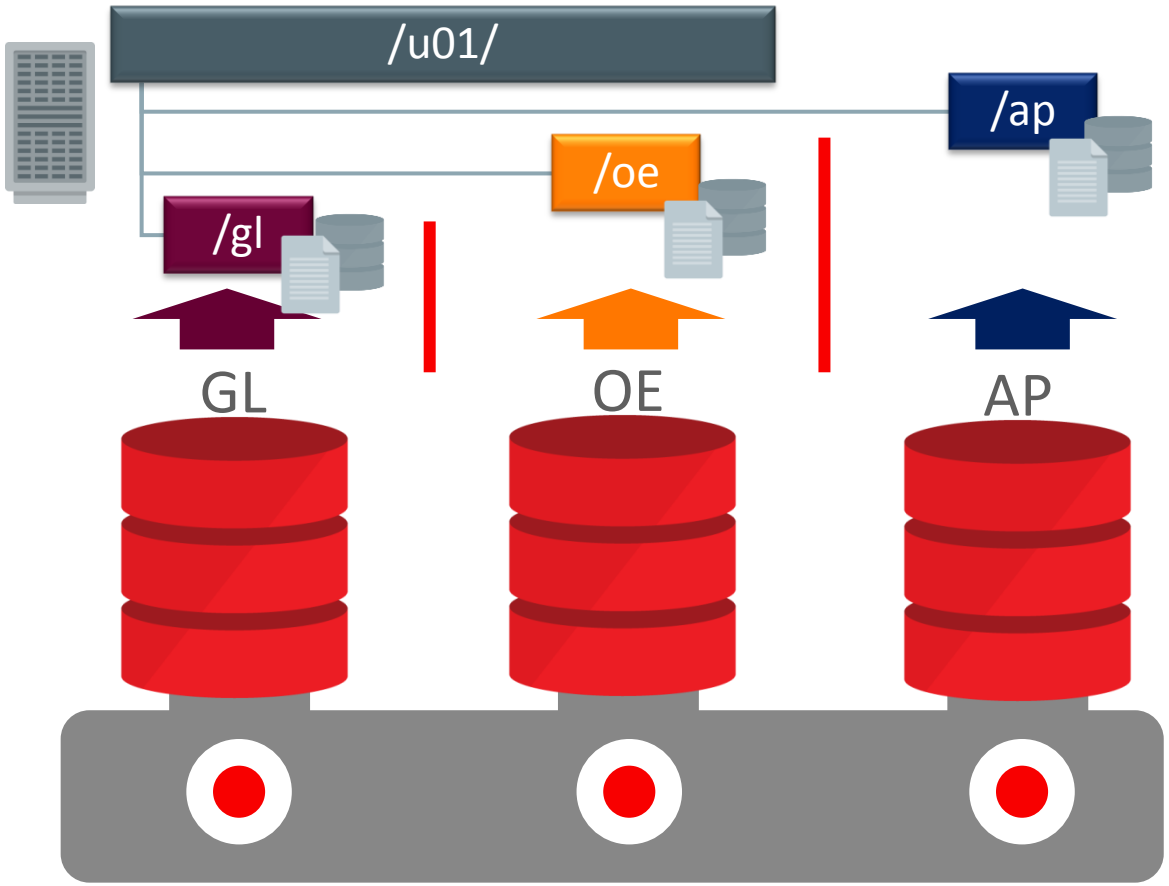
```
SQL> exec dbms_credential.create_credential
        (credential_name=> 'os_gl_user',
         username => 'os_gl',  password => 'welcome');
```

2. **PDB_OS_CREDENTIAL初期化パラメータの設定(PDBで実施、要再起動)**

```
SQL> alter system set pdb_os_credential=os_gl_user scope=spfile;
```

PDBパス・プレフィックス

PDBごとにアクセスできるディレクトリを制限



- 共有のOSディレクトリにある別のPDBのデータへのアクセスを防止。ディレクトリ・オブジェクトはPDBごとに指定したディレクトリ以下にのみ作成可能
- PATH_PREFIX(CREATE_FILE_DEST以下に指定)を利用することで、PDBからは相対パスでディレクトリを設定、PDB作成後に変更は不可

設定例:

1. **PDB作成時にCREATE_FILE_DESTとPATH_PREFIX句を指定** **(CDBで実施)**

```
SQL> create pluggable database gl
 admin user gladm identified by password
 create_file_dest = '/u01/gl'
 path_prefix = '/u01/gl/work/';
```

2. **CREATE DIRECTORY時にはPATH_PREFIXからの相対パスを指定** **(PDBで実施)**

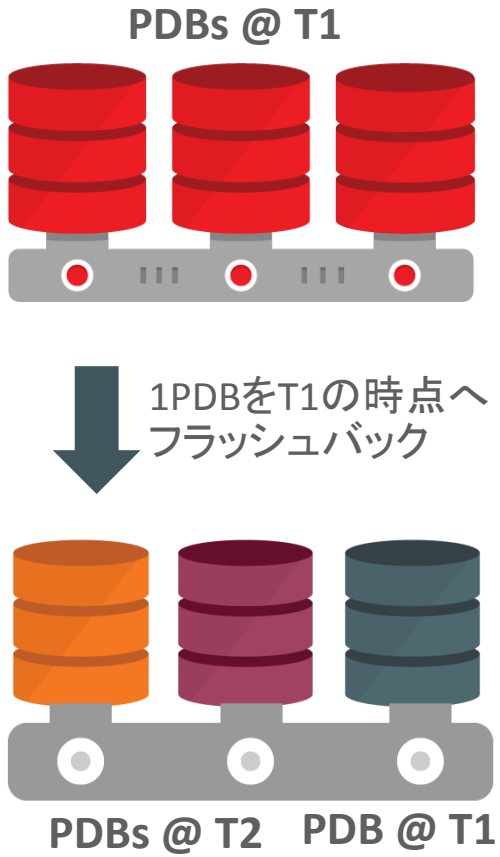
```
create directory dumpdir as 'dumpdir';
```

(実際には/u01/gl/work/dumpdirに作成される)

PDBフラッシュバック

PDBフラッシュバック

- PDBに対するフラッシュバック・データベースをサポート
  - PDBの全てのデータファイルがフラッシュバックされる
  - 他のPDBはオープン、稼働させた状態で特定のPDBのみフラッシュバックが可能
  - PDBリストア・ポイント、CDBリストア・ポイント、SCN、時間のいずれかを指定
  - フィジカル・データガード環境でもPDBフラッシュバックをサポート
- PDBリストア・ポイント
  - PDBリストア・ポイントの名前は、PDBごとに管理されるため、同じ名前のPDBリストア・ポイントを異なるPDBで使用可能
- RMANとSQLのインターフェース
  - 使用するUNDOモード、リストア・ポイントで選択



PDBリストア・ポイント

ユーザー定義のリストア・ポイントを各PDBで作成可能、3種類のリストア・ポイント

- 通常のリストア・ポイント

```
SQL> CREATE RESTORE POINT before_batch;
```

  - SCNまたは特定の時点に割り当てられるラベル
  - 制御ファイルに格納され、エージ・アウトされる(メンテナンス不要)

- 保証付きリストア・ポイント

```
SQL> CREATE RESTORE POINT before_upgrade
     GUARANTEE FLASHBACK DATABASE;
```

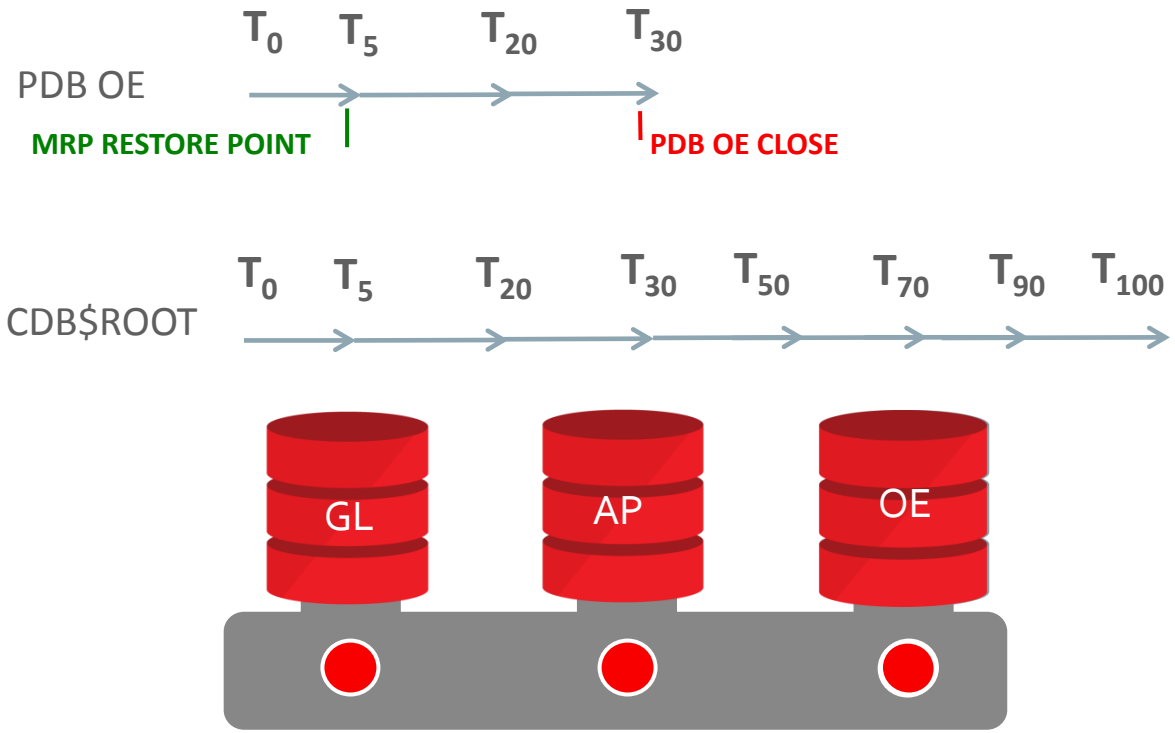
  - フラッシュバックを保証するリストア・ポイント
  - 制御ファイルに格納され、エージ・アウトされない(明示的に削除をする必要がある)

- クリーン・リストア・ポイント

```
SQL> CREATE CLEAN RESTORE POINT before_upgrade;
```

  - PDBがクローズされており、トランザクションがないPDBで作成可能
  - フラッシュバック時にリカバリの必要がない
  - 共有UNDOを使用するCDBでのみ作成可能

PDBフラッシュバック – 通常のリストア・ポイント



PRODUCTION

alter session set container = OE;

T_5 *create restore point mrp;*

T_{30} *alter pluggable database OE close;*

flashback pluggable database OE to restore point mrp;

T_5 *alter pluggable database OE open resetlogs;*

PDBフラッシュバック

ローカルUNDOモード使用時

- SQLを使用
  - データファイルのin-placeでフラッシュバックを実施
  - バックアップのリストアや補助インスタンスは不要

```
SQL> flashback pluggable database
     my_pdb to scn 411010;
```

共有UNDOモード使用時

- RMANを使用
  - 高速リカバリ領域を使用してUNDO表領域をリカバリ
  - 補助インスタンスを利用(PDB PITRと同様)
    - 補助インスタンスの使用する領域をAUXILIARY DESTINATION句で指定が可能
    - クリーン・リストア・ポイントへのPDBフラッシュバックは補助インスタンスは不要

```
RMAN> flashback pluggable database
      my_pdb to scn 411010;
```

PDBフラッシュバック – 設定と実行手順

CDBの構成を確認

- 高速リカバリ領域の構成、アーカイブ・ログ・モード
- UNDOモード(ローカル/共有)

CDBでフラッシュバックを有効化

```
SQL> alter database flashback on;
```

PDBでリストア・ポイントを作成

PDBで行う場合

```
SQL> alter session set container OE;
SQL> create restore point BEFORE_BATCH;
```

CDB$ROOTで行う場合

```
SQL> alter session set container = CDB$ROOT;
SQL> create restore point BEFORE_BATCH
     for pluggable database OE;
```

PDBをクローズして、リストア・ポイントまでフラッシュバックを実行
ローカルUNDOを使用している場合

```
SQL> alter pluggable database OE close;
SQL> flashback pluggable database OE to restore point BEFORE_BATCH;
```

PDBをresetlogsオプションを指定してオープン

```
SQL> alter pluggable database OE open resetlogs;
```

PDBフラッシュバック

- PDBフラッシュバック実行後も取得済みのデータベースは有効
- PDBのPoint-in-Timeリカバリまたは、PDBフラッシュバック実行後に、CDBのフラッシュバック・データベースも可能

- Oracle Flashback Technologyの対応

  12c R2ではPDBレベルでのフラッシュバック機能に対応

  - フラッシュバック・クエリー
  - フラッシュバック・バージョン・クエリー
  - フラッシュバック・データ・アカーブ
  - フラッシュバック・テーブル
  - フラッシュバック・ドロップ
  - フラッシュバック・データベース

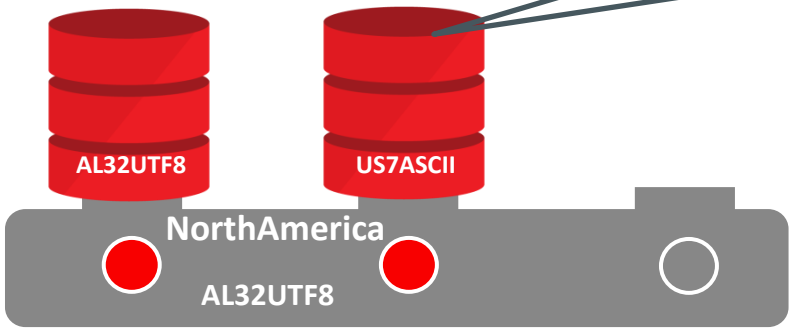
PDBごとのキャラクタ・セット

PDBごとのキャラクタ・セットのサポート

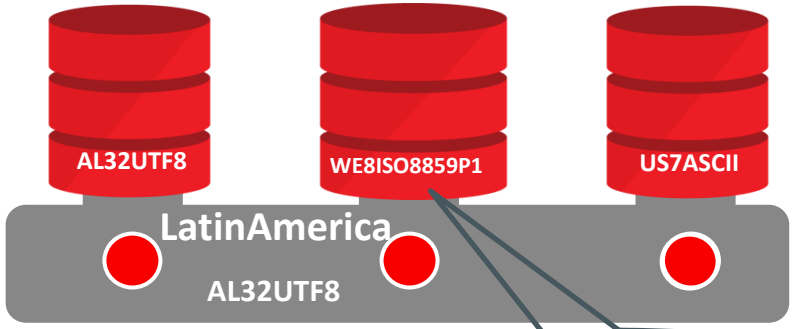
- 1CDB上で異なるキャラクタ・セットのPDBをプラグし、稼働させることが可能
  - CDB$ROOTのキャラクタ・セットは**AL32UTF8**を指定して作成
  - 異なるキャラクタ・セットのPDBをプラグまたはリモート・クローンにより作成することが可能
    - 国際化キャラクタ・セットもPDBごとに別でも可
  - SEEDから作成するPDBは**AL32UTF8**で作成される
    - PDB作成時に異なるキャラクタ・セットの指定は不可
  - アプリケーション・コンテナ内は全て同じキャラクタ・セットを使用
    - アプリケーション・ルートとアプリケーションPDBは同じキャラクタ・セット
- プラグされたデータは不変
  - キャラクタ・セット変換は不要
  - 新しいデータはPDBのキャラクタ・セットで挿入
- CDB$ROOTからContainers句を使用した検索
  - 異なるキャラクタ・セットのPDBをContainers句を使用した検索が可能
  - UNICODEへのデータ変換時にバイト数が増えるため、文字列がトランケートされないよう、UNICODE側の表定義を大きめに定義

PDBごとのキャラクタ・セット

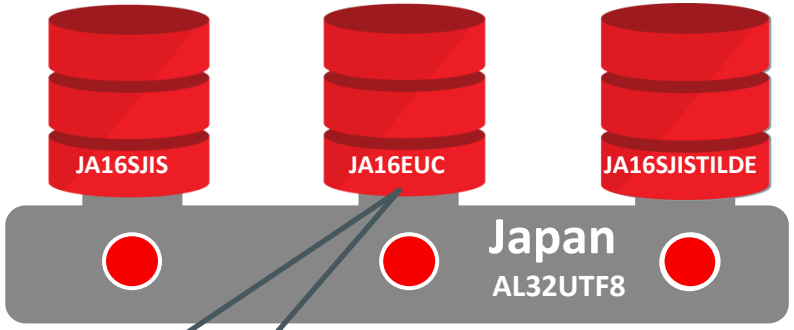
SQL> insert into emp (enum,name) values (1,'Wheeler');
1 row created.



SQL> insert into emp (anzahl,name) values (1,'Fuß-Schröder');
1 Zeile wurde erstellt.



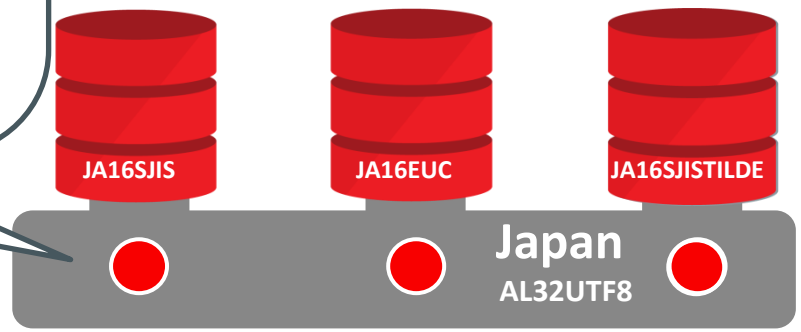
SQL> insert into emp (num,nome) values(1,'Tição');
1 linha criada.



SQL> insert into emp (numero,name) values (1,'ジョン');
1行が作成されました。

PDBごとのキャラクタ・セット

```
SQL> select con_id, parameter, value
     from   CONTAINERS(nls_database_parameters)
     where  parameter like 'NLS_CHARACTERSET';

 CON_ID PARAMETER          VALUE
------- ------------------ --------------
      1 NLS_CHARACTERSET   AL32UTF8
      3 NLS_CHARACTERSET   JA16SJIS
      4 NLS_CHARACTERSET   JA16EUC
      5 NLS_CHARACTERSET   JA16SJISTILDE
```



PDBレベルの診断能力の強化

PDBごとのアラート、トレース、AWR

PDBレベルの診断能力の強化

- 従来のPDBレベルの診断データの参照
  - アラート、トレース・ファイル、ログ・ファイルは、自動診断リポジトリ(Automatic Diagnostic Repository – ADR)に保存される
  - これらのファイルのアクセスは、ファイル・システムが提供するパーミッションで制御される
    - マルチテナント環境では大抵のPDB管理者はファイル・システム上のADRのファイルを参照が不可
  - CDB管理者
    - 特定のPDBに紐づくトレースを参照できると便利
  - PDB管理者 /アプリケーション開発者
    - PDBからアプリケーション・トレースを参照できると管理や開発工数を削減できる

アプリケーション・トレース:
- SQLトレース / Event 10046
- オプティマイザ・トレース/ Event 10053

- 12c R2での強化: V$ビューを介した参照
  - 現在のPDBに紐づくアラート・ログ、トレース・ファイルの情報のみをV$ビューにより参照
  - トレース・ファイル、インシデント・ダンプ、ログ・ファイルのすべてにPDBの情報を付帯
  - PDBの情報は構造化メタデータの一部として、トレース・ファイル(.trmファイル)に保存
  - トレース・レコードの属性
    - CON_ID: CDB単位でのID
    - CON_UID: PDBで一位となるユニークなID
    - NAME: PDBの名前
  - アプリケーション・トレース
    - トレース情報にPDB情報を付帯
    - トレース・ファイル全体にアクセスしなくてもSQLトレースやオプティマイザ・トレースの情報を参照可能

PDBごとのアラート・ログ

- v$diag_alert_extビュー
  - 現在のPDBのアラート・ログの内容を表示
  - CDB$ROOTからはすべてのPDBの内容が参照可能
  - アプリケーション・ルートからはアプリケーション・コンテナ内のPDBの内容が参照可能

例: CON_ID=7のアプリケーションPDBでの検索結果

```
SQL> select con_id, originating_timestamp, message_text

from v$diag_alert_ext;

CON_ID ORIGINATING_TIMESTAMP

------ ---------------------------------------------

MESSAGE_TEXT

--------------------------------------------------------------

....

     7 16-09-02 07:26:58.951000000 +09:00
   alter pluggable database application snowsports sync

     7 16-09-02 07:26:59.083000000 +09:00
   alter pluggable database application snowsports begin install
   '1.0'

     7 16-09-02 07:26:59.123000000 +09:00
   Completed: alter pluggable database application snowsports begin
   install '1.0'

     7 16-09-02 07:26:59.158000000 +09:00
   create tablespace app_tbs datafile size 100M autoextend on next
   10M maxsize 200M

     7 16-09-02 07:26:59.489000000 +09:00
   Completed: create tablespace app_tbs datafile size 100M autoextend
   on next 10M maxsize 200M

....
```

PDBごとのトレース

- 現在のPDBから参照できるトレース・ファイルの一覧
  - V$DIAG_TRACE_FILE
    - 現在のPDBのトレースを含むトレース・ファイル名の一覧
  - V$DIAG_APP_TRACE_FILE
    - アプリケーション・トレースを含むトレース・ファイル名の一覧
- トレース・ファイルの内容を表示
  - V$DIAG_TRACE_FILE_CONTENTS
    - 現在のPDBのトレース・レコードを表示
- アプリケーション依存のトレース・レコードを表示
  - V$DIAG_SQL_TRACE_RECORDS
    - 現在のPDBのSQLトレース・レコードを表示
  - V$DIAG_OPT_TRACE_RECORDS
    - 現在のPDBのオプティマイザ・トレース・レコードを表示
- 現在のセッションのアプリケーション依存のトレース・レコードを表示
  - V$DIAG_SESS_SQL_TRACE_RECORDS
    - 現在のPDBでセッション単位のSQLトレース・レコードを表示
  - V$DIAG_SESS_OPT_TRACE_RECORDS
    - 現在のPDBでセッション単位のオプティマイザ・トレース・レコードを表示

PDBごとのトレース:SQLトレースの出力例

- アプリケーション依存のトレース・レコードを参照するため、application_trace_viewer権限を付与
  - grant application_trace_viewer to <ユーザー名>;
- SQLトレース、オプティマイザ・トレースを取得のために、alter sessionシステム権限を付与
  - grant alter session to <ユーザー名>;
- 実行例:

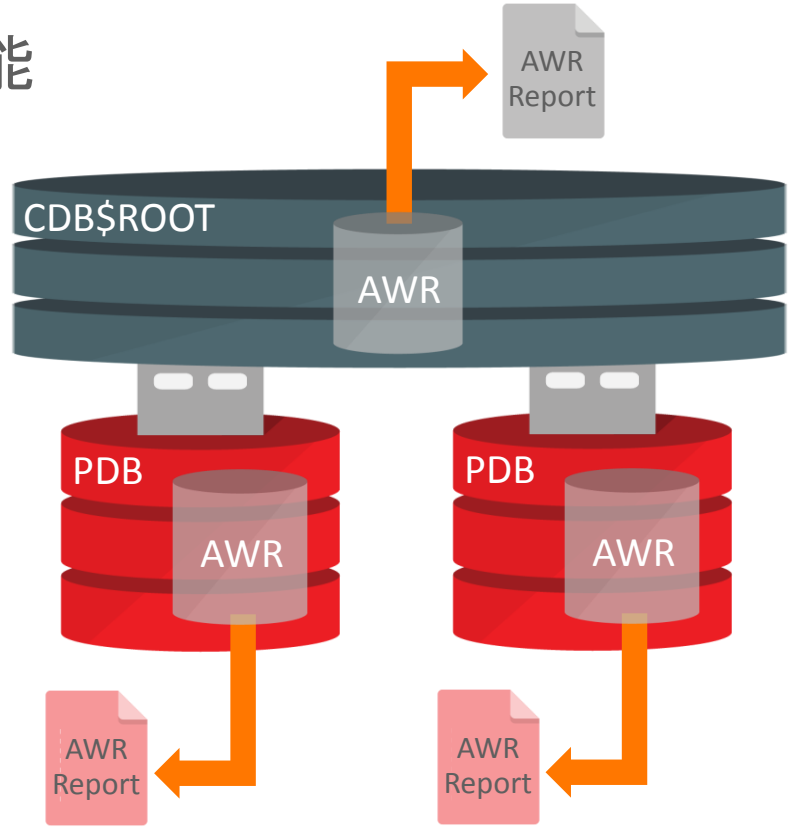
```
alter session set events '10046 trace name context forever,
    level 1';
select * from scott.dept;
alter session set events '10046 off';
select payload from v$diag_sess_sql_trace_records;
```

```
SQL> select payload from v$diag_sess_sql_trace_records;

PAYLOAD
-------------------------------------------------------

CLOSE #140203348047584:c=0,e=13,dep=0,type=1,tim=19499743077100
=====================
PARSING IN CURSOR #140203348024752 len=24 dep=0 uid=107 oct=3
lid=107 tim=19499743078121 hv=911793802 ad='e4ce1d80'
sqlid='f6hhpzwv5jrna'

select * from scott.dept

END OF STMT

PARSE
#140203348024752:c=0,e=157,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=
3383998547,tim=19499743077998
EXEC
#140203348024752:c=0,e=57,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3
383998547,tim=19499743078284
FETCH
#140203348024752:c=0,e=158,p=0,cr=3,cu=0,mis=0,r=1,dep=0,og=1,plh=
3383998547,tim=19499743078553
FETCH
#140203348024752:c=0,e=28,p=0,cr=1,cu=0,mis=0,r=3,dep=0,og=1,plh=3
383998547,tim=19499743078969

STAT #140203348024752 id=1 cnt=4 pid=0 pos=1 obj=73161 op='TABLE
ACCESS FULL DEPT (cr=4 pr=0 pw=0 str=1 time=147 us cost=2 size=120
card=4)'

CLOSE #140203348024752:c=0,e=12,dep=0,type=0,tim=19499747062509
=====================
```

AWRスナップショット

PDB レベルスナップショットのサポート

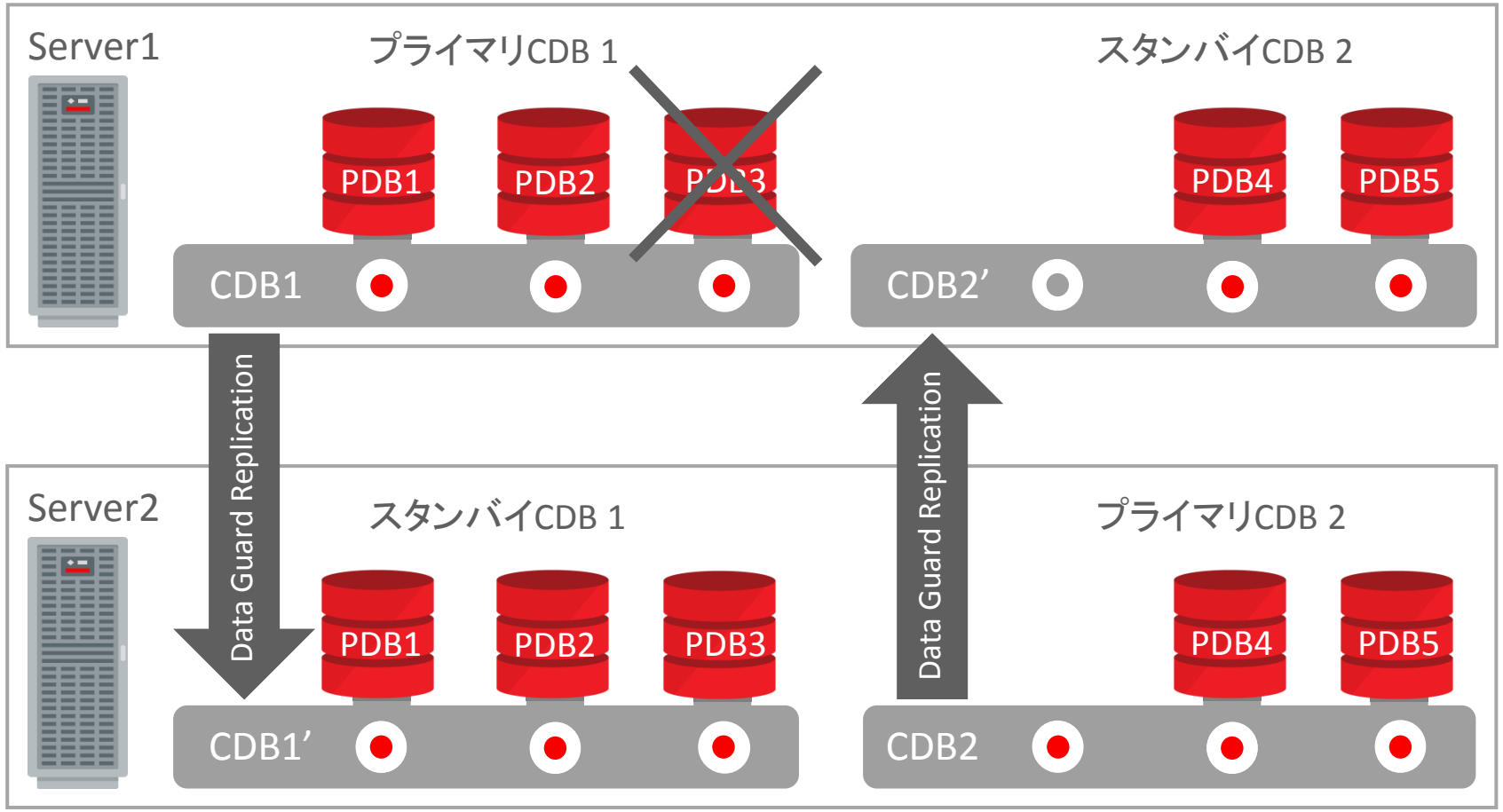
- 各PDBでスナップショットの取得が可能に
  - 取得したスナップショットはPDBのSYSAUX表領域に保存
  - 以下のスナップショット設定はPDBそれぞれで設定可能
    - スナップショット取得間隔(SNAP_INTERVAL)
    - スナップショット保存期間(RETENTION)
    - 収集するTop SQL の数(TOPNSQL)
  - 自動スナップショット取得はデフォルトでは無効
- CDB$ROOTのAWRは12c R1と基本同機能
  - 1 時間毎にスナップショット自動取得、8日間保存
  - 自動スナップショット取得はデフォルトで有効
  - 拡張ポイント：PDB統計情報を強化



構成の柔軟性の強化

Data Guard Brokerのマルチテナント環境への対応強化

PDBレベルのフェイル・オーバーのサポート



- 2つの各サーバー上に別CDBがあり、各CDBごとにData Guardによるレプリケーションを行っている環境
- プライマリでPDB障害が起きた際、対象のPDBを、スタンバイから同サーバー上の別CDBのプライマリに移動
  - フェイル・オーバーに有効

圧縮されたアーカイブ・ファイルによるアンプラグ・プラグ

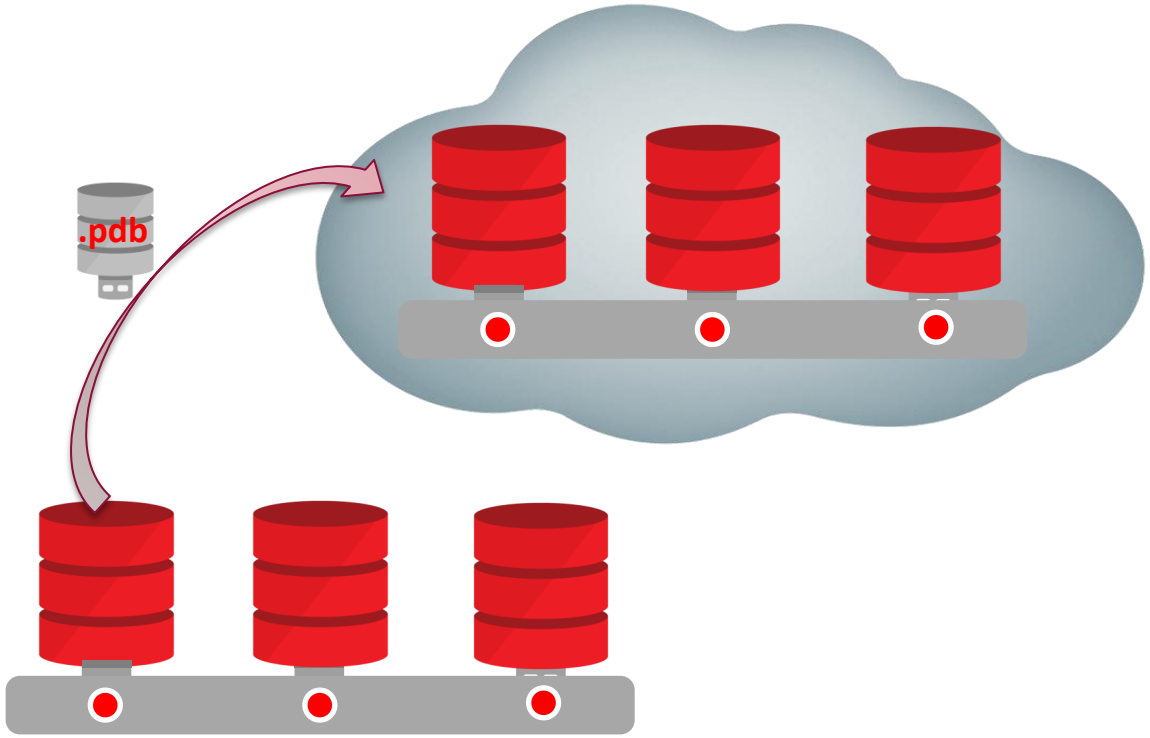
- PDBのアンプラグ時にXMLファイルとPDBのデータファイルを圧縮した1つのアーカイブ・ファイルとして作成可能
  - Walletファイルも含む
  - 作成されたアーカイブは.pdbの拡張子
- プラグ時はXMLファイルを指定する代わりに.pdbファイルを指定、自動でデータファイルが展開される
- アンプラグ

```
SQL> alter pluggable database testpdb1
      unplug into '/temp/testpdb1.pdb';
```

- プラグイン (同じファイルパスで作成する場合)

```
SQL> create pluggable database testpdb2
     using '/temp/testpdb1.pdb';
```

- 異なるCDB環境にPDBに関連するファイルをまとめて移動するのに便利



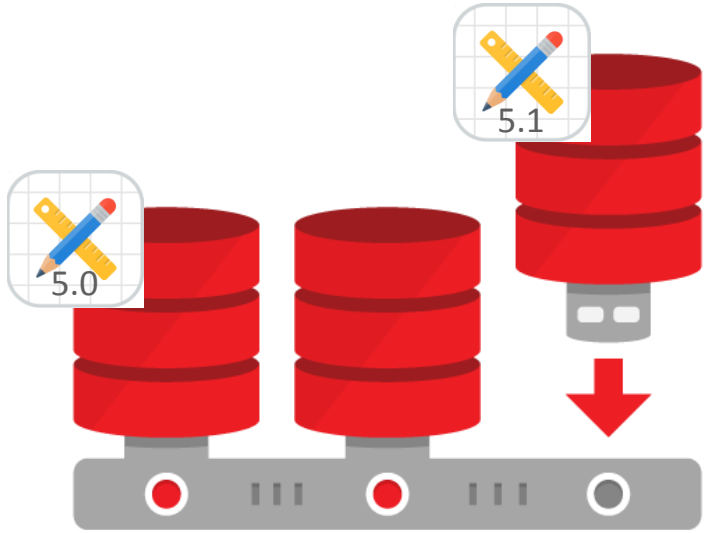
リソース制御に関連するパラメータ

- **MAX_PDBS**
  - CDBまたは、アプリケーション・コンテナ内に作成できるPDB数 (PDB$SEEDはカウント対象外)
  - PDBで設定可否: NO (CDB$ROOT、アプリケーション・ルートのみで設定可能)
  - デフォルト値:　4096
- **CONTAINERS_PARALLEL_DEGREE**
  - CONTAINERS句を使用した場合の並列度を指定
  - PDBで設定可否: YES
  - デフォルト値:　65535
    - 変更しない場合(65535のままの場合)の並列度:
    - CDB$ROOT: 1+PDB数
    - アプリケーション・ルート: 1+アプリケーションPDB数
- **ENABLE_AUTOMATIC_MAINTENANCE_PDB**
  - 自動メンテナンスタスクの有効・無効を設定するパラメータ
  - PDBで設定可否: YES
  - デフォルト値:　true
- **AUTOTASK_MAX_ACTIVE_PDBS**
  - 自動メンテナンスタスクが同時に動くPDBの数
  - PDBで設定可否: NO
  - デフォルト値:　2

PDBレベルでの構成の柔軟性の拡充

- PDBごとに設定可能になった初期化パラメータ
  - v$parameter.Ispdb_modifiable=true
  - 代表的なパラメータ
    - db_files
    - parallel_max_servers
    - undo_retention
    - utl_file_dir

- PDBごとのAPEXの構成
  - CDB$ROOTにAPEXは初期設定ではインストールされていない
  - PDBごとに異なるAPEXのバージョンを構成可能



PDBレベルでの操作/機能サポートの拡張

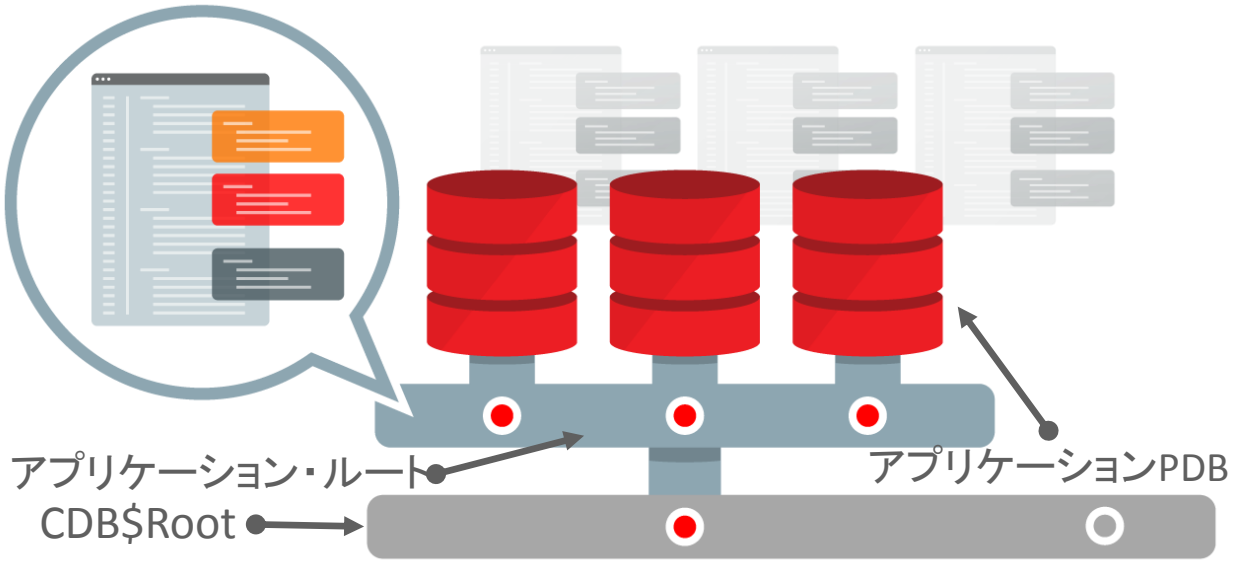
- PDBのabortモードのクローズ
  - abortモードでクローズすることが可能
    - CDBがアーカイブ・ログ・モード
  - abortモードでクローズ後は、次回オープン時にメディア・リカバリが必要となる
    - recover databaseなどを実行
- 自動データ最適化(Automatic Data Optimization - ADO)、ヒート・マップの対応
  - PDBレベルでHEAT_MAPを有効にして、セグメントの移動や圧縮などの操作の自動化が可能

3. アプリケーション・コンテナ

テナントの中央集中管理

アプリケーション・コンテナ

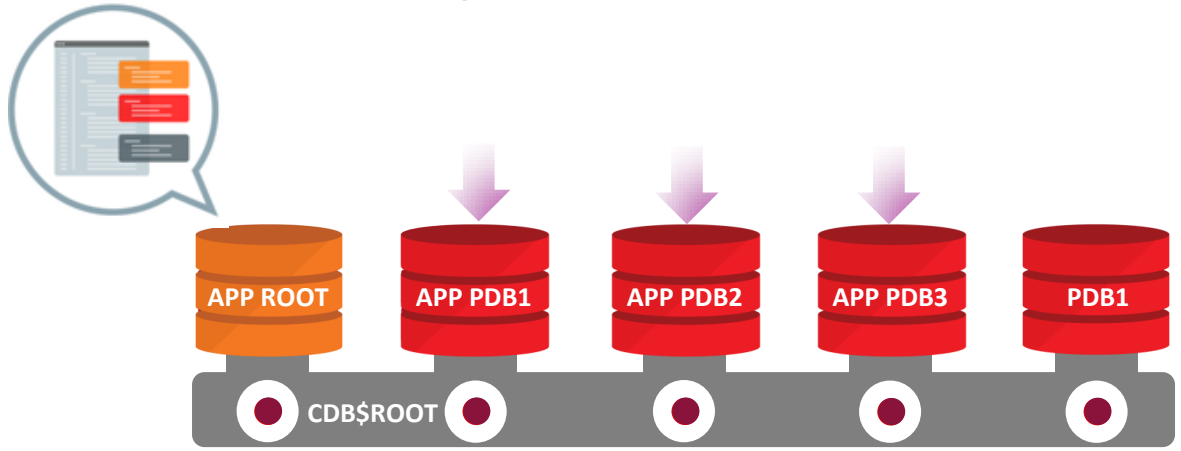
- アプリケーション・コンテナの構成要素
  - アプリケーション・ルート(マスター)
  - アプリケーションPDB (各テナント用)
  - アプリケーション・シード(プロビジョニング用)
- PDB間でオブジェクトを共有
  - コード、メタデータおよびデータ
  - アプリケーションのインストールは一度だけ
- さらに容易な管理
  - アプリケーションPDBの即時プロビジョニング
    - アプリケーション・シードPDBを利用
  - アプリケーション・コンテナにアップデート適用
    - テナントのPDBに対してマスターから同期
- あらゆるアプリケーションに最適
  - パッケージ・アプリケーション、SaaS、部門ごとのアプリなど



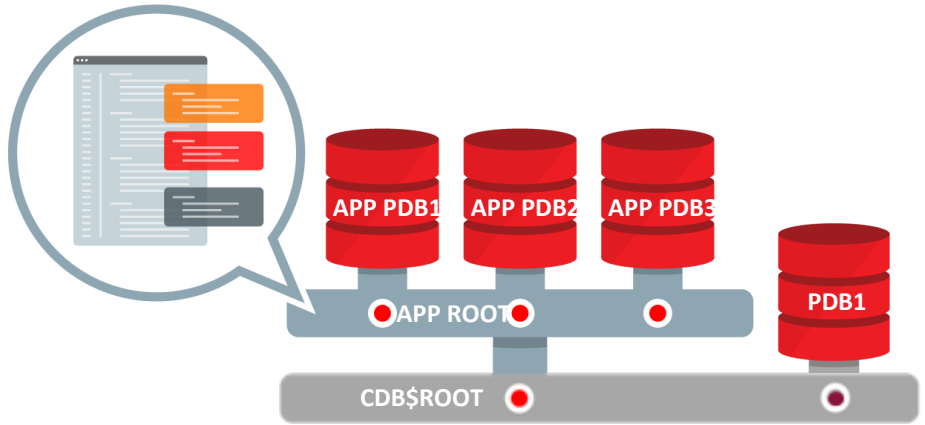
アプリケーション・コンテナでの"アプリケーション"
- アプリケーションのバック・エンドのデータベース・オブジェクト

アプリケーション・コンテナの構成

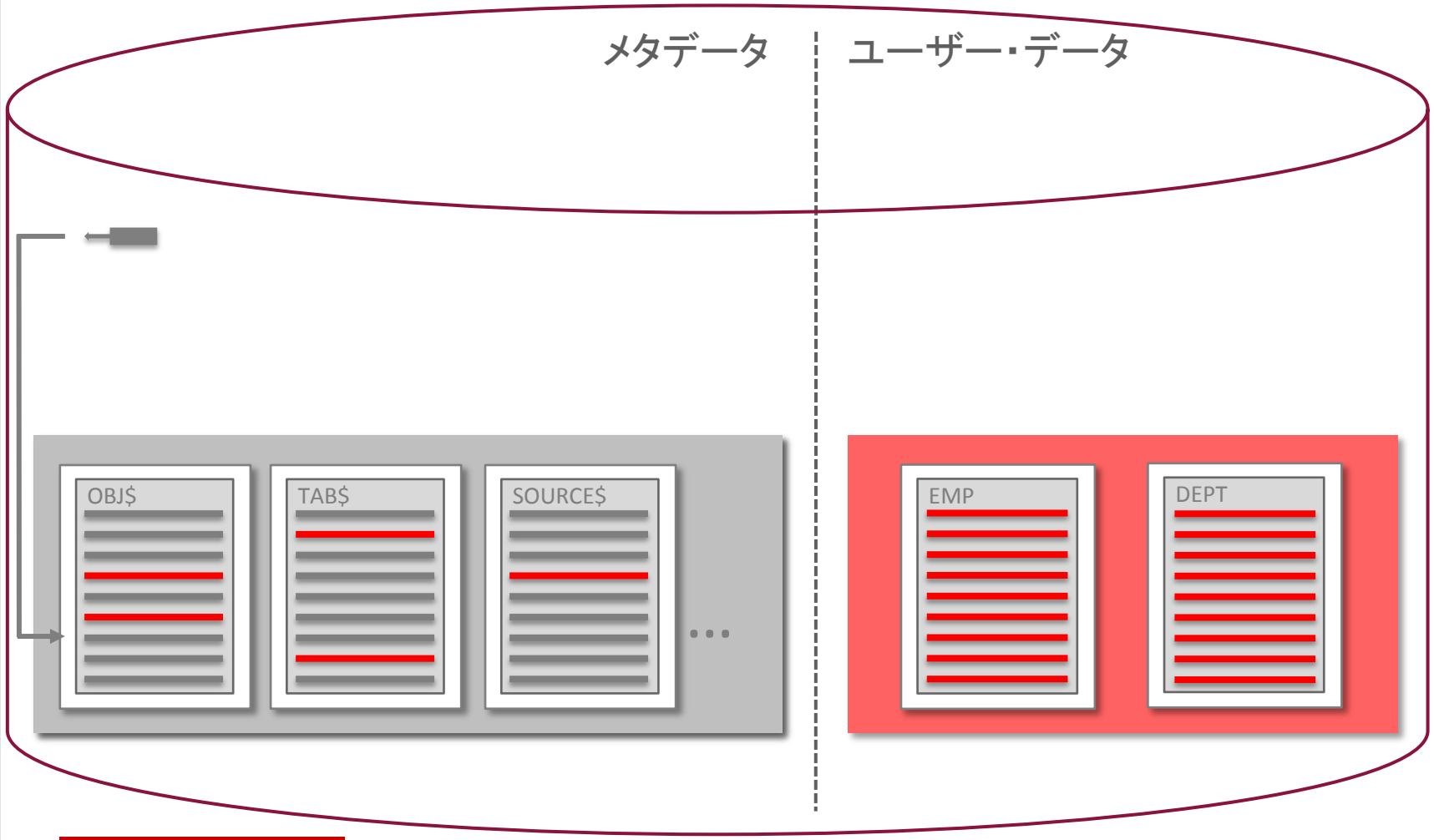
- 物理的な表現
  - CDB$ROOT上からはアプリケーション・ルート、アプリケーションPDB、アプリケーション・シードも通常のPDBと同様のPDBとして扱われる
  - 1CDB上に複数のアプリケーション・ルートを作成することも可能
  - 1CDB上にアプリケーション・コンテナとともに、通常のPDBも稼働できる



- 論理的な表現
  - アプリケーション・ルートからはアプリケーションPDB、アプリケーション・シードのみが管理対象となる
  - アプリケーション・コンテナ
    - アプリケーション・ルートとそれに紐付いたすべてのアプリケーションPDBの集合

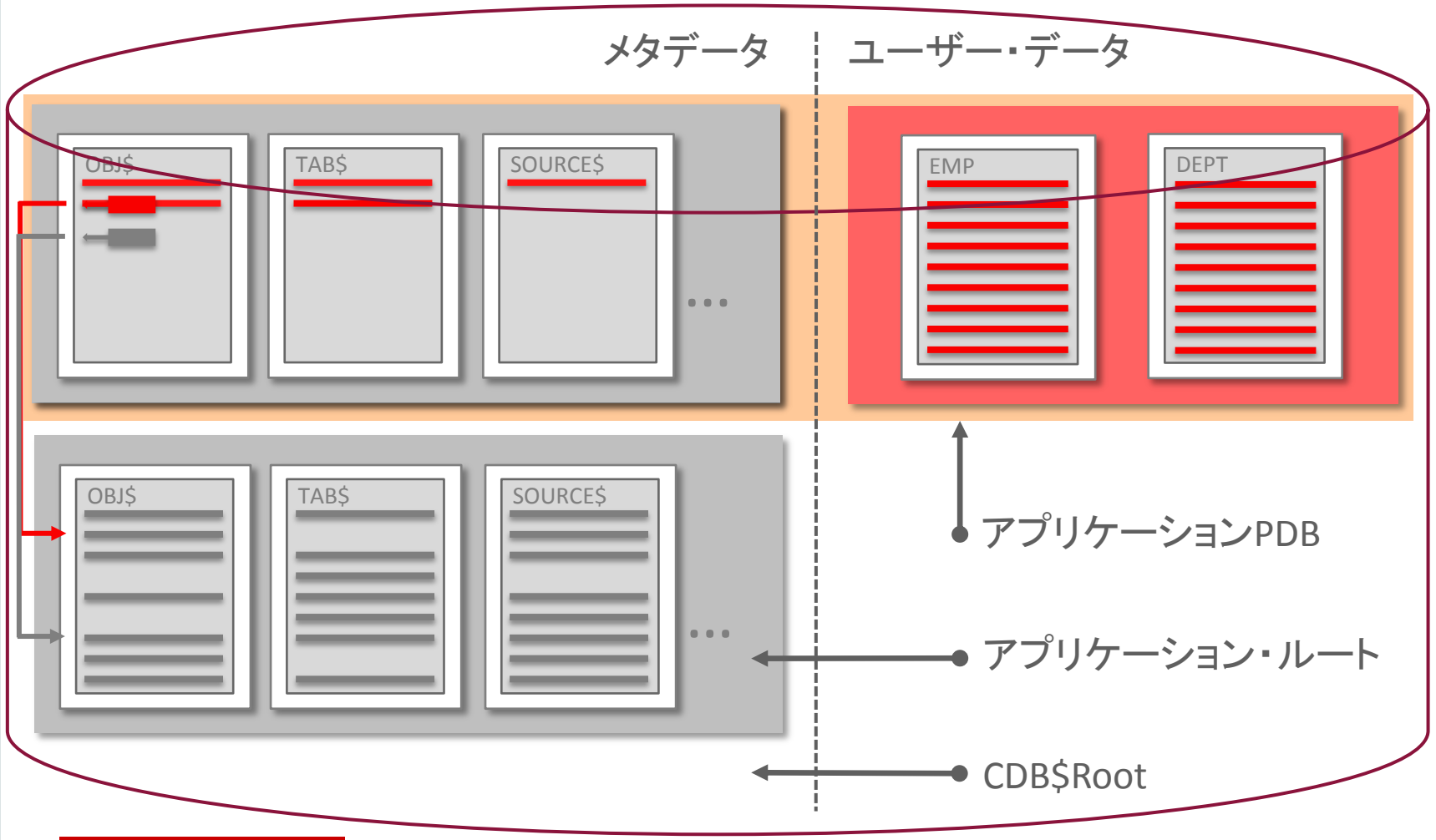


データ・ディクショナリの水平分割
マルチテナント・アーキテクチャによる *可搬性と互換性* の実現



- 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
- データ・ディクショナリの水平分割
  - Only Oracle-supplied metadata remains in Root
  - PDB mobility now easy!
- PDBのデータ・ディクショナリの一部はRootの定義を参照
  - Oracle固有のメタデータのみがRootに存在
  - non-CDBとの互換性を維持

アプリケーションPDBのデータ・ディクショナリ
ユーザー作成の共通オブジェクトの定義をアプリケーション・ルートで管理



- アプリケーション・コンテナはアプリケーション・ルートが基となる
- アプリケーション・ルートは特殊なタイプのPDB
- アプリケーション・ルートにアプリケーションの共通オブジェクトのマスター定義を含む
- アプリケーションPDBはアプリケーション・ルート上で定義される
- アプリケーションPDBのデータ・ディクショナリの一部はアプリケーション・ルートの定義を参照

アプリケーション共通オブジェクト

- アプリケーションの共通するオブジェクトはアプリケーション・ルートにのみ存在
  - CDB$ROOTに存在するOracleが提供する共通オブジェクトに類似
- 用途に応じて3つの共有タイプを指定して作成
  - METADATA
  - DATA
  - EXTENDED DATA
- オブジェクトの作成時にsharing句で指定
  - default_sharingパラメータによりsharing句を省略することも可能、デフォルト値はMETADATA

Metadata

- オブジェクトの定義を共有
  - 例: 表、PL/SQLパッケージなど

Data

- アプリケーション・ルートに存在するデータを全てのPDBで共有
  - 例: 参照用データのテーブル

Extended Data

- 全PDBで共通するデータとPDB個別のデータ
  - 例: 全てのPDBで共有するデータとPDB内でのみ利用するデータを含むテーブル

アプリケーション共通オブジェクト

Sharing = Metadata

- アプリケーション・ルートで定義し、各アプリケーションPDBで利用可能
- データは各PDB内に配置
- アプリケーション・ルート内のみにデータを配置することも必要に応じて可能
- *default_sharing*パラメータのデフォルト値は*Metadata*
- ほとんどのアプリケーション・オブジェクトはMetadataタイプとなると考えられる

Sharing = Data

- 定義とデータの両方をアプリケーション・ルート内に配置
- 共通するアプリケーション・データを配置し、全ての各アプリケーションPDB/テナントで共有する場合に有用
- DWHのディメンジョン・テーブルに最適

Sharing = Extended Data

- データの一部は全てのアプリケーションPDBで共通
- アプリケーション・ルートに格納されている共通データを参照しながら、各アプリケーションPDB /テナントでは固有のデータを持てる

アプリケーションの定義と管理

- アプリケーションの属性
  - アプリケーション名
    - アプリケーションを一意に特定するためのもの
  - アプリケーション・バージョン
    - 文字、数字、記号、スペースを使用可能、大文字小文字を区別
- CDBに複数のアプリケーション・ルートを作成可能
- アプリケーション・ルートに複数のアプリケーションのインストール可能
- アプリケーションの情報は次のビューで確認

- DBA_APPLICATIONS
- DBA_APP_PDB_STATUS
- DBA_APP_STATEMENTS
- DBA_APP_VERSIONS
- DBA_APP_PATCHES
- DBA_APP_ERRORS

- アプリケーションに対する操作
  - アプリケーション・ルート上で実施
  - ALTER PLUGGABLE DATABASE APPLICATION ... BEGIN文とALTER PLUGGABLE DATABASE APPLICATION ... END文の間でアプリケーションに対するSQL文を実行
  - インストール
    - アプリケーションの新規作成、名前とバージョンを指定
  - アップグレード
    - 既存のアプリケーションの変更、アプリケーション名とアップグレード元のバージョンと先のバージョンを指定
  - パッチ
    - オブジェクトの構造を変更を伴わない小規模の変更
    - アプリケーション名、パッチ番号、適用可能バージョンを指定パッチ番号は数字のみ使用可能
  - アンインストール
    - アプリケーションの削除、アプリケーション名を指定

アプリケーションのインストール

スクラッチで作成

- アプリケーション管理者ユーザーがSQLスクリプトを使用して作成
- スクリプト内のオブジェクト定義に適切な共有タイプとなるようSharing句を指定
- 必要に応じて*default_sharing*パラメータを指定

Data Pumpを利用

- 対象となるアプリケーションのデータベース・オブジェクトをData Pump Exportにより、ダンプ・ファイルを生成
- アプリケーションのインストール・スクリプトをダンプ・ファイルからData Pump ImportでSQLFILEオプションを指定して実行し、生成
- 作成したスクリプトを編集

既存PDBをプラグイン

- 既存のPDBをCDBプラグインまたはリモート・クローンで作成
- アプリケーション名、バージョンをALTER PLUGGABLE DATABASE文で指定
- DBMS_PDBパッケージで共通ユーザー、共通オブジェクト等を指定

アプリケーション・コンテナ – 作成と運用のプロセス

1. アプリケーション・コンテナを作成
2. アプリケーションをアプリケーション・ルートにインストール
   - *アプリケーションのデータ・モデルの作成、アプリケーション共通ユーザー、共通オブジェクトの構成*
3. アプリケーション・ルートにアプリケーションPDB/テナントを作成
4. 各アプリケーションPDBで、アプリケーション・ルートで構成されたアプリケーションを同期
   - *コードの変更を伴わずにマルチテナント環境を構築*
5. 各アプリケーションPDBでデータのロード
6. アプリケーションのマスター定義をアップグレード
   - *テナントへの影響を発生させずにアプリケーションのアップグレード可能*
7. 各テナントの状況に応じてアプリケーションの定義を同期
   - *全テナントのアプリケーションのアップグレードを行うためのメンテナンス期間の調整は不要*
8. 新しいアプリケーションPDBを必要に応じて作成

アプリケーション・コンテナ – 設定と実行手順

PDBをアプリケーション・ルートとして作成、またはプラグ

```
SQL> create pluggable database app_root as application container..
```

アプリケーション・ルート内でアプリケーション・データ・モデルを作成

```
SQL> alter pluggable database application MYFIRSTAPP begin install '1.0';
     アプリケーション用オブジェクトのDDLを実行
SQL> alter pluggable database application MYFIRSTAPP end install '1.0';
```

アプリケーション・ルート上でアプリケーションPDBの作成

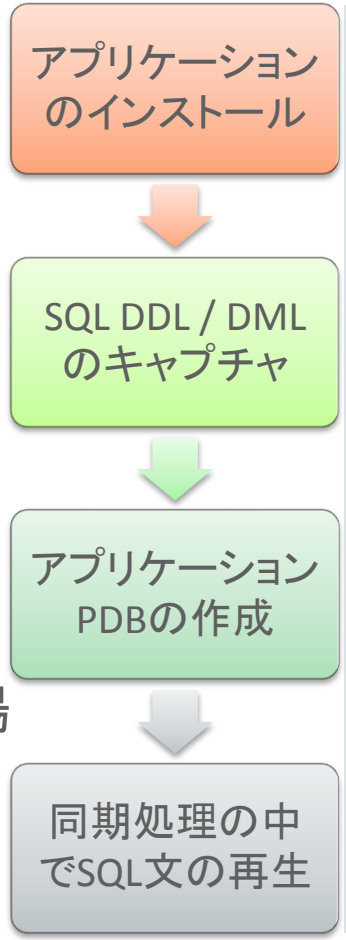
```
SQL> create pluggable database app_pdb1 ..
```

アプリケーションPDB内でアプリケーションの同期

```
SQL> alter pluggable database application MYFIRSTAPP sync;
```

アプリケーションの同期の仕組み

- Begin InstallとEnd Installの間に実行されるSQLがキャプチャ対象
  - キャプチャされた処理はDBA_APP_STATEMENTSビューから確認可能
  - アプリケーションPDBでSync句を使用したアプリケーションの同期を行った際に、キャプチャされたSQLは同じ順番で実行される
  - 同期処理中に問題が発生した場合は、問題を修正し、再度アプリケーションの同期処理が可能
    - エラーが発生したところから処理の実行が行われる



- キャプチャ対象のセッション
  - Begin Installとアプリケーション・オブジェクトに関する操作とEnd Installのセッションは同じでも、異なっていてもよい

```
SQL> alter pluggable database application
       MYFIRSTAPP begin install '1.0';
SQL> create table sales_data sharing=metadata
       (year number(4)…
SQL> alter pluggable database application
       MYFIRSTAPP end install '1.0';
```

  - 異なるセッションで実行されている処理がキャプチャされるのを防ぐため、モジュール名をdbms_application_info.set_moduleパッケージを使用して明示的に設定し、同じモジュール名のセッションのみキャプチャするように設定可能

モジュール指定によるキャプチャ対象セッションの制限

- アプリケーション開始時と同じサービス名、モジュール名のセッションにて実行されるSQLがキャプチャされる
  - DBMS_APPLICATION_INFO.SET_MODULEを使用してモジュール名を指定

モジュール名の設定

```
EXEC DBMS_APPLICATION_INFO.SET_MODULE('salesapp','');
ALTER PLUGGABLE DATABASE APPLICATION salesapp BEGIN INSTALL '4.2';
```

キャプチャされるSQL

```
EXEC DBMS_APPLICATION_INFO.SET_MODULE('salesapp','');
CREATE TABLE postalcodes SHARING=EXTENDED DATA
    (code         VARCHAR2(7),
     country_id   NUMBER,
     place_name   VARCHAR2(20));
```

```
EXEC DBMS_APPLICATION_INFO.SET_MODULE('salesapp','');
ALTER PLUGGABLE DATABASE APPLICATION salesapp END INSTALL '4.2';
```

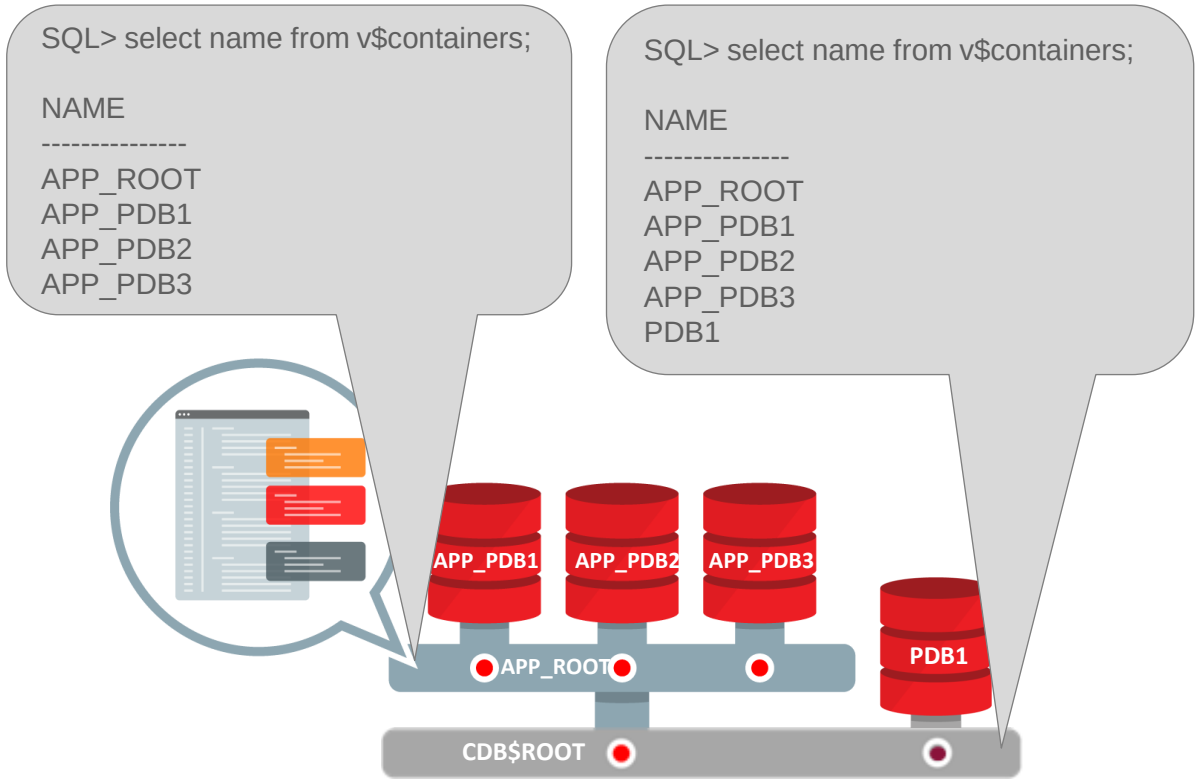
アプリケーション・コンテナ内のデータの管理

- 共通データの反映のタイミング
  - Sharing=Data、Extended Dataで定義されるテーブルに対してアプリケーション・ルートでデータの挿入、更新、削除が行われた場合、トランザクションが完了した時点でアプリケーションPDBから最新のデータを参照可能
  - アプリケーション外(Begin Install とEnd Installの外)でもデータ操作は可能だが、キャプチャが行われず、同期の対象外となる
    - アプリケーション・ルート・レプリカを使用時は注意
- Data Pumpを使用したデータの挿入
  - アプリケーションとしての実行は不可
  - アプリケーション外では実行可能 - 投入データは同期の対象にならない
- SQL*Loaderを使用したデータの挿入
  - アプリケーション内、アプリケーション外の双方で実行可能
  - アプリケーション外での実行は、投入データは同期の対象にならない
  - 従来型パス・ロードの場合、アプリケーションとしてキャプチャされ、同期の対象となる
    - アプリケーション内で実行しキャプチャさせる場合、モジュール名に 'SQL Loader Convetional Path Load' を指定
  - ダイレクト・パス・ロードの場合、SQLがキャプチャされないため、同期の対象にならない

アプリケーション・コンテナの管理

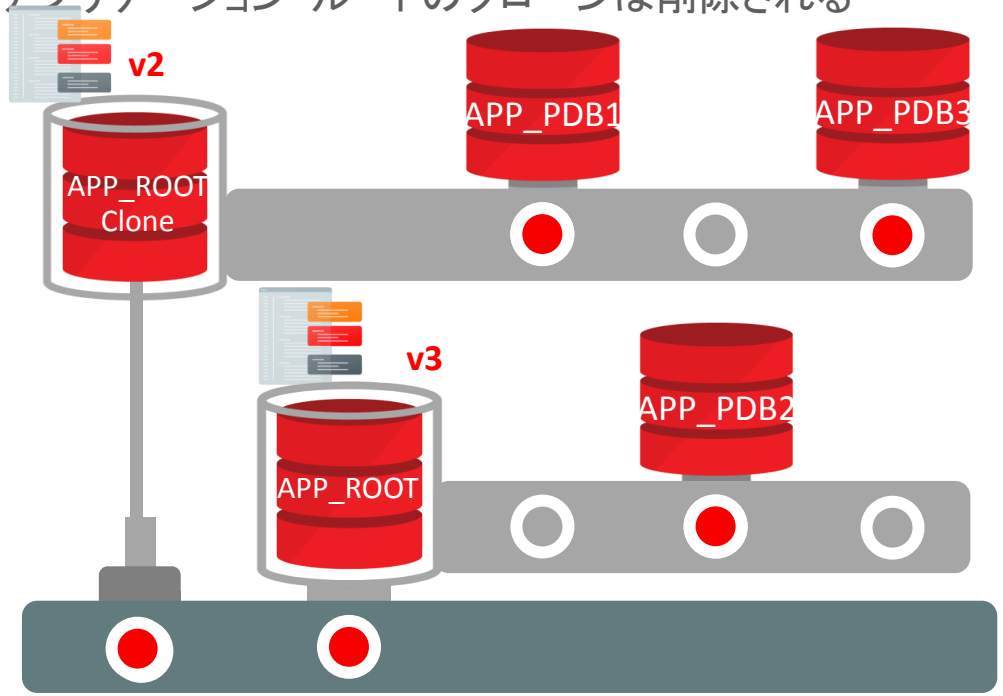
- アプリケーション・ルートからは同じアプリケーション・コンテナ内のアプリケーションPDBのみを参照、管理
- アプリケーション・コンテナ内では同じキャラクタ・セットのみ利用可能
- アプリケーション・ルートをクローズすると同じアプリケーション・コンテナ内のアプリケーションPDBもクローズする
  - CDB$ROOTでの動作と同様
- アプリケーション・ルートを削除する場合、同じアプリケーション・コンテナ内の全てのアプリケーションPDBが削除されている必要がある

- アプリケーション・ルートでALL句を使用してアプリケーション・コンテナ内のアプリケーションPDB全体に対するオープン、クローズ操作が可能



アプリケーション・コンテナの管理

- アプリケーションのアップグレード、アンインストールを行う場合、内部的にアプリケーション・ルートのクローンが作成される
  - アプリケーション・ルートのクローンの名前はシステムで自動で命名される
  - クローンでは以前のバージョンのアプリケーションの状態が維持される
  - アプリケーションのアップグレード/アンインストールはアプリケーションPDBごとに同期(SYNC)を行う
  - 既存のアプリケーションPDBはクローンの情報を参照し、同期によりアプリケーションが完了するとアップグレードされたアプリケーション・ルートの情報を参照するようになる
- 特定のバージョン以下のアプリケーションの互換性を確認
  - ALTER PLUGGABLE DATABASE APPLICATION <アプリケーション名> SET COMPATIBILITY VERSION <バージョン|CURRENT >;
  - 全てのアプリケーションPDBがクローンを参照していなければアプリケーション・ルートのクローンは削除される



共通オブジェクトのクエリー

- アプリケーション・コンテナ内のアプリケーションPDBの表、ビューを横断して検索対象とするクエリーはアプリケーション・ルートから実行
- CONTAINERS句を使用

```
SQL> select year, con$name, sum(revenue)
     from containers(sales_data)
     group by year, con$name
     order by 1, 2;
```

- 同じSQLをアプリケーションPDBから実行した場合は、結果はそのPDBのデータのみに限定される

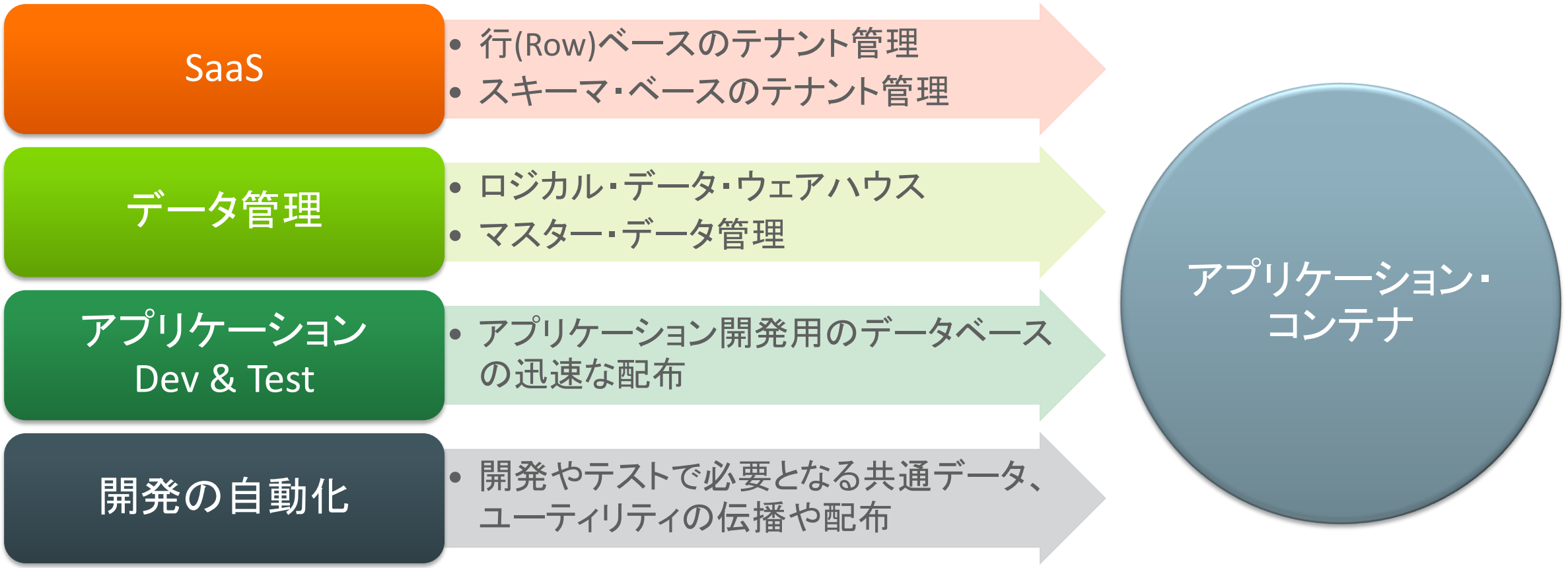
- Containers_defaultを有効にすることで、対象の表、ビューへのクエリーに対して常にCONTAINERS句でラップされるように指定可能

```
SQL> ALTER TABLE sales_data
     ENABLE CONTAINERS_DEFAULT;
```

  - 共通オブジェクトに対する変更となるため、ALTER PLUGGABLE DATABASE APPLICATION ... BEGIN文とALTER PLUGGABLE DATABASE APPLICATION ... END文の間で実行

```
SQL> select year, con$name, sum(revenue)
     from sales_data
     group by year, con$name
     order by 1, 2;
```

アプリケーション・コンテナ: ユースケース

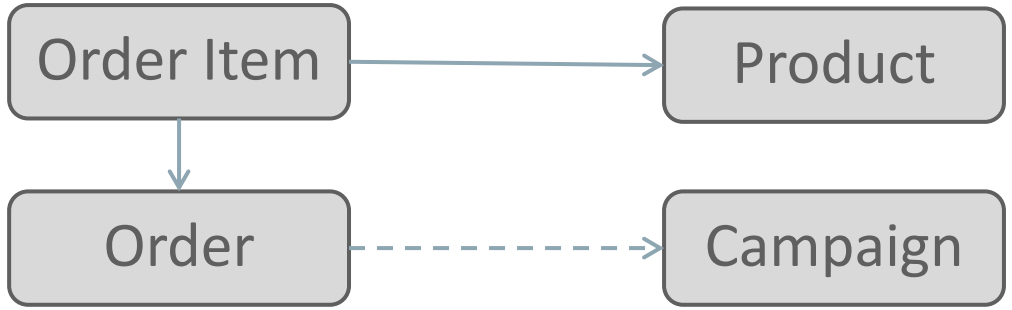


アプリケーション・コンテナの実装例 *pocoStore*:複数拠点で営業活動を行う小売業

pocoStore : 販売店管理アプリケーション

- 設定
  - 日本の主要都市に支店を持つ小売業
  - 販売している商品は基本的には全店で同じであるが、各支店で特徴的な商品も扱う
  - キャンペーンに基づく売り上げを分析
  - 販売店管理アプリケーションを開発し、ビジネス・ニーズに合わせてメンテナンスを行う
  - 各支店のデータは個別に管理
  - ビジネスの拡大のため、支店を増やす予定

pocoStore データ・モデル(一部)



- Product
  - 標準的な商品構成
  - 各拠点ごとに独自の商品が加わる可能性もある
- Campaign
  - 全てのキャンペーンは中央で管理

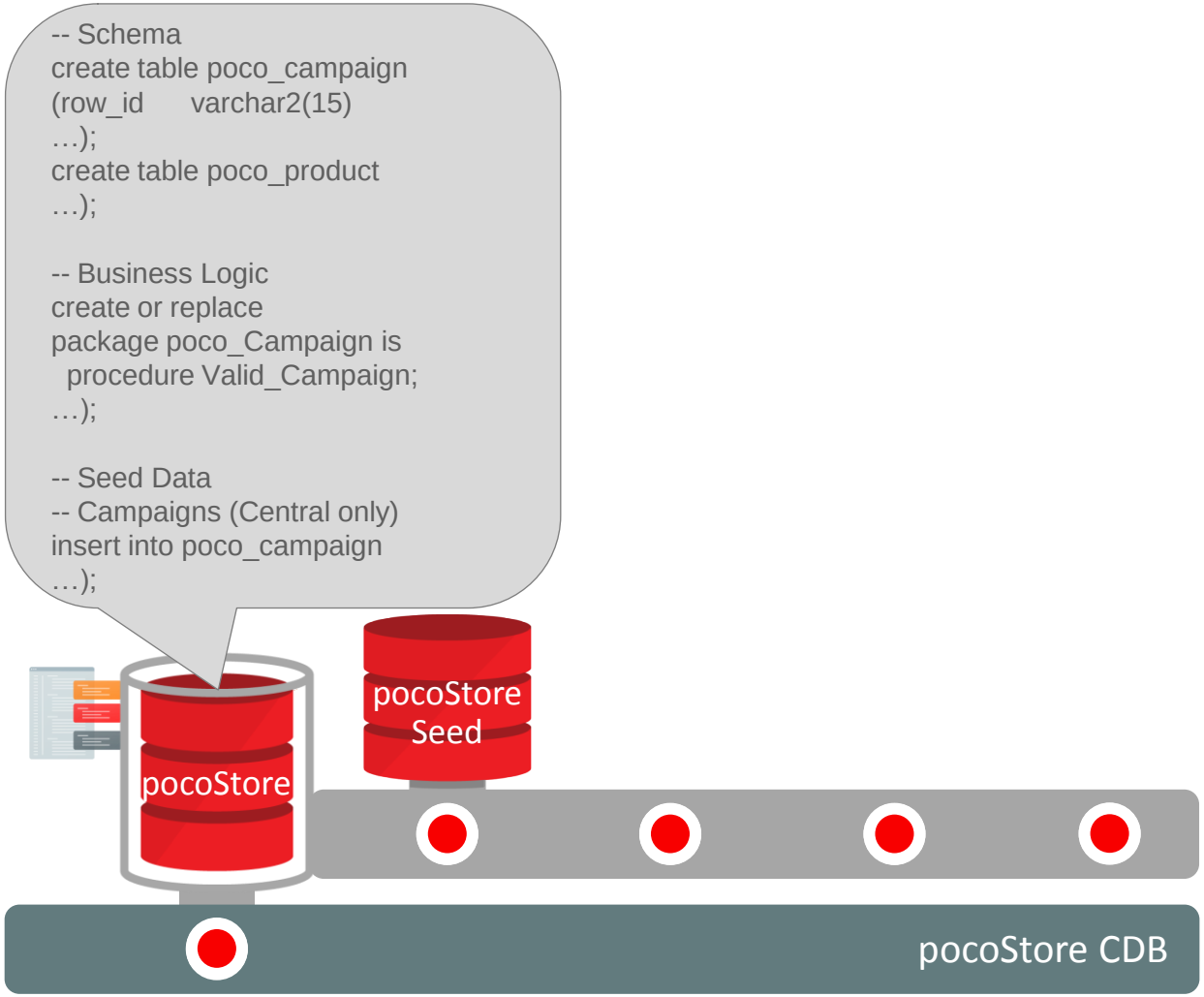
アプリケーション・コンテナの実装例
pocoStore: DBオブジェクト(一部)


```
-- Schema

create table poco_campaign
(row_id      varchar2(15) not null
,name        varchar2(30) not null
…);

create table poco_product
(row_id      varchar2(15) not null
,name        varchar2(30) not null
…);

create table poco_order
(row_id      varchar2(15) not null
,campaign_id varchar2(15)
…);

create table poco_order_item
(row_id      varchar2(15) not null
,order_id    varchar2(15) not null
,prod_id     varchar2(15) not null
…);
```

```
-- Business Logic

create or replace
package poco_Campaign is
  procedure Valid_Campaign;
  procedure Special_Discount;
…);

create or replace
package poco_Sales_Tax is
  procedure Consumption_Tax;
  procedure Tax_Exemption;
…);
```

```
-- Seed Data

-- Campaigns (Central only)
insert into poco_campaign values
 (‘1’,’Golden Week 2016’);

insert into poco_campaign values
 (‘2’, ’Silver Week 2016’);

insert into poco_campaign values
 (‘3’, ’Christmas 2016’);

-- Products (Central + local)
insert into poco_product values
 (‘1’, ’Tornado Twisted’);

insert into poco_product values
 (‘2’, ’Candy Shake’);

insert into poco_product values
 (‘3’, ’Duke Float’);

insert into poco_product values
 (‘4’, ’Shinkansen Strawberry Scone’);
```

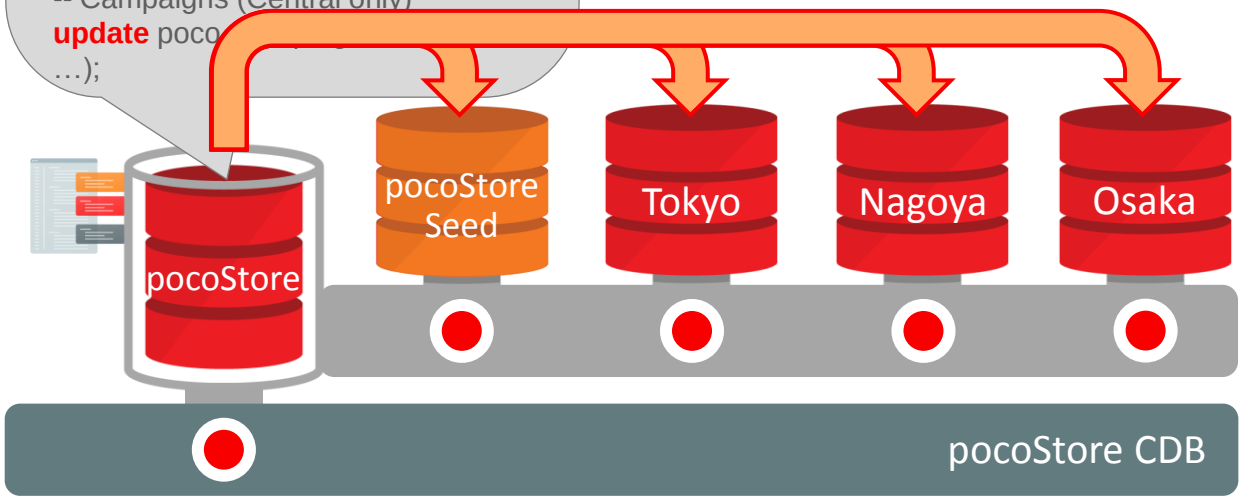
アプリケーション・コンテナ
アプリケーションのマルチテナント環境の構築



- プラガブル・データベース*pocoStore*をアプリケーション・コンテナとして作成
- *pocoStore*アプリケーションのマスター定義をアプリケーション・ルートにインストール
- *pocoStore*シードを作成
- 新規フランチャイズ・テナント用PDBを*pocoStore*シードを使って提供

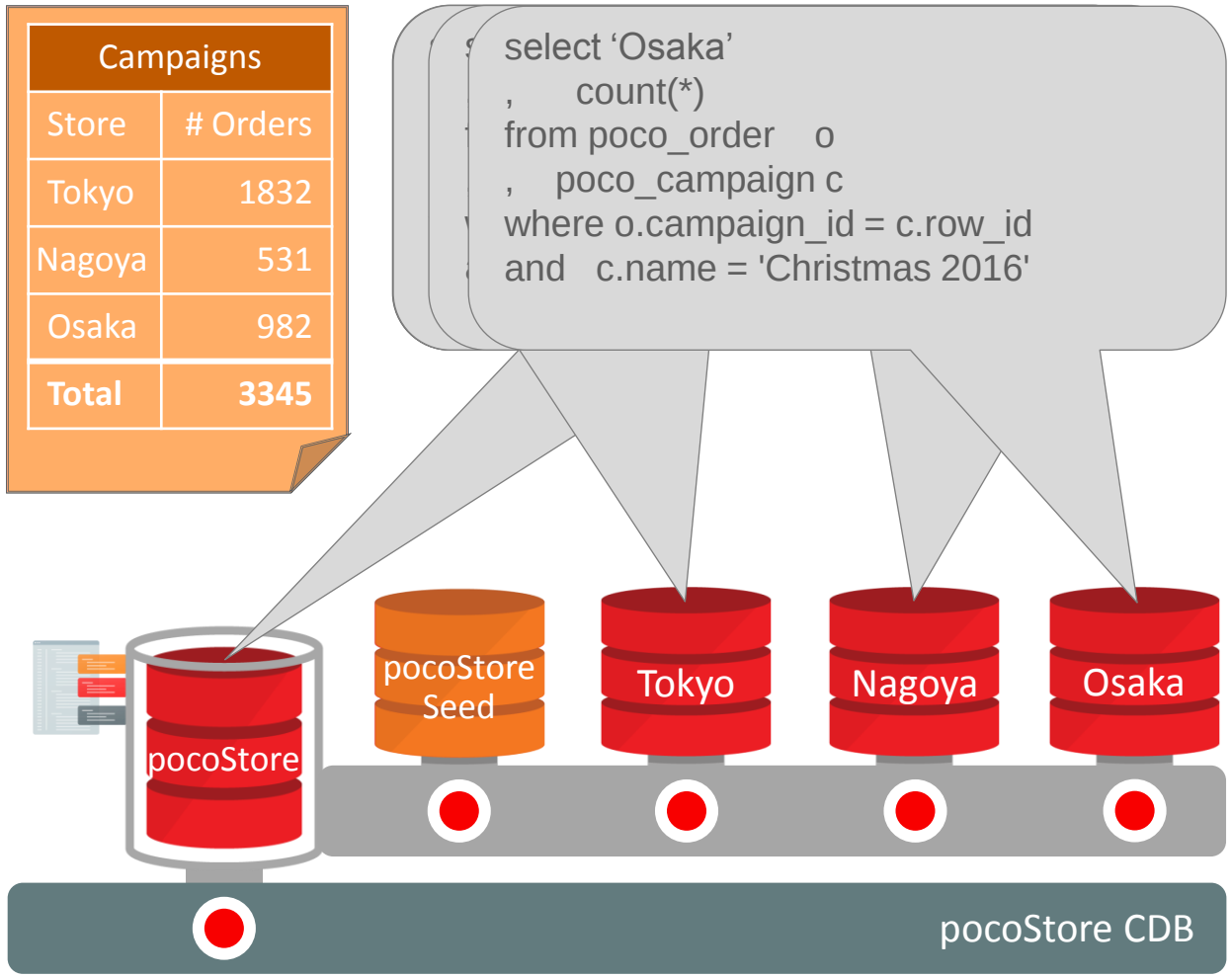
アプリケーション・コンテナ アップグレード

```
-- Schema upgrade
alter table poco_campaign add
(start_date  date
…);
alter table poco_product
…);

-- Business Logic
create or replace
package poco_Campaign is
  procedure Valid_Campaign;
…);

-- Seed Data
-- Campaigns (Central only)
update poco
…);
```



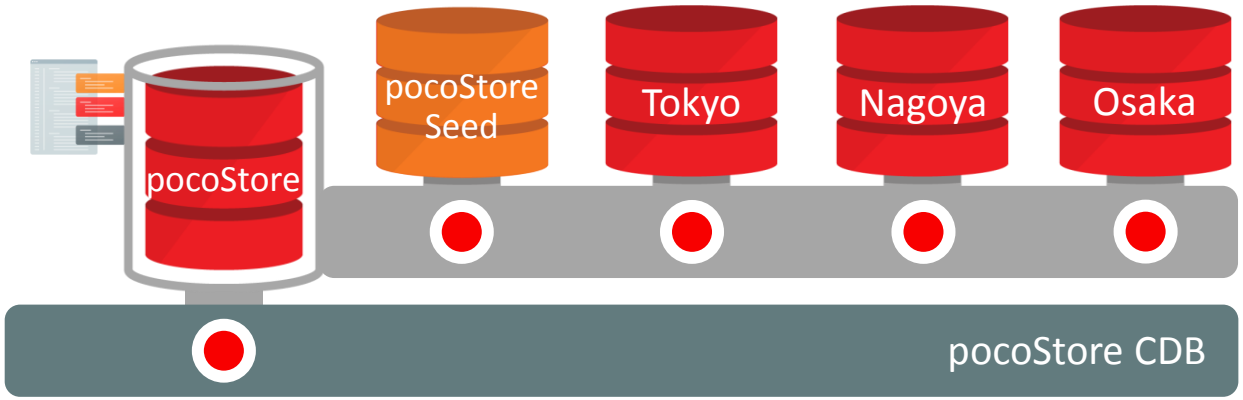
- アプリケーション・ルート内でアップグレードを実施
- アプリケーションPDBはアプリケーション・ルートとシンプルに同期が可能
  - フランチャイズ・テナントはそれぞれのスケジュールに合わせて同期できる

アプリケーション・コンテナ
コンテナ間の集約



- 各テナントのデータ分析をアプリケーション・コンテナを利用して実施
  - 従来の手法:
    - 各フランチャイズ・テナントごとに分析対象のデータを取得
    - 同じSQL文を各テナントで実行し、全テナントのデータをスプレッドシートなどを利用して集計処理を実施
- ソリューション:
  SQL文で`Containers()`句を利用
  - アプリケーション・ルートで単一のSQL文を実行するのみ
  - 各PDBで再帰的に実行される
  - アプリケーション・ルート内で集約される

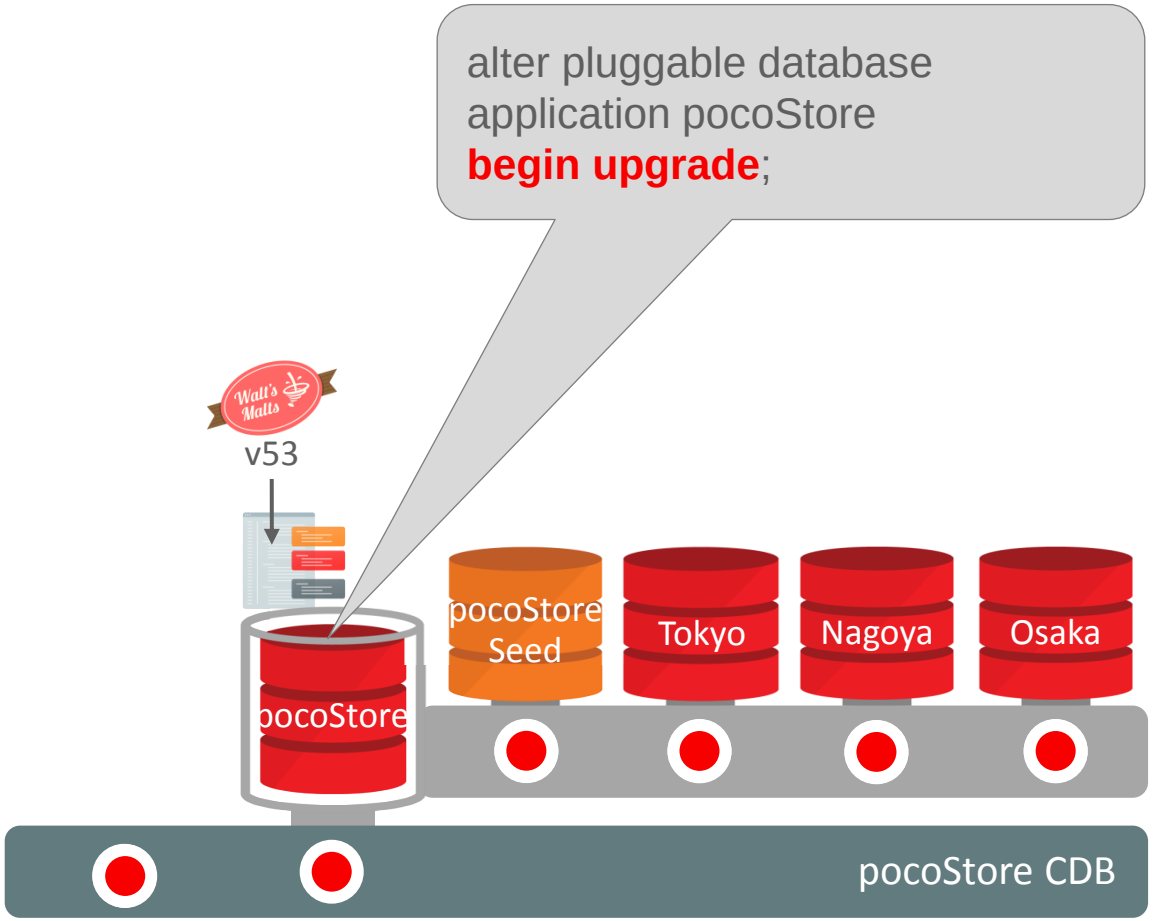
アプリケーション・コンテナ – 優位性



- 優位性
  - テナント間の分離性を維持
  - 新規テナントのセットアップをごく短時間で実施可能(マスターのクローン)
  - 小規模のサーバー環境でも多くのテナントを対応できる
  - 多数のテナントの管理を典型的な***DBA***のタスクとして、まとめて行える
- ***さらなる優位性***
  - ***アプリケーション*** 管理の集中化
  - シンプルかつ強力なテナント間の集約処理

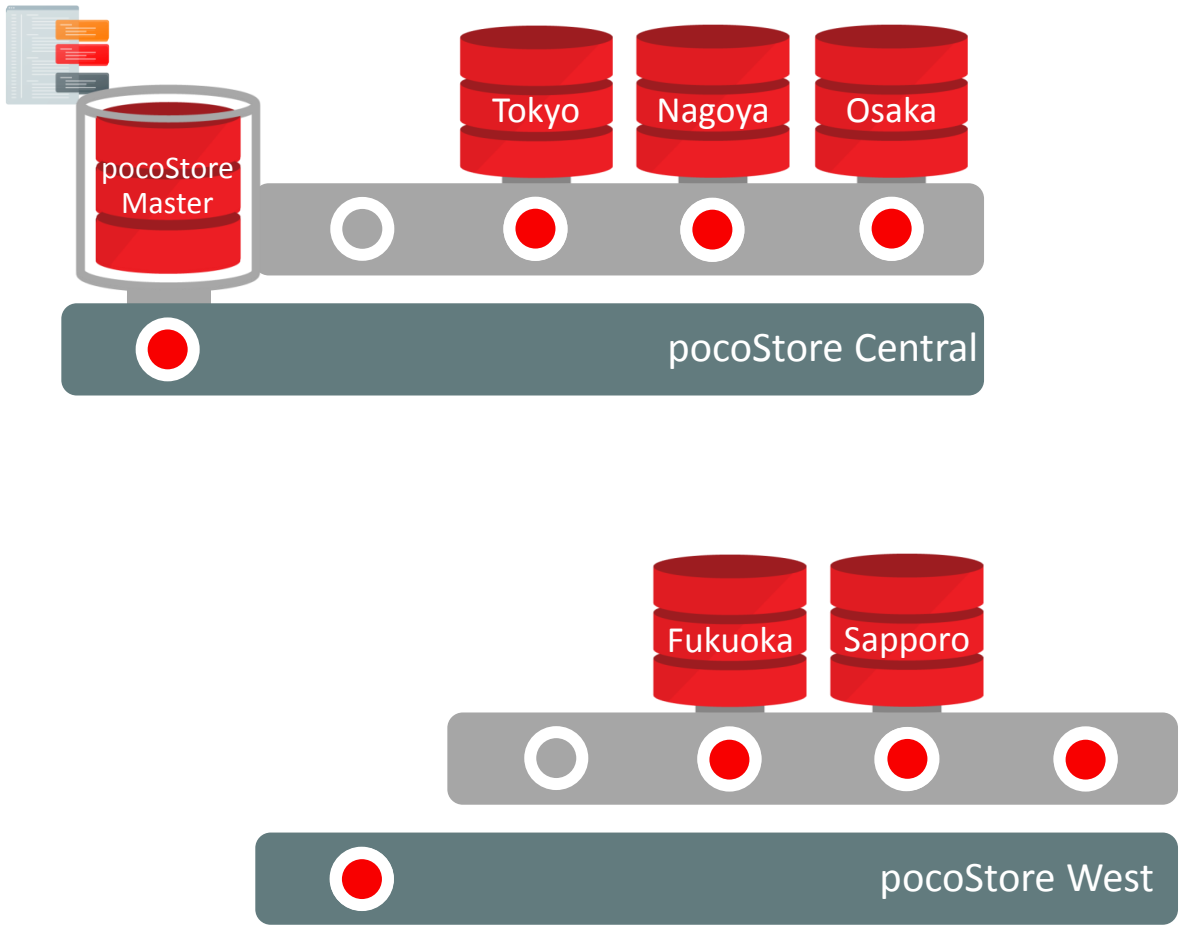
アプリケーションのアップグレード

テナントに影響を与えずにアプリケーション・ルートをアップグレード



1. アプリケーション・ルートでアップグレードの開始
   - アプリケーション・ルートのクローンが自動的に作成される
   - スナップショット・クローンとして利用も可能
   - アプリケーションPDBは、メタデータなどのマスター定義情報はアプリケーション・ルートのクローンから提供される
2. アプリケーション・ルートでアプリケーションのアップグレードを実施
   - *pocoStore*アプリケーションのマスター定義の完全アップグレード
   - アプリケーションPDBには影響が及ばない
3. アプリケーションPDBはアプリケーション・ルートの状態に同期することでアップグレードを実施
   - アップグレード・スクリプトをアプリケーションPDB上で実行
     - ローカル・データへの変更を適用
     - メタデータも適宜変更される
   - アプリケーションPDBへのメタデータなどのマスター定義情報の提供は再度アプリケーション・ルートから行われるように戻る

アプリケーション・コンテナの複数の環境間の連携



- 複数サーバーに渡ってアプリケーション・コンテナを稼働させるケース

シナリオ:

  - テナント追加により、別サーバー上で新たにCDBを構成
  - 各CDB上にアプリケーション・ルートの完全なレプリカの作成が必要
  - アップグレードに対応するため、ルート・レプリカ間は同期された状態であることが求められる
  - 同期処理は同じCDB上のPDB間でのみ実施可能
  - …*リモート*で稼働するアプリケーション・ルートのレプリカに対して、プロキシとなるような*ローカル*PDBを作成が求められる

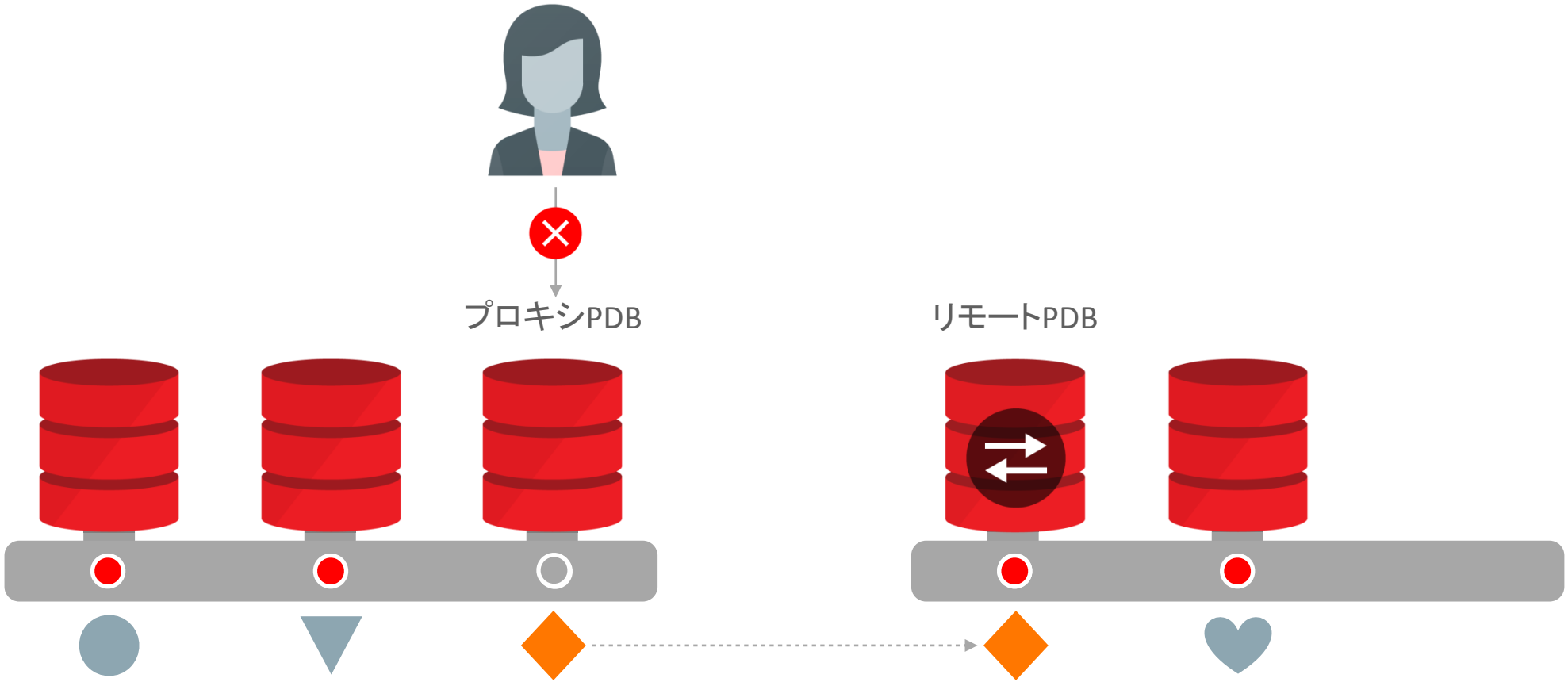
- ***プロキシPDB***を導入

4.位置透過性を実現する機能

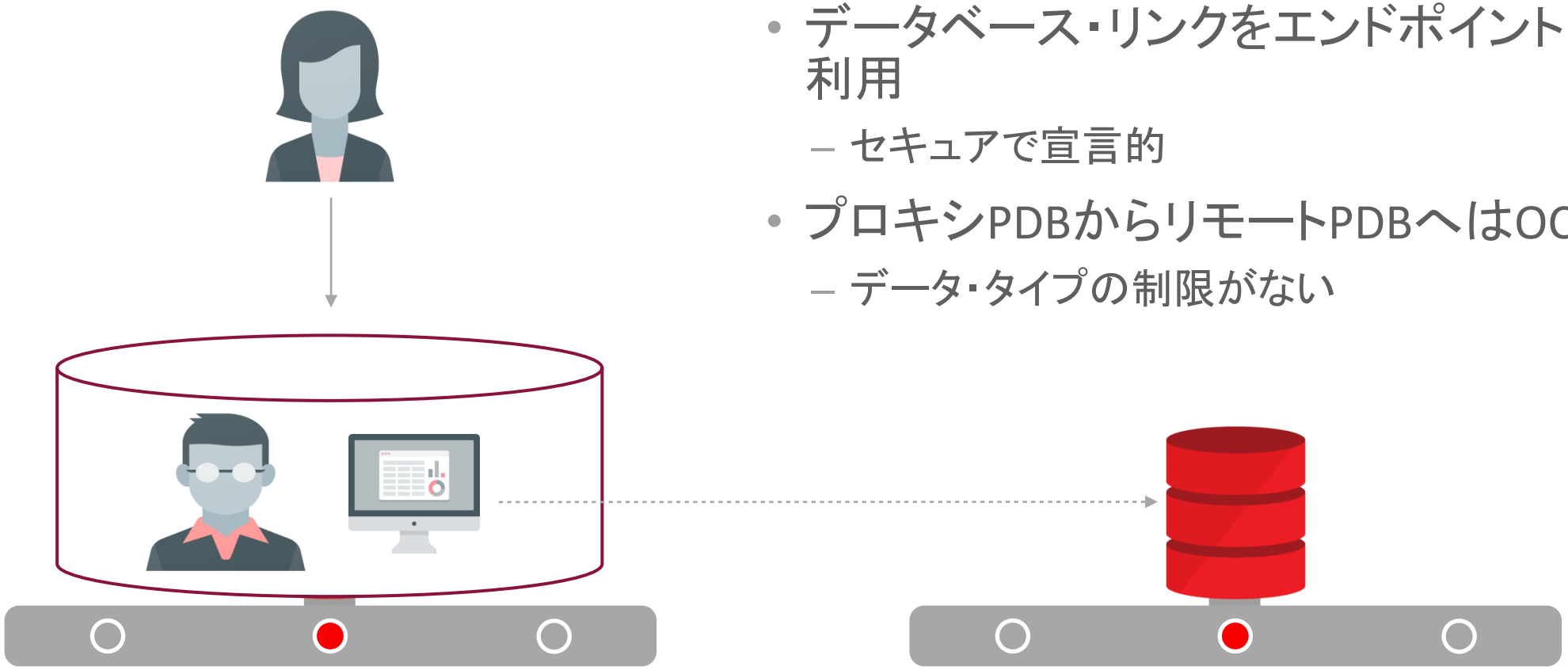
ハイブリッド・クラウドを実現

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



プロキシPDB



- データベース・リンクをエンドポイントとして利用
  - セキュアで宣言的
- プロキシPDBからリモートPDBへはOCI接続
  - データ・タイプの制限がない

プロキシPDB: 構成

- プロキシPDBを参照するPDBと同じCDB上に作成も可能
- データベース・リンクはプロキシPDBが作成されるCDBから、アプリケーション・コンテナの場合は、アプリケーション・ルートから接続がはられるように作成する
- プロキシPDB作成時は参照するPDBはRead Writeモードでオープンしている状態にする
- プロキシPDBによるリソースの消費は大きくはない

- プロキシPDBの構成ファイル
  - SYSTEM表領域
    - 参照するPDBのSYSTEM表領域の完全なコピー
  - SYSAUX表領域
    - 参照するPDBのSYSAUX表領域の定義情報のみのコピー
  - ユーザーが作成した表領域は対象外
- データベース・リンクはエンドポイントの管理用途にのみ使用
  - すべてのSQLはリモートPDBへ渡される
    - ALTER PLUGGABLE DATABASE文とALTER DATABASE文は例外
  - すべての結果セットがプロキシPDBへ返される

プロキシPDB – 設定と実行手順

プロキシPDBが参照するPDBが稼働するCDBの構成を確認

- アーカイブ・ログ・モード
- ローカルUNDOモード

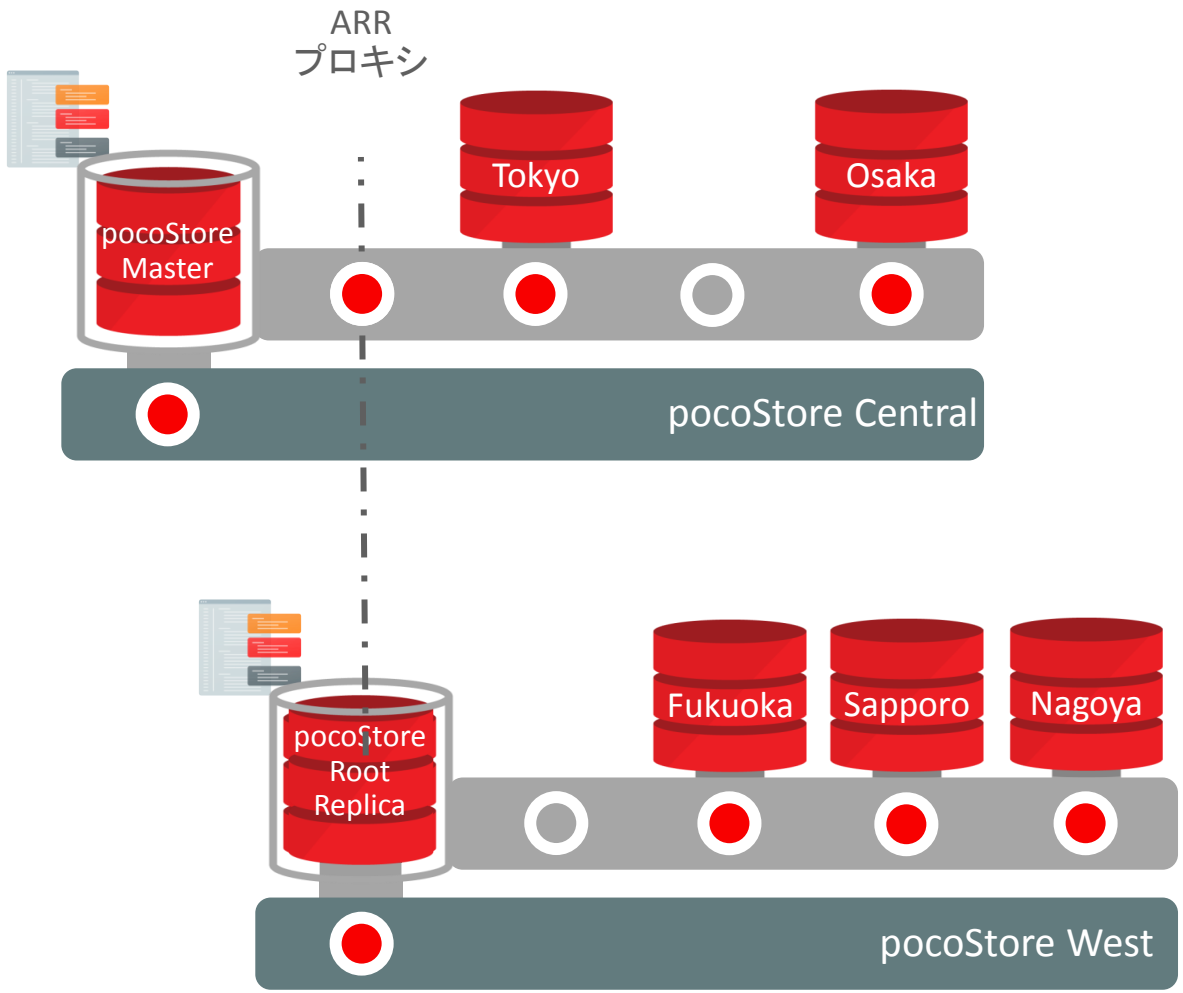
プロキシPDBを作成するCDB$ROOTから参照するPDBに対するデータベース・リンクを作成

```
SQL> create 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

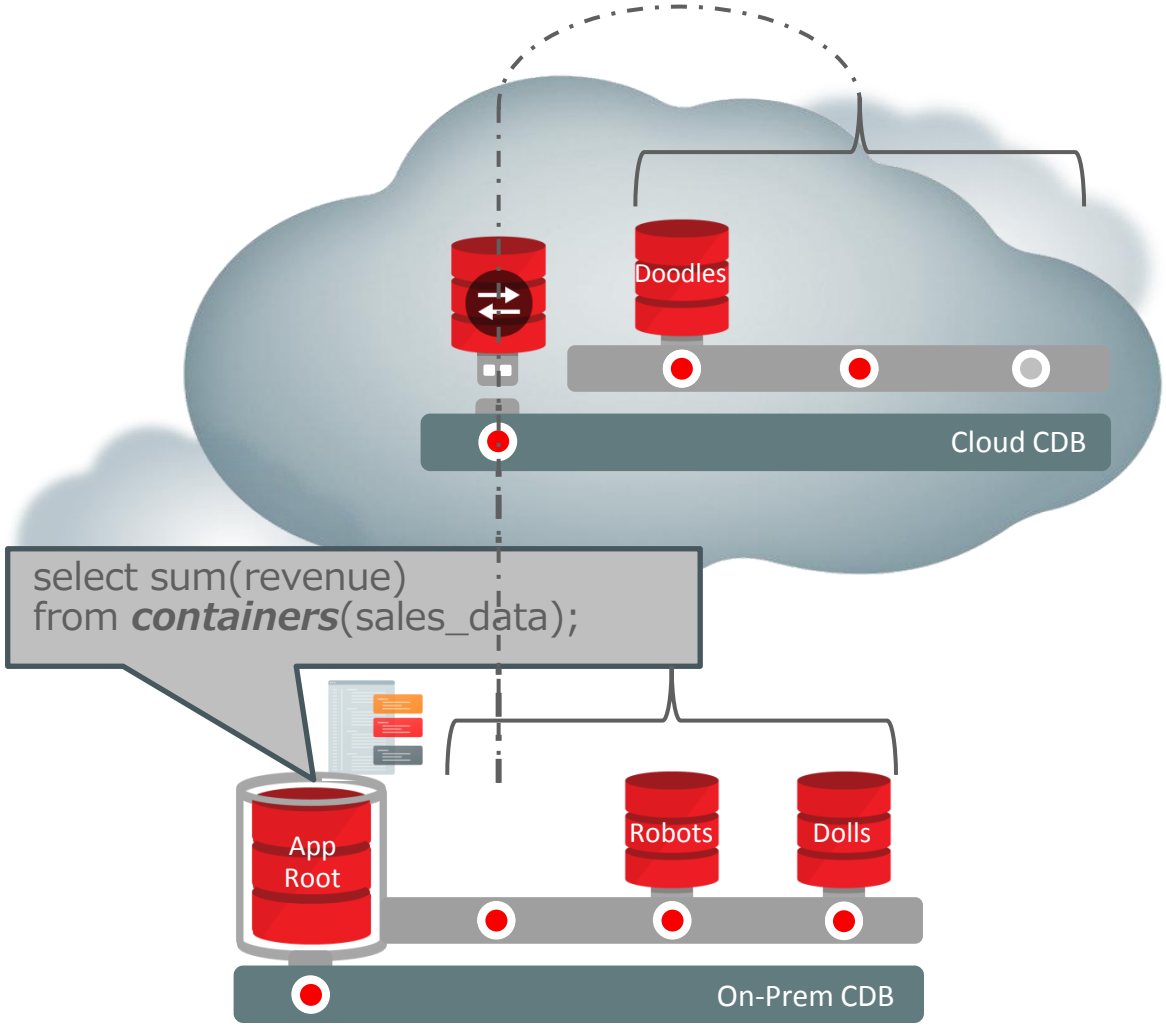
```
SQL> create pluggable database oe_proxy as proxy from oe@dblink;
```

アプリケーション・コンテナ
プロキシPDBを通じてルート・レプリカを同期



- メタデータと共通データの共有と伝搬
- アプリケーション・ルートの変更をリモートのCDBにあるアプリケーション・ルート・レプリカに同期する

プロキシDPBによるハイブリッド環境の実現
オンプレミスとクラウドをシームレスに連携

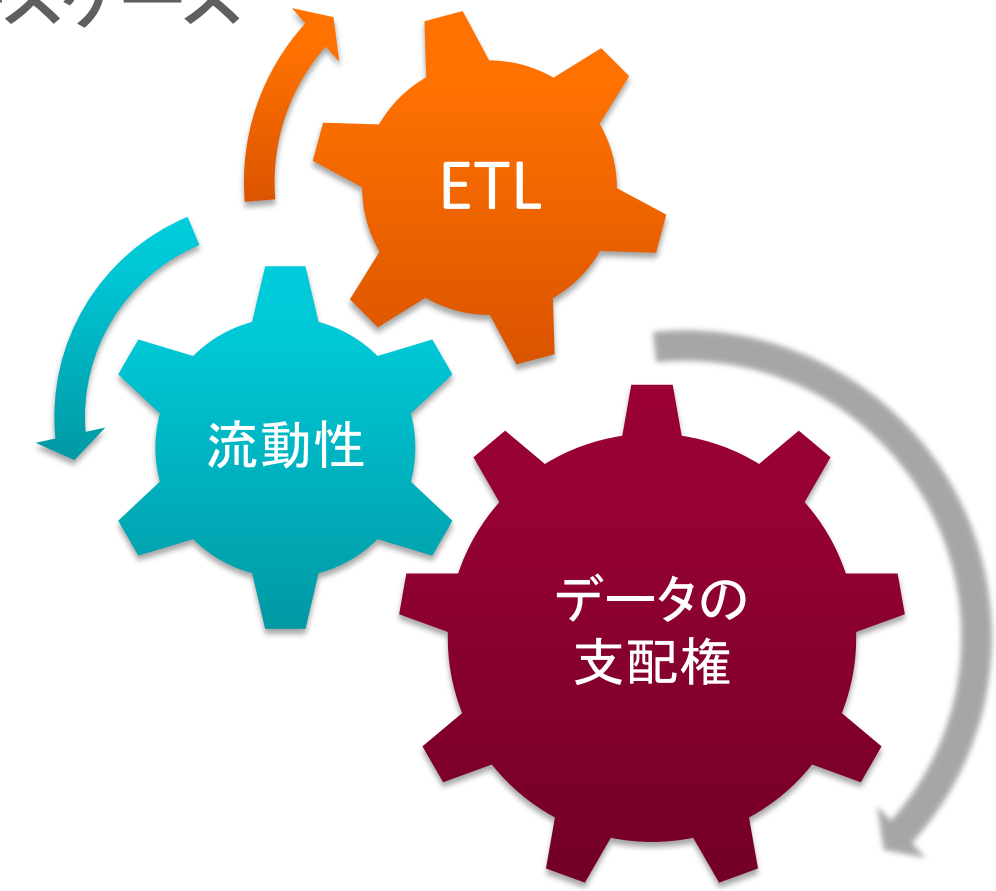


1. **オンプレミスのCDB:**
   1. アプリケーション・コンテナApp_Rootを作成
   2. アプリケーションPDBとしてRobotsとDollsを作成
2. **クラウド上のCDB:**
   1. アプリケーション・コンテナApp_RRを作成
   2. アプリケーションPDBとしてDoodlesを作成
3. **オンプレミスのCDB :**
   1. SQL> create PDB ARR_Proxy as proxy from App_RR@Link;
   2. Robots、DollsおよびDoodlesのPDBをまたがった集計を行うアプリケーション・コードを書く
4. **負荷分散:**
   1. PDB DollsをApp_RootからApp_RRに再配置
   2. アプリケーション・コードは変更なしで実行可能
      これは永続的な**位置透過性**の一例

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
- アプリケーション・ルート内でデータベース・プロパティCONTAINER_MAPにマップ・オブジェクトを指定
  - ALTER DATABASE SET CONTAINER_MAP ='map_table_schema.map_table_name';

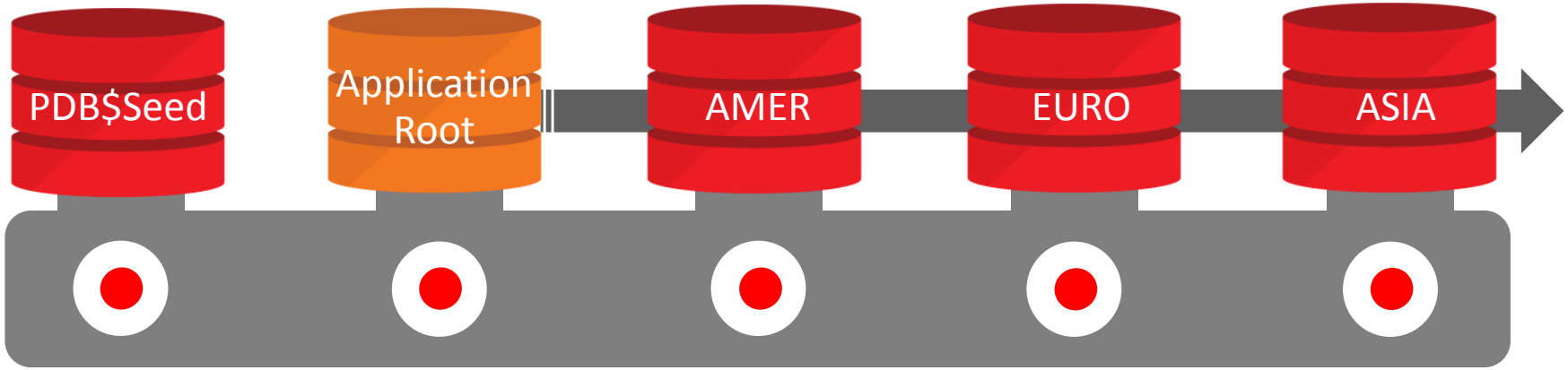
ユースケース



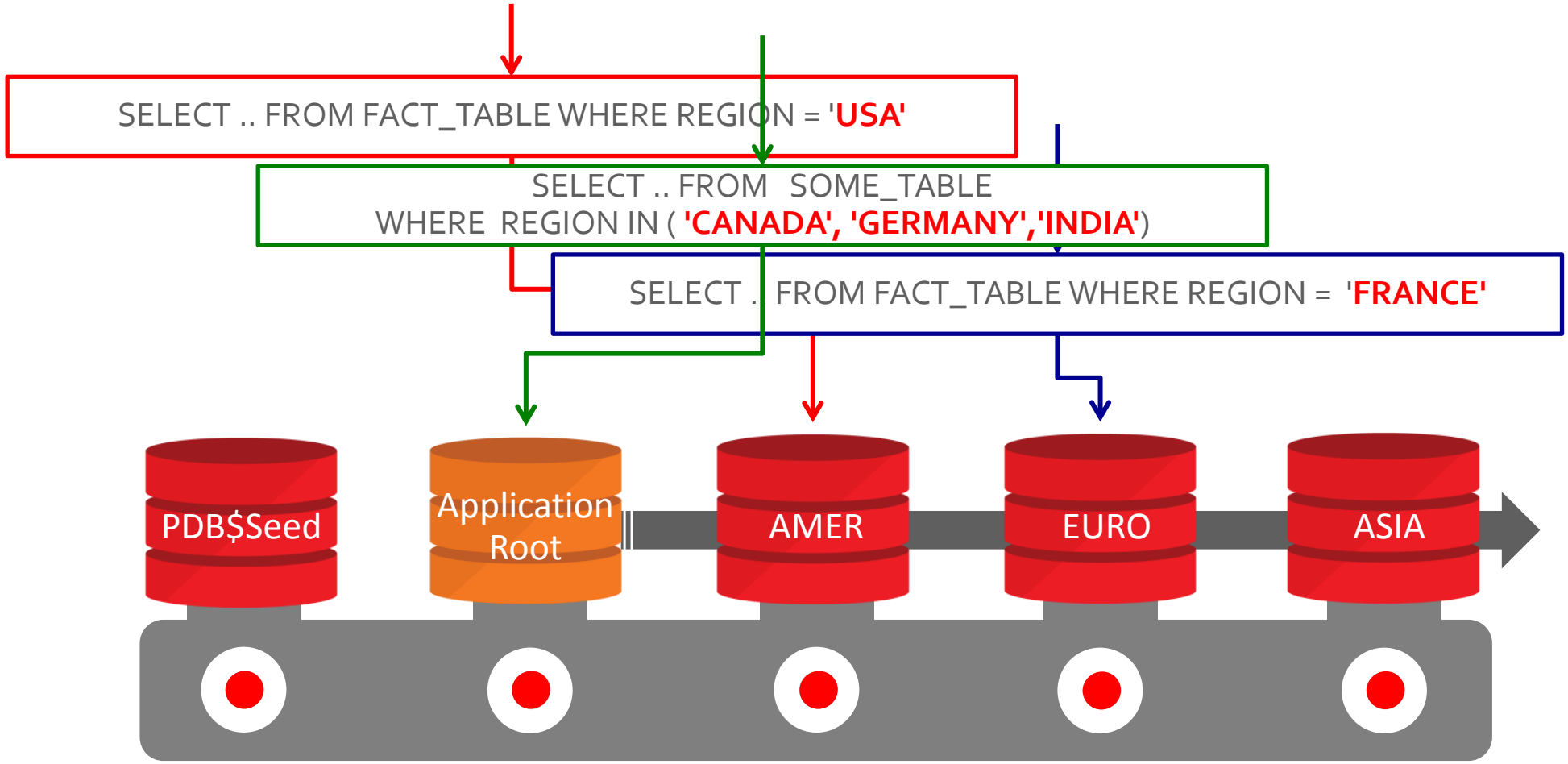
コンテナ・マップ: 設定

```
Container_Map = <schema>.conmap

CREATE TABLE CONMAP
( COLUMNS ..,
  REGION VARCHAR2..)
PARTITION BY LIST (REGION)
     (PARTITION AMER VALUES ('USA','MEXICO','CANADA'),
      PARTITION EURO VALUES ('UK', 'FRANCE','GERMANY'),
      PARTITION ASIA VALUES  ('INDIA', 'CHINA','JAPAN'))
```

使用可能なパーティション手法
- レンジ
- リスト
- ハッシュ

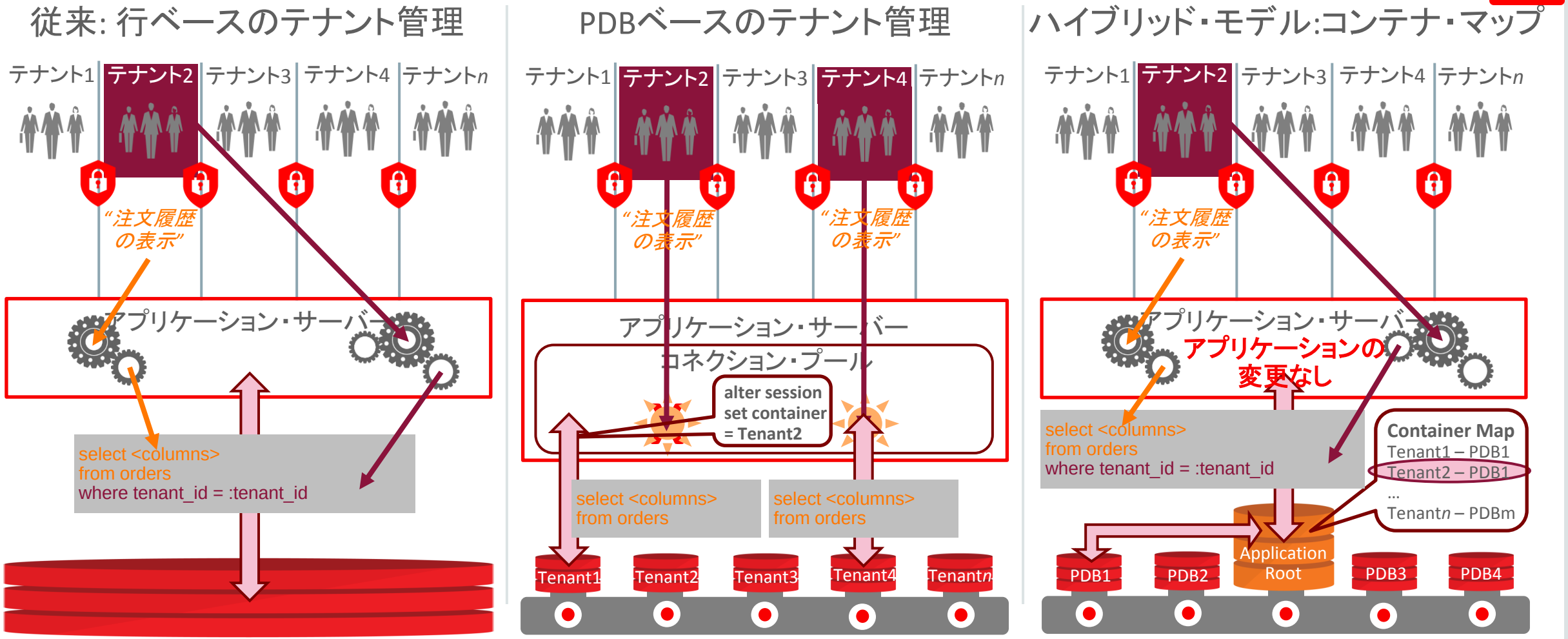


コンテナ・マップ: クエリーを適切にルーティング



SaaSアプリケーションのコンテナ・マップの活用

NEW IN 12.2



コンテナ・マップ:設定時の留意点

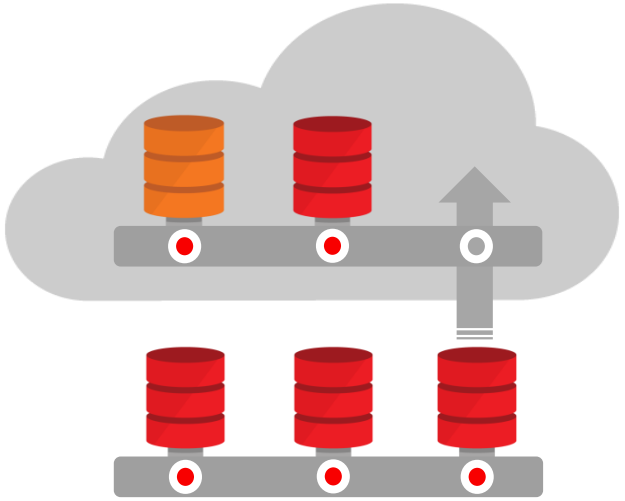
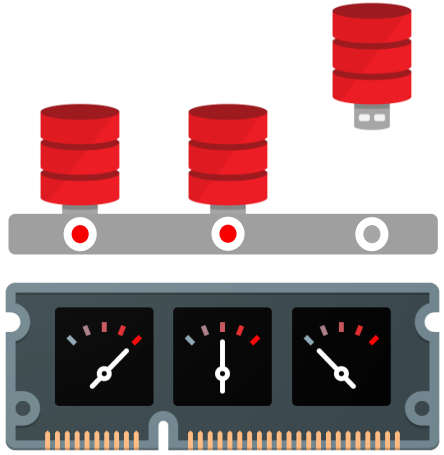
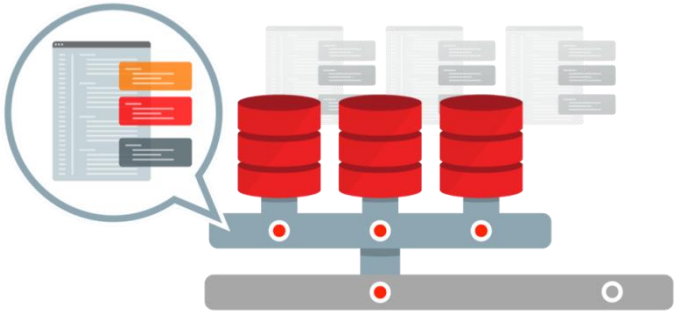
- コンテナ・マップを使用した処理
  - SELECT文が対象 / DML文は対象外
- コンテナ・マップはアプリケーション・コンテナに1つのみ指定可能
- コンテナ・マップを利用したクエリーの対照表
  - 対照表はメタデータ・リンク (Sharing=Metadata)の表
  - CONTAINER_MAPプロパティを有効化
    - ALTER TABLE <schema>.<表 > ENABLE CONTAINER_MAP;
  - CONTAINERS_DEFAULTプロパティを有効化
    - ALTER TABLE <schema>.<表 > ENABLE CONTAINERS_DEFAULT;
  - データはアプリケーションPDB内にロードされている必要がある
- マップ・オブジェクト
  - 実際のデータ配置に合わせてパーティションを定義
  - アプリケーションPDBの構成を変更した場合は、マップ・オブジェクトも更新を行う
    - マップ・オブジェクトは自動でメンテナンスは行われない
  - ALTER DATABASE SET CONTAINER_MAP ='<マップ・オブジェクト名>';を実行時は関連のアプリケーションPDBが作成済みである必要がある

5. まとめ

12c R2におけるOracle Multitenantの革新

Oracle Multitenant - Oracle Database 12*c*

NEW IN 12.2

アジリティ	クラウド規模の運用	Software as a Service
		

Release 12.1	• 迅速なクローニング • PDBのアンプラグ / プラグ	• 一括管理 • CPUとI/O管理	• SaaSアーキテクチャ • アプリケーション・コードの変更不要
Release 12.2	• ホット・クローニング / リフレッシュ • オンライン再配置	• CDBあたり4k PDBs • メモリー管理	• 共有アプリケーション・オブジェクト • 位置透過性の提供

Announcing Exadata Express Cloud Service

簡単に使えて、低コストな12c R2で動作するDatabase Cloud Service

- Exadata上で動作するオプション込みの#1 Database
- Oracleが管理
- 低コスト：$175/月から始められる
  - ユーザーごとにPDBを提供

リファレンス
マニュアル・ドキュメント

- Oracle Database概要 12cリリース2 (12.2)
  - 第VI部 マルチテナント・アーキテクチャ
    http://docs.oracle.com/cd/E82638_01/CNCPT/multitenant-architecture.htm
- Oracle Database管理者ガイド 12cリリース2 (12.2)
  - 全般
    http://docs.oracle.com/cd/E82638_01/ADMIN/toc.htm
- Oracle Databaseセキュリティ・ガイド 12cリリース2 (12.2)
  - 4 権限とロール認可の構成
    http://docs.oracle.com/cd/E82638_01/DBSEG/configuring-privilege-and-role-authorization.htm

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®