IT資格道場

ORACLE MASTER Gold DBA Oracle AI Database 26ai 学習用テキスト

管理操作を実行して覚える —— 読んで理解する1冊目

無料サンプル(第1章まで)

Oracle Japan「Oracle AI Database 26ai Administration Professional 1Z0-183-JPN」試験内容チェックリスト(5分野・35細目)準拠
2026年11月版

SAMPLE

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

SAMPLE

目次

この教材群の使い方(3点セット)

この試験が問うもの

第1章 マルチテナント・アーキテクチャの管理 [1.1/1.2/1.3/1.4/1.5]

- CDB を作る 2 つの道 [1.1]
- シードから PDB を作る [1.1]
- アプリケーション・コンテナの構成と共通オブジェクト [1.2]
- アプリケーション・コンテナと PDB を作る [1.2]
- アプリケーションのインストール・アップグレード・パッチ・同期 [1.2]
- PDB をクローンする [1.3]
- アンプラグとプラグイン [1.3]
- PDB の再配置とスナップショット [1.3]
- CDB と PDB の起動・停止とオープン・モード [1.3]
- PDB のサービスと接続 [1.3]
- 初期化パラメータの範囲 [1.3]
- ローカル UNDO と共有 UNDO [1.3]
- CDB と PDB のリソース管理 [1.3]
- 共通ユーザー・ローカル・ユーザー・ロール・権限 [1.4]
- PDB ロックダウン・プロファイル [1.4]
- 統合監査と共通・ローカルの監査方針 [1.4]
- TDE の鍵の管理：統一モードと分離モード [1.4]
- アプリケーション・コンテナの共通の方針 [1.4]
- 以前のリリースの PDB をプラグインしてアップグレードする [1.5]
- 非 CDB を PDB に変える [1.5]
- PDB の転送の使い分け [1.5]

SAMPLE

サンプルはここまでです

SAMPLE

この教材群の使い方(3点セット)

種類	役割
① 学習用テキスト	これだけで問題が解けるようになる本編
② 頻出度別ノート	読んだ後に使う。頻出順に絞った問題と解説で「解ける」に橋渡し。試験直前の最終チェックにも
③ 暗記用ノート	覚える要点だけの一問一答。空き時間の反復と用語の再確認用

使い方:

1. ① 学習用を読む
2. ② 頻出度別で解き方を掴む
3. ③ 問題演習で腕試し
4. ④ 頻出度別に戻って再確認(試験直前もここ)

暗記用は空き時間や用語の再確認に。

SAMPLE

この試験が問うもの

ORACLE MASTER Gold DBA Oracle AI Database 26ai。

- 試験番号：1Z0-183-JPN
- 試験名：Oracle AI Database 26ai Administration Professional

問うのは、Silver DBA の上の段にあたる、データベース管理者の仕事です。操作を知っているだけでなく、場面に合う手段を選べるかを見ます。

範囲は次のとおり。SQL の書き方と、Silver DBA で扱う基本の管理は含みません。

- マルチテナント・アーキテクチャの管理
- バックアップおよびリカバリ
- パフォーマンスの管理およびチューニング
- デプロイ、パッチ適用およびアップグレード
- Oracle AI Database 26ai の新機能

項目	内容
運営	Oracle Japan（試験の配信は Pearson VUE）
出題数	65問
試験時間	120分
合格ライン	60%（65問のうち39問の正解が目安）
出題形式	選択問題

SAMPLE

公式の試験内容チェックリストは5分野。このテキストも同じ順番で5章に並べました。

SAMPLE

第1章 マルチテナント・アーキテクチャの管理 [1.1/1.2/1.3/1.4/1.5]

マルチテナント・アーキテクチャは、1 つのデータベースの中に複数のコンテナを置く構成です。26ai のデータベースはすべてこの構成で、非 CDB は新しく作れません。以前のリリースの非 CDB も、アップグレードの時に PDB へ変えます。[1]

コンテナは次の種類に分かれます。CDB から上の 2 つを外すことはできません。

コンテナ	役割
CDB ルート（CDB$ROOT）	Oracle 提供のメタデータと共通ユーザーを持つ。CON_ID は 1
PDB シード（PDB$SEED）	新しい PDB を作る時の雛形。読取り専用。CON_ID は 2
PDB	利用者のデータを持つ普通のコンテナ
アプリケーション・ルート	アプリケーション・コンテナの親。共通のデータとメタデータを持つ
アプリケーション PDB	アプリケーション・ルートに差し込まれる PDB
アプリケーション・シード	アプリケーション PDB を素早く作る雛形（任意）

出力例は Oracle AI Database 26ai Free（23.26.3）で実行した結果です。Free エディションに無い機能は、返ってきた ORA- の番号をそのまま載せています。

SAMPLE

CDB を作る 2 つの道 [1.1]

CDB の作り方は DBCA と CREATE DATABASE 文の 2 つです。DBCA の動かし方は 3 つあります。[23]

動かし方	中身
対話	画面（GUI）の質問に答えて進める
サイレント	`dbca -silent -createDatabase ...` と引数で指定する。画面は出ない
レスポンス・ファイル	`ORACLE_HOME/assistants/dbca` のテンプレート `dbca.rsp` を編集し、`-responseFile` で渡す。`-silent` を省くとレスポンス・ファイル・モードになり、ファイルに書いた項目の画面も、書き漏らした項目の画面も出る

DBCA が CDB を作る時に使う主な引数は次のとおりです。

引数	意味
-templateName	元にするテンプレート（XML ファイル）。Oracle 提供は「汎用または OLTP」と「データウェアハウス」の 2 種類。自作もできる
-createAsContainerDatabase	true で CDB を作る。false は 20.3 から未サポート
-numberOfPDBs	中に作る PDB の数。既定は 0
-pdbName	PDB の基本名。数が 2 以上なら名前の後ろに番号が付く。数が 0 を超えるなら必須
-pdbAdminPassword	PDB 管理者のパスワード
-useLocalUndoForPDBs	true で PDB にローカル UNDO を使う

SAMPLE

引数	意味
-pdbStorageMAXSizeInMB	PDB が使う記憶域の上限（MB）

```
dbca -silent -createDatabase -templateName General_Purpose.dbc \
  -gdbName oradb.example.com -sid oradb -characterSet AL32UTF8 \
  -createAsContainerDatabase true -numberOfPDBs 2 -pdbName sales \
  -pdbAdminPassword ******** -useLocalUndoForPDBs true
```

DBCA には PDB を扱うコマンドもあります。`-createPluggableDatabase` は新規作成・クローン・XML や .pdb からの作成に使います。`-deletePluggableDatabase` は削除、`-createPDBSnapshot` はスナップショットの作成、`-relocatePDB` は再配置です。[24]

CDB では、RMAN が制御ファイルの自動バックアップを既定で有効にします。PDB の特定時点リカバリ（PITR）は RMAN で行います。[5]

CREATE DATABASE 文で作る手順は、初期化パラメータ・UNDO・シードのファイル名の 3 つを決めてから実行します。この文が作るコンテナは CDB ルートと PDB$SEED だけです。それ以外のコンテナは手で作ります。[5]

SAMPLE

```
CREATE DATABASE newcdb
  USER SYS IDENTIFIED BY sys_password
  USER SYSTEM IDENTIFIED BY system_password
  ...
  ENABLE PLUGGABLE DATABASE
    SEED
    FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/newcdb/',
                         '/u01/app/oracle/oradata/pdbseed/')
    SYSTEM DATAFILES SIZE 125M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
    SYSAUX DATAFILES SIZE 100M
  LOCAL UNDO ON;
```

文の読み方は次のとおりです。

- `ENABLE PLUGGABLE DATABASE` が CDB にする指定。初期化パラメータ ENABLE_PLUGGABLE_DATABASE を true にしておく。
- `DB_NAME` は 8 文字まで。グローバル名は DB_NAME と DB_DOMAIN で決まる。
- CDB には UNDO 表領域が要り、UNDO_MANAGEMENT は AUTO にする。省略しても AUTO になる。
- `LOCAL UNDO ON`（`undo_mode_clause`）でローカル UNDO モードになる。この句を省くか `LOCAL UNDO OFF` にすると共有 UNDO モード。既定は共有。
- 文はディレクトリを作らない。使うディレクトリは先に作っておく。
- `SEED` 句の中の `SYSTEM DATAFILES` などで、シードのファイルだけ別の大きさにできる。シードの SYSTEM データ・ファイルは AUTOEXTEND が有効で、ルートの SYSTEM データ・ファイルは既定で無効。

SAMPLE

シードのファイルの名前と場所は、次の 3 つのどれかで決めます。複数を使うと、一覧の上にある方法だけが使われます。

順位	方法	省略した時
1	`SEED FILE_NAME_CONVERT` 句	`NONE` を指定したのと同じ
2	Oracle Managed Files（DB_CREATE_FILE_DEST を設定）	
3	初期化パラメータ PDB_FILE_NAME_CONVERT	

作った後は PDB$SEED を変更できません。datapatch でパッチ・レベルをそろえ、全体のバックアップを取ります。[5]

シードから PDB を作る [1.1]

PDB の作成は、CDB ルートに接続して `CREATE PLUGGABLE DATABASE` を流します。シードから作る時は、PDB の管理者を必ず指定します。管理者は PDB のローカル・ユーザーとして作られ、PDB_DBA ロールがローカルに付与されます。[6]

```
CREATE PLUGGABLE DATABASE salespdb ADMIN USER salesadm IDENTIFIED BY
pwd;
```

ファイルの置き場は、句・パラメータの優先順で決まります。CREATE DATABASE のシード用の順とは違うので、取り違えに注意します。[7]

SAMPLE

順位	句・パラメータ	OMF を使うか
1	`FILE_NAME_CONVERT` 句	使わない
2	`CREATE_FILE_DEST` 句	使う
3	初期化パラメータ DB_CREATE_FILE_DEST	使う
4	初期化パラメータ PDB_FILE_NAME_CONVERT	使わない

FILE_NAME_CONVERT と CREATE_FILE_DEST を両方書くと、ファイルは FILE_NAME_CONVERT の指定で置かれます。CREATE_FILE_DEST の値は、PDB の DB_CREATE_FILE_DEST に設定されます。FILE_NAME_CONVERT のパターンは、OMF が管理するファイルには使えません。`CREATE_FILE_DEST=NONE` は、その PDB の OMF を無効にします。[7]

主な追加の句は次のとおりです。

句	働き
`ROLES=(DBA)`	PDB_DBA ロールに、指定した事前定義ロールをローカルに付与する
`STORAGE (MAXSIZE 2G)`	PDB の全表領域が使う領域を 2GB までに制限する
`DEFAULT TABLESPACE sales DATAFILE '...' SIZE 250M`	シードから作る時は bigfile 表領域を作り、既定の表領域にする
`PATH_PREFIX = '/disk1/.../'`	PDB のディレクトリ・オブジェクトのパスを、そのディレクトリの下に限る

SAMPLE

句	働き
`TEMPFILE REUSE`	同じ名前の一時ファイルが移動先にある時に付ける

PDB シードにはユーザー定義のサービスを持たせられません。シードの内容をコピーして新しい場所に結び付けるのが、PDB 作成の既定の仕組みです。[7]

作成直後の PDB は、マウント・モードで状態は NEW です。オープン・モードは V$PDBS の OPEN_MODE、状態は CDB_PDBS の STATUS で見ます。

```
SQL> select name, open_mode from v$pdbs where name='T1_SALES';
T1_SALES    MOUNTED
SQL> select pdb_name, status from cdb_pdbs where pdb_name='T1_SALES';
T1_SALES    NEW
SQL> ALTER PLUGGABLE DATABASE t1_sales OPEN READ ONLY;
ORA-65085: cannot open pluggable database in read-only mode
SQL> ALTER PLUGGABLE DATABASE t1_sales OPEN;
Pluggable database altered.
SQL> select pdb_name, status from cdb_pdbs where pdb_name='T1_SALES';
T1_SALES    NORMAL
```

新しい PDB は、最初に読み書きで開いて CDB への組込みを終えます。読取り専用では開けません。開くと状態は NORMAL になります。[6]

作成した PDB には、PDB と同じ名前の既定のサービスが作られます。クライアントはこのサービスで PDB に接続します。

SAMPLE

```
SQL> ALTER SESSION SET CONTAINER=t1_sales;
SQL> show con_name
T1_SALES
SQL> show pdbs
    CON_ID CON_NAME      OPEN MODE   RESTRICTED
         4 T1_SALES      READ WRITE  NO
SQL> select granted_role, common from dba_role_privs where
grantee='SALESADM';
PDB_DBA     NO
```

作成の途中でエラーになると、PDB が使えない（UNUSABLE）状態で残ることがあります。UNUSABLE の PDB は削除しかできません。同じ名前で作り直す前に、削除します。作った PDB は、バックアップを取るまでリカバリできません。作成後は datapatch でパッチ・レベルもそろえます。[6]

アプリケーション・ルートで同じ文を流すと、アプリケーション PDB ができます。アプリケーション・シードがあればその雛形を使い、無ければ PDB$SEED を使います。[6]

アプリケーション・コンテナの構成と共通オブジェクト [1.2]

アプリケーション・コンテナは、アプリケーション・ルートと、1 つ以上のアプリケーションのデータを持つアプリケーション PDB の集まりです。アプリケーション PDB はアプリケーション・ルートに差し込まれます。新しいアプリケーション PDB を素早く作りたい時だけ、アプリケーション・シードを任意で作ります。[8]

SAMPLE

アプリケーションとは、アプリケーション・ルートに置く、名前と版を持つ共通のデータとメタデータの組です。同じ構造のデータベースを顧客ごとに持つ SaaS が典型の使い道です。共通の部分を 1 か所で直せば、全顧客の PDB に配れます。[9]

アプリケーションが共有する物は、共通オブジェクトと呼びます。共有のしかたは `SHARING` 句で決め、種類は 3 つです。共有しない時は `NONE` です。

種類	SHARING	メタデータ	データ
メタデータ・リンク	`METADATA`（既定）	ルートで 1 度定義し共有	コンテナごとに別
データ・リンク	`DATA`	ルートで 1 度定義し共有	ルートに 1 つだけ。PDB では読取り専用で共有
拡張データ・リンク	`EXTENDED DATA`	ルートで 1 度定義し共有	ルートの共通分に、各 PDB が自分の分を足せる

`SHARING` を省くと、初期化パラメータ DEFAULT_SHARING（アプリケーション・ルートで設定）の値が使われ、その既定は `METADATA` です。[9]

CREATE 文に SHARING 句を書くと、句の指定が DEFAULT_SHARING より優先されます。[9]

オブジェクトごとに、選べる `SHARING` の値は決まっています。拡張データ・リンクにできるのは、表とビューだけです。[9]

オブジェクト	選べる値
表（オブジェクト表を除く）・ビュー（オブジェクト・ビューを除く）	`METADATA` ・ `DATA` ・ `EXTENDED DATA` ・ `NONE`

SAMPLE

オブジェクト	選べる値
オブジェクト表・オブジェクト・ビュー	`METADATA` ・ `NONE`
順序	`METADATA` ・ `DATA` ・ `NONE`
そのほかの多くのオブジェクト	`METADATA` ・ `NONE`

実機で、3 種類の表を作って PDB から触った結果です。データ・リンクの表は PDB では書けません。拡張データ・リンクの表は、PDB が自分の行を足せます。

```
SQL> insert into t1appusr.country_dlt values (82,'KR');    -- アプリケーション PDB で
ORA-65097: DML into a data link table is outside an application action
SQL> insert into t1appusr.zip_edt values ('200');          -- 拡張データ・リンク
1 row created.
SQL> select code from t1appusr.zip_edt order by 1;          -- PDB 側: ルートの共通分も見える
100
200
```

ルートの `zip_edt` を引くと、ルートが持つ `100` だけが見えます。PDB で足した `200` はその PDB だけの行です。

アプリケーション共通オブジェクトは、アプリケーション PDB の中では作れません。作成・変更・削除ができるのは、アプリケーション・ルートで、インストール・アップグレード・パッチの作業の最中だけです。ローカルの物と名前が衝突

SAMPLE

する PDB は、制限付き（RESTRICTED）で開きます。衝突を直すまで通常のモードで開けません。[9]

アプリケーション・コンテナと PDB を作る [1.2]

アプリケーション・コンテナは、CDB ルートで `CREATE PLUGGABLE DATABASE ... AS APPLICATION CONTAINER` を流して作ります。作り方は 4 つあります。PDB シードから作る・既存の PDB をクローンする・再配置する・アンプラグした PDB を差し込む、です。[10]

```
CREATE PLUGGABLE DATABASE salesact AS APPLICATION CONTAINER
  ADMIN USER salesadm IDENTIFIED BY password;
```

作る前の条件は次のとおりです。

- 現在のコンテナが CDB ルートで、CDB が読み書きで開いている。
- 作る人は共通ユーザーで、CREATE PLUGGABLE DATABASE システム権限を持つ。
- 名前は、1 つの CDB のすべてのコンテナの中で一意。名前の規則はサービス名と同じで、大文字小文字を区別しない。
- Oracle Managed Files で作る。

作成直後はマウント・モードで状態は NEW、既定のサービスも PDB と同じく作られます。最初は読み書きで開く必要があり、読取り専用で開こうとするとエラーです。STORAGE の上限をアプリケーション・ルートに付けても、差し込まれたアプリケーション PDB には効きません。アプリケーション PDB を持つアプリケーション・コンテナは、アンプラグも削除もできません。[10][11]

SAMPLE

アプリケーション PDB は、アプリケーション・ルートに接続して同じ `CREATE PLUGGABLE DATABASE` で作ります。アプリケーション・シードは、アプリケーション・ルートで `CREATE PLUGGABLE DATABASE ... AS SEED` と書いて作ります。[7] 実機でルートに入って作り、開いて確かめると、表はまだ見えません。同期（後述）して初めて共通オブジェクトが現れます。

```
SQL> ALTER SESSION SET CONTAINER=t1_ac;                    -- アプリケーション・ルート
SQL> CREATE PLUGGABLE DATABASE t1_apdb ADMIN USER ap1adm IDENTIFIED BY "..." FILE_NAME_CONVERT=(...);
SQL> ALTER SESSION SET CONTAINER=t1_apdb;
SQL> select count(*) from t1appusr.country_dlt;
ORA-00942: table or view "T1APPUSR"."COUNTRY_DLT" does not exist
SQL> ALTER PLUGGABLE DATABASE APPLICATION t1app SYNC;
SQL> select count(*) from t1appusr.country_dlt;
         1
```

同じ文を CDB ルートで流すと、作られるのは普通の PDB です。普通の PDB は、アプリケーション・ルートからは開けません。実機で、アプリケーション・ルートから `ALTER PLUGGABLE DATABASE t1_ap1 OPEN` を流すと、`ORA-65194` が返ります。メッセージは、その PDB がアプリケーション・コンテナに属さないという意味です。

アプリケーションのインストール・アップグレード・パッチ・同期 [1.2]

アプリケーションの保守は、`ALTER PLUGGABLE DATABASE APPLICATION ... BEGIN` で始め、`END` で終えます。間に流した文が、アプリケーションの変更として記録されます。名前と版（文字列）の指定は省けません。[9]

SAMPLE

作業	始め	終わり	使う場面
インストール	`BEGIN INSTALL '1.0'`	`END INSTALL '1.0'`	最初の定義。ユーザー・表・パッケージを作る
アップグレード	`BEGIN UPGRADE '1.0' TO '2.0'`	`END UPGRADE TO '2.0'`	構造の大きな変更
パッチ	`BEGIN PATCH 101 MINIMUM VERSION '1.0'`	`END PATCH 101`	バグ修正など小さな変更。新しい関数やパッケージは追加できる

```
ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app BEGIN INSTALL
'1.0';
CREATE USER saas_sales_adm IDENTIFIED BY manager CONTAINER=ALL;
CREATE TABLE saas_sales_adm.sales_mlt SHARING=METADATA (YEAR NUMBER(4),
REVENUE NUMBER);
ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app END INSTALL '1.0';
```

アップグレードの最中も、アプリケーションは使えます。データベースがアプリケーション・ルートをクローンし（元の版のまま・読取り専用）、アプリケーション PDB はクローンを指します。アップグレードを終えて PDB が同期するまで、PDB は旧版のクローンを使い続けます。別の版を使い続ける PDB を同じコンテナに置けます。[9]

パッチには制限があります。DROP 文や、列を落とす・データ型を変える ALTER TABLE は入れられません。実機の結果は次のとおりです。パッチやアップグレードが進行中の間は、別のパッチも始められません。

SAMPLE

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION t1app BEGIN PATCH 101 MINIMUM
VERSION '1.0';
SQL> alter table t1appusr.prod_mlt drop column name;
ORA-65270: operation is not allowed in an application patch
SQL> alter table t1appusr.prod_mlt add (price number);
Table altered.
SQL> ALTER PLUGGABLE DATABASE APPLICATION t1app BEGIN PATCH 102 MINIMUM
VERSION '1.0';
ORA-65213: existing application action in progress
```

破壊的な変更はアップグレードで行います。パッチで出した文が「パッチでは許されない」エラーになった時も、同じくアップグレードに替えます。[9]

ルートでの変更は、アプリケーション PDB に自動では届きません。アプリケーション PDB に接続して、`SYNC` を流します。同期とは、アプリケーション PDB でアプリケーションをルートの最新の版とパッチに更新する、利用者が起こす操作です。[9]

同期の書き方	意味
`ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC`	1 つのアプリケーションを最新に
`... APPLICATION saas_sales_app SYNC TO PATCH 100`	パッチ 100 まで。1 つの名前を指定した時だけ使える
`... APPLICATION saas_sales_app SYNC TO '2.0'`	版 2.0 まで。1 つの名前を指定した時だけ使える
`... APPLICATION apexapp, ordsapp SYNC`	複数の名前。TO PATCH と TO 版は使えない

SAMPLE

同期の書き方	意味
`... APPLICATION ALL SYNC`	全部。暗黙に作られた物も含む
`... APPLICATION ALL EXCEPT apexapp SYNC`	指定した物を除く全部

暗黙のアプリケーションは、`BEGIN` の無いまま共通ユーザーの操作（`CREATE USER ... CONTAINER=ALL` など）をした時に、データベースが作る物です。名前は `APP$` と GUID です。`ALL SYNC` は、この暗黙の物も同期します。[9]

受け入れる最も古い版は、アプリケーション・ルートで決めます。文は `ALTER PLUGGABLE DATABASE APPLICATION salesapp SET COMPATIBILITY VERSION '4.2'`（または `CURRENT`）です。それより古い版の PDB は差し込めません。古い版のルートのクローンも、この時に暗黙に削除されます。[9]

既に PDB の中に入っているアプリケーションは、アプリケーション・コンテナへ移せます。アプリケーション・ルートで `BEGIN INSTALL` を流し、その最中に DBMS_PDB_ALTER_SHARING（または DBMS_PDB）で共有モードを設定します。共有モードは METADATA・DATA・EXTENDED DATA です。ユーザー・ロール・プロファイルも、同じ最中にアプリケーションへ結び付けます。大量のデータの投入に使えるのは、SQL*Loader の従来型パス・ロードだけで、直接パス・ロードは使えません。[9]

PDB をクローンする [1.3]

クローンとは、元の PDB から新しい PDB を作ることです。元は同じ CDB のローカル PDB でも、別の CDB のリモート PDB でも構いません。すべて `CREATE PLUGGABLE DATABASE` の `FROM` 句を使います。シードや、アプリケーション・シードを `FROM` に指定することはできません。[12]

SAMPLE

```
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1;                    -- ローカル
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1@pdb1_link;          -- リモート（データベース・リンク）
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1 NO DATA;            -- 構造だけ
```

ローカルのクローンには条件があります。

- 元の PDB は閉じていてはいけない。リモートでも同じ。
- CDB がローカル UNDO モードでなければ、元の PDB は読取り専用で開いている。
- CDB が ARCHIVELOG モードでなければ、元の PDB は読取り専用で開いている。
- 作る人は、ルートと元の PDB の両方で CREATE PLUGGABLE DATABASE システム権限を持つ。

CDB が ARCHIVELOG モードでローカル UNDO モードなら、元の PDB を読み書きで開いたままクローンできます。これをホット・クローンと呼びます。クローンの間に出るアーカイブ・ログは、終わるまで残しておきます。[12]

リモートのクローンでは、移し先の CDB から元へデータベース・リンクを張ります。リンクは、元の CDB のルートへ共通ユーザーでつなぐか、元の PDB へ共通ユーザーかローカル・ユーザーでつなぎます。リンクのユーザーには、元の PDB での CREATE PLUGGABLE DATABASE 権限（共通でもローカルでもよい）か、SYSOPER 権限のどちらかが要ります。源と移し先は、エンディアンが同じで、源のデータベース・オプションが移し先と同じか、その部分集合である

SAMPLE

必要があります。文字セットは、移し先の CDB が AL32UTF8 なら条件がありません。AL32UTF8 でなければ、互換性のある文字セットが要ります。[12]

`NO DATA` は、元の PDB のデータ・モデルの定義だけを複製します。文書は、辞書のデータは複製し、利用者が作った表と索引のデータは捨てると書きます。実機（26ai Free 23.26.3）では、USERS 表領域に置いた利用者の表は 0 行になり、SYSTEM 表領域に置いた利用者の表は 2 行のまま残りました。元の PDB は読取り専用で開いておきます。クラスタ表・AQ の表・索引構成表・抽象データ型の列を持つ表がある PDB には使えません。[7]

クローンした PDB も、作った直後はマウント・モードで状態は NEW です。読み書きで開いて NORMAL になります。バックアップを取るまで、この PDB はリカバリできません。元の PDB の既定の一時表領域を使っていたユーザーは、新しい PDB の既定の一時表領域を使います。[12]

リフレッシュ可能クローン（リフレッシュ可能 PDB）は、元の PDB を複製して、後から更新できるクローンです。`REFRESH MODE` 句で作ります。

REFRESH MODE	意味
`NONE`（既定）	リフレッシュできない PDB を作る
`MANUAL`	`ALTER PLUGGABLE DATABASE ... REFRESH` で手動に更新する
`EVERY n MINUTES`	n 分ごとに自動で更新する。手動の更新もできる

```
CREATE PLUGGABLE DATABASE pdb1_ref_cln FROM pdb1@pdb1_link REFRESH MODE
EVERY 60 MINUTES;
```

SAMPLE

`REFRESH MODE` が使えるのは、`CREATE PLUGGABLE DATABASE ... FROM` の文だけです。元の PDB は ARCHIVELOG モードかつローカル UNDO モードに限ります。データベース・リンクが要り、リンクは同じ CDB に戻るものでも別の CDB へのものでも構いません。更新していない間は読取り専用で開けます。更新は PDB を閉じて行います。開いたままだと、自動の更新は次の予定まで延ばされ、手動の更新はエラーを返します。[12]

実機で、60 分ごとに自動で更新するクローンを作った直後の状態と、普通の PDB に戻して再び戻そうとした結果です。

```
SQL> select pdb_name, refresh_mode, refresh_interval, status from
cdb_pdbs where pdb_name='T1_REF2';
T1_REF2    AUTO      60     REFRESHING
SQL> alter pluggable database t1_ref refresh mode none;
SQL> alter pluggable database t1_ref open read only;
ORA-65085: cannot open pluggable database in read-only mode
SQL> alter pluggable database t1_ref open read write;
SQL> select pdb_name, refresh_mode, status from cdb_pdbs where
pdb_name='T1_REF';
T1_REF     NONE      NORMAL
SQL> alter pluggable database t1_ref refresh mode manual;
ORA-65261: pluggable database T1_REF not enabled for refresh
```

リフレッシュ可能クローンは、`ALTER PLUGGABLE DATABASE ... REFRESH MODE NONE` を流して読み書きで開くと、普通の PDB に変わります。普通の PDB をリフレッシュ可能クローンに変えることはできません。変えた後に元へ戻すこともできません。元の PDB へリンクで届かない時のために、クローン側で REMOTE_RECOVERY_FILE_DEST に、リフレッシュの時に読むアーカイブ・ロ

SAMPLE

グのディレクトリを指定できます。`ALTER PLUGGABLE DATABASE ... SWITCHOVER` で、元とクローンの役割を入れ替えられます。[12]

形	句	特徴
スナップショットから	`FROM pdb1 USING SNAPSHOT 名前（または SCN・時刻）`	PDB スナップショットから、完全に独立した PDB を作る
スナップショット・コピー PDB	`FROM pdb1 SNAPSHOT COPY`	ストレージのスナップショットに依存する。CDB ルートからのアンプラグも、アプリケーション・ルートへのプラグもできない
ミラーの分割	`USING MIRROR COPY 名前`	ASM のミラーを分割して作る

スナップショット・コピー PDB は、`ALTER PLUGGABLE DATABASE MATERIALIZE` で完全な PDB に変えられます。元になったストレージのスナップショットは、スナップショット・コピー PDB が残る間は削除できません。[12]

アンプラグとプラグイン [1.3]

アンプラグは PDB を CDB から切り離す操作です。PDB は、CDB に差し込まれている間だけ使えます。切り離す理由は、別の CDB へ移す・保管しておく・使えなくする、の 3 つです。[13]

切り離した PDB の情報は、次のどちらかに書き出します。

形式	中身
XML メタデータ・ファイル（.xml）	PDB とそのファイル（データ・ファイル・ウォレット）を記述する。ファイル本体は別に残る

SAMPLE

形式	中身
アーカイブ(.pdb)	XML と PDB のファイルを両方含む、圧縮した 1 つのファイル

```
ALTER PLUGGABLE DATABASE salespdb UNPLUG INTO
'/oracle/data/salespdb.xml';
ALTER PLUGGABLE DATABASE salespdb UNPLUG INTO '/oracle/data/sales.pdb';
```

アンプラグには、SYSDBA か SYSOPER の権限(共通でもローカルでもよい)が要ります。その PDB は、少なくとも 1 度は開いたことがなければなりません。PDB は先に閉じておきます。RAC なら全インスタンスで閉じます。実機では、開いたまま流すとエラーです。切り離した後の PDB は、マウント・モードのままで、状態は UNPLUGGED になります。[13]

```
SQL> alter pluggable database t1_clone unplug into
'/tmp/t1unplug/t1_clone.xml';
ORA-65025: Pluggable database T1_CLONE is not closed on all instances.
SQL> alter pluggable database t1_clone close immediate;
SQL> alter pluggable database t1_clone unplug into
'/tmp/t1unplug/t1_clone.xml';
SQL> select pdb_name, status from cdb_pdbs where pdb_name='T1_CLONE';
T1_CLONE     UNPLUGGED
SQL> alter pluggable database t1_clone open;
ORA-65086: cannot open/close the pluggable database
```

切り離した PDB にできる操作は、削除だけです。切り離した PDB も同じ CDB の全体バックアップに入るので、保管の手段にもなります。同じ CDB に差し込

SAMPLE

み直すには、先に削除します。`DROP PLUGGABLE DATABASE` の句は次のとおりです。[13]

句	動き
`KEEP DATAFILES`（既定）	データ・ファイルを残す。この時 PDB はアンプラグ済みでなければならない。一時ファイルは消える
`INCLUDING DATAFILES`	データ・ファイルも消す。`SNAPSHOT COPY` で作った PDB は、必ずこの句
`FORCE`	孤立したアプリケーション・ルートのクローンを削除する

削除する PDB は、マウント・モードか、アンプラグ済みです。作る人は SYSDBA か SYSOPER の権限を持ちます。削除すると制御ファイルから PDB への参照が消えます。アーカイブ・ログとバックアップは残るので、RMAN で消します。[13]

差し込む前に、まず互換性を確かめます。`DBMS_PDB.CHECK_PLUG_COMPATIBILITY` 関数が TRUE か FALSE を返します。FALSE の時は、`PDB_PLUG_IN_VIOLATIONS` ビューで理由を見ます。まだアンプラグしていない PDB は、`DBMS_PDB.DESCRIBE` で XML を作ってから確かめます。[14]

SAMPLE

```
BEGIN
  DBMS_PDB.DESCRIBE(pdb_descr_file => '/disk1/oracle/salespdb.xml',
pdb_name => 'SALESPDB');
END;
/
SELECT CASE DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
         pdb_descr_file => '/disk1/oracle/salespdb.xml', pdb_name =>
'SALESPDB')
       WHEN TRUE THEN 'YES' ELSE 'NO' END FROM dual;
```

差し込みは `CREATE PLUGGABLE DATABASE ... USING` に、XML か .pdb のファイルを指定します。条件は次のとおりです。移動元と移動先で、エンディアンが同じで、オプションが同じかその部分集合です。文字セットの条件はクローンと同じです。移動元の CDB が移動先より新しいリリースなら、差し込めません。自動のダウングレードが無いからです。[14]

差し込みの句は、ファイルの位置と、PDB の識別子の 2 点で選びます。

場面	付ける句
ファイルが今の場所に置いたままで使える	`NOCOPY`（要なら `TEMPFILE REUSE`）
ファイルを新しい場所に写す	`COPY`（既定）と `FILE_NAME_CONVERT` か OMF
ファイルを新しい場所に動かす	`MOVE`
XML の記述と実際のファイルの場所が違う	`SOURCE_FILE_NAME_CONVERT` か `SOURCE_FILE_DIRECTORY`（どちらか一方。XML の時だけ。.pdb では使えない）

SAMPLE

場面	付ける句
同じアンプラグ済み PDB から作った PDB が、既に同じ CDB にある	`AS CLONE`（新しい DBID と GUID を持たせる）
XML が壊れた・見つからない	`DBMS_PDB.RECOVER` で、データ・ファイルから XML を作る

.pdb のアーカイブを使うと、XML には相対のファイル名だけが入るので、元のファイルの場所を指定する句は要りません。実機で、同じ XML を 2 回差し込むと 2 回目は GUID の衝突で止まります。ここで `AS CLONE` が要ります。OMF も変換も使わず `NOCOPY` も付けないと、位置の指定を求められます。[14]

```
SQL> CREATE PLUGGABLE DATABASE t1_plugged USING
'/tmp/t1unplug/t1_clone.xml' NOCOPY TEMPFILE REUSE;
Pluggable database created.
SQL> CREATE PLUGGABLE DATABASE t1_plugged2 USING
'/tmp/t1unplug/t1_clone.xml' NOCOPY TEMPFILE REUSE;
ORA-65122: Pluggable database GUID conflicts with the GUID of an
existing container.
SQL> CREATE PLUGGABLE DATABASE t1_arch USING
'/tmp/t1unplug/t1_plugged.pdb';
ORA-65018: FILE_NAME_CONVERT or NOCOPY must be specified
```

差し込んだ後の PDB は、作成直後と同じマウント・モードで状態は NEW です。読み書きで開いて、バックアップを取ります。最初に読取り専用で開こうとするとエラーになり、CDB への組み込みは読み書きで開いた時に終わります。リリースの違う PDB を開くと、自動でアップグレードされます（後述の 1.5）。差し込む先に無い共通ユーザーは、所有する物が無ければ同期の時に削除さ

SAMPLE

れ、あればロックされます。同じ名前の共通ユーザーが先にあれば、その共通ユーザーに結び付きます。[14]

TDE で暗号化したデータを含む PDB は、暗号化の鍵を移す手順が加わります（1.4 の TDE）。クローンする時は、`KEYSTORE IDENTIFIED BY` でキーストアのパスワードを渡します。源に暗号化データがあるかは V$ENCRYPTION_KEYS で分かります。[12]

PDB の再配置とスナップショット [1.3]

再配置は、PDB を別のコンテナへ移す操作で、停止時間がほとんど無い方法です。移動元の PDB は、移動の間も読み書きで動き続けます。[4]

```
CREATE PLUGGABLE DATABASE pdb1 FROM pdb1@link2src RELOCATE AVAILABILITY
NORMAL;
ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

動きは次の順です。

1. `RELOCATE` の文が、オンラインでデータ・ブロック・REDO・UNDO を新しい場所へ写す（ホット・クローンと同じ）。
2. `OPEN` を流すと、移動先が最後の REDO・UNDO を適用して一貫した状態になる。
3. 同時に、移動元の接続を排出（ドレイン）し、移動元の PDB を閉じて、移動先を読み書きで開く。

移動元と移動先が同時に開くことはありません。条件は、移動元の CDB がローカル UNDO モードで、移動先から移動元へのデータベース・リンクがあることです。リンクのユーザーには CREATE PLUGGABLE DATABASE か

SAMPLE

SYSOPER が要ります。移動元が CDB ルートなら共通ユーザーのリンクです。移動先の CDB が ARCHIVELOG でなければ、操作中は移動先の PDB を読取り専用で開きます。[4]

`AVAILABILITY` 句でリスナーの扱いを選びます。

句	使う場面	動き
`NORMAL`（既定）	共通のリスナー・ネットワーク（同じリスナー、または互いに登録し合うリスナー）	SQL*Net の層が接続を自動で付け替える。PDB のサービス名は、共通のネットワークで一意
`MAX`	リスナーが独立している（データセンター間など）	元のリスナーに転送の先を設定する。一時的な設定

`MAX` では、移動元に「トゥームストーン PDB」が残り、DBA_PDBS で状態が RELOCATED になります。接続文字列を移動先へ直した後で削除します。SCAN リスナーを経由する時は、リスナーごとに `ALLOW_MULTIPLE_REDIRECTS_リスナー名=YES` を設定します（ノード・リスナーには設定しません）。[4]

PDB スナップショット・カルーセルは、PDB のある時点の複製（スナップショット）を、決まった個数まで循環して持つ仕組みです。上限に達すると、一番古いスナップショットを上書きします。[15]

操作	文
手動で作る	`ALTER PLUGGABLE DATABASE SNAPSHOT 名前`
モードを選ぶ	`SNAPSHOT MODE MANUAL`（既定）／`EVERY 間隔 MINUTES` か `EVERY 間隔 HOURS`（分は 3000 未満、時間は 2000 未満）／`NONE`

SAMPLE

操作	文
数の上限	`SET MAX_PDB_SNAPSHOTS=n` 。0 にすると全部消える
作り直す	`CREATE PLUGGABLE DATABASE 新 FROM 元 USING SNAPSHOT 名前`
見る	DBA_PDB_SNAPSHOTS・DBA_PDB_SNAPSHOTFILE・DBA_PDBS の SNAPSHOT_MODE と SNAPSHOT_INTERVAL

自動の作成にも手動の作成にも、CDB のローカル UNDO モードが要ります。ファイル・システムがスパース・ファイルに対応すれば、最初の 1 つだけが完全なコピーで、後は差分だけのスパースなコピーです。名前を付けない時は、`SNAP_` に SCN を含む識別子の名前が自動で付くので、決める必要はありません。クローンの章は、カルーセルを「最大 8 個のスナップショットの集まり」と説明します。実機（23.26.3 Free）の MAX_PDB_SNAPSHOTS も 8 です。管理の章は、同じプロパティの既定と上限を 4096 と書き、数が揃いません。[1215]

```
SQL> select property_name, property_value from database_properties
where property_name like 'MAX_PDB%';
MAX_PDB_SNAPSHOTS     8
SQL> alter pluggable database snapshot snap1;
ORA-12754: Feature PDB SNAPSHOT CAROUSEL is disabled due to missing
capability .
```

Free エディションではカルーセルが使えず、上のエラーになります。

SAMPLE

CDB と PDB の起動・停止とオープン・モード [1.3]

CDB の起動は、インスタンスの起動・マウント・オープンの順で、通常のデータベースと同じです。CDB ルートを開いた時点で PDB が自動で開くとは限りません。実機でインスタンスが再起動した直後、それまで開いていた PDB はすべて MOUNTED になっています。開いた状態を残す手段が `SAVE STATE` です。[16]

停止は、SYSOPER・SYSDBA・SYSBACKUP・SYSDG のどれかで接続して `SHUTDOWN` を流します。モードは 4 つです。

モード	動き
NORMAL（既定）	全利用者の切断を待つ。次の起動でインスタンス・リカバリは要らない
IMMEDIATE	実行中の SQL を止め、未コミットの変更を戻して切断する
TRANSACTIONAL	新しいトランザクションを止め、実行中のトランザクションの終了を待つ
ABORT	インスタンスを即時に止める

待つ型の上限は 1 時間です。PDB に接続して `STARTUP` を流すとその PDB が開きます。`SHUTDOWN` を流しても、インスタンスは止まりません。その PDB が閉じるだけです。ルートからは `STARTUP PLUGGABLE DATABASE 名前` で 1 つの PDB を開けます。[17]

PDB のオープン・モードは 5 つです。

SAMPLE

モード	動き
マウント（MOUNTED）	データ・ファイルを読み書きしない。DBA だけが接続でき、一貫したバックアップを取れる
読み書き（READ WRITE）	既定。`ALTER PLUGGABLE DATABASE OPEN` だけを指定した時もこれ。ただし物理スタンバイの PDB は読取り専用が既定
読取り専用（READ ONLY）	問合せだけ
ハイブリッド読取り専用（HYBRID READ ONLY）	CDB の共通ユーザーには読み書き、PDB のローカル・ユーザーには読取り専用に見える
MIGRATE（`OPEN UPGRADE`）	アップグレードのスクリプトを流すモード

`OPEN` には `RESTRICTED`（RESTRICTED SESSION 権限を持つ利用者だけが入れる）と `FORCE`（開いている PDB の状態も変える）を付けられます。`RESTRICTED FORCE` なら、その権限を持たない接続は切られてトランザクションが戻ります。複数の PDB を指定するには、名前を並べるか `ALL`、`ALL EXCEPT 名前` を使います。[17]

`FORCE` を付けない時は、PDB がマウント・モードでなければ開けません。

目標のモード	FORCE	ルートの状態	変えられる PDB の状態
読み書き	あり	読み書き	マウント・読取り専用・読み書き
読み書き	なし	読み書き	マウントのみ
読取り専用	あり	読取り専用か読み書き	マウント・読取り専用・読み書き

SAMPLE

目標のモード	FORCE	ルートの状態	変えられる PDB の状態
読取り専用	なし	読取り専用か読み書き	マウントのみ

CDB を再起動しても PDB のモードを残すには、ルートで `ALTER PLUGGABLE DATABASE 名前 SAVE STATE` を流します。元に戻すのは `DISCARD STATE` で、その PDB は再起動のたびにマウントになります。`ALL`・`ALL EXCEPT` も使え、実行には ALTER DATABASE 権限が要ります。保存した状態は DBA_PDB_SAVED_STATES で見ます。RAC では、SAVE STATE と DISCARD STATE は今のインスタンスのモードにしか効きません。[17]

```
SQL> alter pluggable database t1_arch save state;
SQL> select con_name, state from dba_pdb_saved_states where con_name
like 'T1%';
T1_ARCH     OPEN
SQL> alter pluggable database t1_arch discard state;
SQL> select con_name, state from dba_pdb_saved_states where con_name
like 'T1%';
no rows selected
```

23ai 以降は、PDB に優先順位を付けられます。`ALTER PLUGGABLE DATABASE 名前 PRIORITY n`（n は 1 から 4096）で、小さい値の PDB から先に開く・状態を戻す・アップグレードします。優先順位を付けていない PDB が最後で、アプリケーション PDB はアプリケーション・ルートの後に開きます。プラグインやリフレッシュ可能クローンでは、優先順位は引き継がれません。PDB を開く時には、CDB との互換性も確かめられます（1.5 のリプレイ・アップグレード）。[17]

SAMPLE

PDB のサービスと接続 [1.3]

PDB を作ると、PDB と同じ名前の既定のサービスが自動で作られます。既定のサービスは管理できず、管理の作業にだけ使います。サービス名は 1 つの CDB の中で一意で、同じリスナーでつながる CDB の間でも一意でなければなりません。アプリケーションには、自分で作ったサービスを使います。[18]

サービスは、PDB プロパティを持たせて作ります。PDB プロパティを持つサービスにつなぐと、現在のコンテナがその PDB になります。作り方は 2 つです。

- SRVCTL の `srvctl add service ... -pdb 名前` 。`-pdb` を省くと、サービスはルートに付く。
- DBMS_SERVICE のプロシージャ。

サービスと PDB の対応は ALL_SERVICES で見ます。サービスはリスナーにも登録されます。PDB をアンプラグ・削除しても、サービスは自動では消えません。[18]

```
SQL> exec DBMS_SERVICE.CREATE_SERVICE(service_name => 't1_sales_app',
network_name => 't1_sales_app');
SQL> exec DBMS_SERVICE.START_SERVICE('t1_sales_app');
SQL> select name, pdb from v$services where name like 't1_sales%' order
by 1;
t1_sales          T1_SALES
t1_sales_app      T1_SALES
-- リスナー側: lsnrctl services
Service "freepdb1" has 1 instance(s).
Service "t1_ac" has 1 instance(s).
Service "t1_arch" has 1 instance(s).
```

SAMPLE

PDB を開く時の `SERVICES` 句は、起動するサービスを決めます。値は `SERVICES = ('名前', ...)`、`ALL`、`ALL EXCEPT(...)`、`NONE`（既定）です。既定のサービスは、`SERVICES` の指定にかかわらず必ず起動します。Oracle Restart や Clusterware の管理下で、閉じている PDB のサービスを SRVCTL で起動すると、PDB が読み書きで開きます。サービスを止めても、PDB のモードは変わりません。[17][18]

接続の方法は 2 つです。

- `CONNECT hr@hrapp` のように、PDB のサービスの Easy Connect か、ネット・サービス名で入る。共通ユーザーのルートへの接続も同じ形で、CREATE SESSION が要る。
- 共通ユーザーが、入った後に `ALTER SESSION SET CONTAINER = 名前 [SERVICE = サービス名]` で別のコンテナへ移る。

マウント状態の PDB に接続できるのは、SYSDBA・SYSOPER・SYSBACKUP・SYSDG の権限を持つ利用者だけです。`SET CONTAINER` の移り先は、`CDB$ROOT`・`PDB$SEED`・PDB の名前です。サービスを省くと既定のサービスを使います。既定のサービスでは、サービス・メトリック・FAN・Resource Manager・Transaction Guard・Application Continuity などが使えません。[19]

`ALTER SESSION SET CONTAINER` には次の決まりがあります。

- 移った後のカレント・スキーマは、共通ユーザーがそのコンテナに持つスキーマになる。
- ログイン・トリガーは動かない。代わりに、PDB で SET CONTAINER の前後に動くトリガーを作れる。
- パッケージの状態はコンテナの間で共有されない。

SAMPLE

- トランザクションを始めてから移ると、元のコンテナに戻るまで DML・DDL・COMMIT・ROLLBACK を流せない。開いたカーソルも、戻るまでフェッチできない。

現在のコンテナは `SYS_CONTEXT('USERENV','CON_NAME')` と `'CON_ID'` で見られます。[19]

初期化パラメータの範囲 [1.3]

PDB は、ルートの初期化パラメータの値を引き継ぎます。引き継ぐかどうかは、パラメータごとの継承の属性（真か偽）で決まります。`V$SYSTEM_PARAMETER.ISPDB_MODIFIABLE` が TRUE のパラメータだけ、PDB ごとに値を変えられ、継承を偽にできます。ISPDB_MODIFIABLE が FALSE のパラメータは、PDB では変えられません。[19]

操作	結果
ルートで `ALTER SYSTEM SET パラメータ = 値 CONTAINER = CURRENT`（既定）	ルートと、継承が真の PDB に効く
ルートで `... CONTAINER = ALL`	すべてのコンテナに効き、全 PDB の継承が真になる
PDB で `ALTER SYSTEM SET パラメータ = 値`	その PDB だけ。継承は偽になる
PDB で `ALTER SYSTEM RESET パラメータ`	継承が真に戻る

`CONTAINER = ALL` を使えるのは、現在のコンテナが CDB ルートかアプリケーション・ルートの時です。使えるのは、SET CONTAINER 権限を共通に持つ共通ユーザーです。ルートで流すと、ルートのパラメータを変える権限（共通に持つ ALTER SYSTEM）が要ります。実機の結果です。[19]

SAMPLE

```
SQL> alter session set container=t1_arch;
SQL> alter system set open_cursors=333 scope=both;
System altered.
SQL> alter system set processes=500 scope=spfile;
ORA-65040: Operation is not allowed from within a pluggable database.
SQL> select name, value, con_id, ispdb_modifiable from
v$system_parameter where name='open_cursors';
open_cursors   333     16    TRUE
SQL> alter system reset open_cursors scope=both;
SQL> select name, value, con_id from v$system_parameter where
name='open_cursors';
open_cursors   300      0
```

`CONTAINER = ALL` は、PDB が個別に持っていた値を継承に戻します。[19]

ローカル UNDO と共有 UNDO [1.3]

CDB は、ローカル UNDO か共有 UNDO のどちらかで動きます。モードは CDB 全体に 1 つです。CREATE DATABASE で UNDO の句を書かなければ、共有 UNDO になります。[19]

項目	ローカル UNDO	共有 UNDO（作成時の既定）
UNDO 表領域	コンテナごとに、開いているインスタンスごとに持つ	単一インスタンスなら有効なものが 1 つ。RAC はインスタンスごとに 1 つ
作れる人	対象コンテナで権限を持つ人	CDB ルートの共通ユーザーだけ。PDB で作るとエラー

SAMPLE

項目	ローカル UNDO	共有 UNDO（作成時の既定）
見える範囲	全コンテナの静的ビューと V$ ビュー	ルートでは静的ビューと V$ ビュー。PDB では V$ ビューだけ
働き	コンテナの独立性が上がる。アンプラグや PDB の PITR などが効率よく動く	既定。UNDO を CDB で 1 つに集める

ローカル UNDO は、次の操作で必須です。読み書きで開いたままのホット・クローン、PDB の再配置、リフレッシュ可能クローンの元の PDB、PDB スナップショット・カルーセルです。ローカル UNDO でない時に PDB をクローンするなら、元の PDB は読取り専用で開いていなければなりません。[19124]

モードを見るには、ルートで次の問合せを流します。TRUE ならローカル UNDO です。

```
SQL> select property_name, property_value from database_properties
where property_name='LOCAL_UNDO_ENABLED';
LOCAL_UNDO_ENABLED     TRUE
```

モードを変える手順は次のとおりです。ローカルへ変える時の手順と、共有へ戻す時の手順は、`ON` と `OFF` の違いだけです。

1. インスタンスを停止する。
2. `STARTUP UPGRADE` で OPEN UPGRADE モードで起動する。
3. ルート（`SHOW CON_NAME` が CDB$ROOT）で `ALTER DATABASE LOCAL UNDO ON`（共有へ戻すなら `OFF`）を流す。

SAMPLE

4. インスタンスを停止し、再び起動する。

変更は、各コンテナで変更後に最初に開いた時に効きます。ローカル UNDO に変えると、UNDO 表領域が無いコンテナには自動で作られます。変更の前の時点や SCN へは、フラッシュバックできません。共有に戻すと、ローカルの UNDO 表領域は無視されます。Oracle は、使わなくなった表領域の削除を勧めています。PDB を、共有 UNDO の CDB で作った後にローカル UNDO の CDB に差し込むと、最初に開く時に UNDO 表領域が自動で作られます。[19]

CDB と PDB のリソース管理 [1.3]

Resource Manager は、CDB と PDB の 2 段でリソースを管理します。CDB リソース・プランが、ディレクティブに従って PDB にリソースを配ります。PDB の中は、PDB リソース・プランが管理します。[20]

CDB リソース・プランのディレクティブは、PDB または PDB 性能プロファイルを対象に、次の値を持ちます。

属性	意味
`shares`	保証するリソースの割合を決める値。全 PDB の shares の合計に対する比で効く
`utilization_limit`	CPU 使用率の上限（0 から 100%）
`parallel_server_limit`	並列実行サーバーの上限（0 から 100%）。CDB ルートの PARALLEL_SERVERS_TARGET への割合。Oracle は、このディレクティブより PARALLEL_SERVERS_TARGET の設定を勧めています

SAMPLE

3 つの PDB の shares が 3•3•1 なら、合計 7 のうち 3/7•3/7•1/7 を保証します。保証は競合している時の話で、競合が無ければ、どの PDB も保証を超えて使えます。CPU は、使用率の上限を超えようとする PDB のセッションを絞って守ります。[20]

ディレクティブが無い PDB は、既定のディレクティブを使います。初期値は shares が 1、utilization_limit と parallel_server_limit が 100 です。`UPDATE_CDB_DEFAULT_DIRECTIVE` で変えられます。差し込んだばかりの PDB は、ディレクティブを持ちません。アンプラグしてもディレクティブは残り、同じ PDB を戻せば、削除していない限りその値が働きます。CDB プランが `DEFAULT_CDB_PLAN` か `DEFAULT_MAINTENANCE_PLAN` の時は、shares が PDB の CPU_MIN_COUNT になります。使用率の上限は、PDB の CPU_COUNT を CDB の CPU_COUNT で割った値です。[20]

PDB 性能プロファイルは、同じ重要度の PDB をまとめるための名前（ゴールド•シルバー•ブロンズなど）です。ディレクティブは PDB ではなくプロファイルに付けるので、PDB が多い環境の管理が軽くなります。各 PDB で初期化パラメータ `DB_PERFORMANCE_PROFILE` にプロファイル名を設定します。Oracle は、この設定をロックダウン•プロファイル（後述）に入れて、PDB の管理者が付け替えられないようにするよう勧めています。[20]

CDB プランの作り方は、次の順です。PDB ごとのディレクティブも、性能プロファイルの場合も、順は同じです。

1. `DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA` で保留領域を作る。
2. `CREATE_CDB_PLAN` でプランを作る。
3. `CREATE_CDB_PLAN_DIRECTIVE` で PDB ごとのディレクティブを作る（プロファイルなら `CREATE_CDB_PROFILE_DIRECTIVE`）。

SAMPLE

4. 必要なら `UPDATE_CDB_DEFAULT_DIRECTIVE` で既定を直す。
5. `VALIDATE_PENDING_AREA` で検証し、`SUBMIT_PENDING_AREA` で確定する。

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN(plan => 'newcdb_plan', comment
=> 'CDB resource plan');
  DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE(plan =>
'newcdb_plan',
    pluggable_database => 'salespdb', shares => 3,
    utilization_limit => 100, parallel_server_limit => 100);
END;
/
EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

プランは初期化パラメータ `RESOURCE_MANAGER_PLAN` で有効にします。値の頭に `FORCE:` を付けると、メンテナンス・ウィンドウがプランを切り替えなくなります。Free エディションでは、`CREATE_CDB_PLAN` を流した時点で、次のエラーが返ります。[20]

```
ORA-00439: feature not enabled: Database resource manager
```

PDB のメモリー・CPU は、PDB で設定する初期化パラメータでも抑えられます。

SAMPLE

パラメータ	働き	守る条件
`SGA_TARGET`	PDB が使える SGA の上限を決める。PDB に設定すべき共有メモリーの大きさのパラメータは、これだけ	ルートの SGA_TARGET が 0 でない時に効く。ルートの値以下
`SGA_MIN_SIZE`	PDB の SGA の下限	効く条件と上限の割合は、文書によって書き方が違う（管理者ガイドは「ルートの SGA_TARGET が未設定か 0 でないと効かない」、リファレンスは「ルートに SGA_TARGET が要る」と書き、割合も 50% と 20% で違う）
`SHARED_POOL_SIZE`	PDB に保証する共有プールの最小の大きさ。PDB で設定しなければ、使える量の上限は CDB の共有プールの大きさだけ	ルートで SGA_TARGET を設定せず SHARED_POOL_SIZE を設定した時は、CDB の値の 50% 以下（全 PDB の合計も 50% 以下）
`PGA_AGGREGATE_LIMIT`	PDB の PGA の上限	ルートの値以下で、PDB の PGA_AGGREGATE_TARGET の 2 倍以上
`CPU_COUNT`	PDB が同時に使える CPU スレッド数の上限	utilization_limit と両方あれば、小さい方が働く
`CPU_MIN_COUNT`	最低限のスレッド数（0.1 から CPU_COUNT まで、0.05 の倍数）	1 未満なら、スレッドを必要とする時間の割合

`CPU_COUNT` は、utilization_limit と同じ働きを、割合ではなくスレッド数で言う値です。PDB が差し込み先を変えても、設定は残ります。`UTILIZATION_LIMIT` 属

SAMPLE

性は、Resource Manager のコンシューマ・グループの CPU 使用率に、絶対の上限を付ける別の属性です（PDB のディレクティブとは別の話）。[20]

共通ユーザー・ローカル・ユーザー・ロール・権限 [1.4]

共通ユーザーは、CDB ルート（またはアプリケーション・ルート）に作り、そのルートに属するすべての PDB で同じ名前で存在するユーザーです。ローカル・ユーザーは、1 つの PDB にだけ存在します。共通ユーザーの名前は、COMMON_USER_PREFIX（既定は C##）で始めます。ローカルの名前は、この接頭辞で始められません。実機の環境は COMMON_USER_PREFIX が空で、接頭辞の無い名前でも共通ユーザーを作れます。作った共通ユーザーは、全コンテナに複製されます。[21]

権限は、付与のしかたで共通にもローカルにもなります。

	共通に付与	ローカルに付与
書き方	ルートに接続して `GRANT ... CONTAINER=ALL`	そのコンテナに接続して `GRANT ... CONTAINER=CURRENT`
使える範囲	今ある全コンテナと今後のコンテナ	付与したコンテナの中だけ。ルートで付与するとルートだけ
付与できる人	共通ユーザーだけ。受け取るのも共通ユーザー・共通ロール・PUBLIC	共通ユーザーも、ローカル・ユーザーも
注意	PUBLIC には共通に付与しない	PUBLIC には、ローカルならだれでも付与できる

ルートで `GRANT` に `CONTAINER` を省くと、ローカルに付与されます。ルートで `ALTER USER` を流して `CONTAINER` を省くと、既定は `CONTAINER=ALL` です。共通に

SAMPLE

付与するには、付与する人がその権限の付与権（ADMIN OPTION か GRANT OPTION）を共通に持っていなければなりません。ローカルに付与された権限は、その PDB の中でだけ使えます。[21]

```
SQL> grant create table to c##t1_u container=all;
SQL> select grantee, privilege, common, con_id from cdb_sys_privs where
grantee='C##T1_U' and con_id in (1,16);
C##T1_U   CREATE TABLE    YES    1
C##T1_U   CREATE TABLE    YES   16
SQL> alter session set container=t1_arch;
SQL> create user c##t1_v identified by "..." ;          -- PDB 内で C##
の名前
ORA-65094: invalid local user or role name
SQL> grant create session to t1_loc container=all;     -- t1_loc は PDB
のローカル・ユーザー
ORA-65030: cannot grant a privilege commonly to a local user or role
SQL> grant select any table to c##t1_u container=current;
SQL> select grantee, privilege, common from dba_sys_privs where
grantee='C##T1_U' order by privilege;
C##T1_U   CREATE TABLE        YES
C##T1_U   SELECT ANY TABLE     NO
```

ロールも同じ分け方です。共通ロールは CDB ルートかアプリケーション・ルートにあり、そのルートの全 PDB に及びます。ローカル・ロールは 1 つの PDB にだけあり、共通に付与された権限を持ちません。[22]

- 共通ロールの作成・変更・削除は、それぞれ共通に付与された CREATE ROLE・ALTER ROLE・DROP ROLE 権限を持つ共通ユーザーだけが行え

SAMPLE

る。

- ルートで `CREATE ROLE` を流すと、`CONTAINER=ALL` が既定になる。
- CDB の共通ロールの名前は、COMMON_USER_PREFIX の文字（既定 C##）で始める。
- ローカル・ユーザーは共通ロールを作れないが、共通・ローカルのユーザーに付与はできる。
- ローカル・ユーザーやローカル・ロールへの付与は、ローカルにしかできない。共通ロールを PDB でローカルに付与すると、その PDB の中だけで権限が働く。
- Oracle 提供のロール（DBA や PUBLIC など）はすべて共通で、PUBLIC に付けられるシステム権限だけはローカルに付与されている。

```
SQL> create role t1_lr;                    -- PDB 内の通常の名前: ローカル・ロール
Role created.
SQL> create role c##t1_r container=current;   -- PDB 内の C## の名前
ORA-65094: invalid local user or role name
```

PDB ロックダウン・プロファイル [1.4]

PDB ロックダウン・プロファイルは、PDB の利用者が使える機能とオプションを制限する、名前付きの規則の集まりです。PDB 同士で ID（OS・ネットワーク・共通オブジェクト・データベース・リンク）を共有すると、権限が上がることがあります。ロックダウンはそれを防ぎます。[23]

対象は次の 6 つの分類です。

- ネットワーク機能（UTL_TCP や UTL_HTTP など）

SAMPLE

- 共通ユーザーや共通オブジェクトへのアクセス
- OS へのアクセス（UTL_FILE や DBMS_FILE_TRANSFER）
- 接続
- 管理機能（ALTER SYSTEM・ALTER SESSION・ALTER DATABASE）
- データベース・オプション

OS 側の入口は、`PDB_OS_CREDENTIAL`（OS のユーザーを資格情報で指定）と、PDB 作成時の `PATH_PREFIX`・`CREATE_FILE_DEST` で絞ります。[23]

作る手順は 3 ステップです。

1. CDB ルートかアプリケーション・ルートに、CREATE LOCKDOWN PROFILE システム権限を持つユーザーで接続し、`CREATE LOCKDOWN PROFILE 名前` を流す。
2. 同じ場所で `ALTER LOCKDOWN PROFILE` を流して、規則（`DISABLE` ／ `ENABLE` の `STATEMENT`・`FEATURE`・`OPTION`）を足す。外すのも ALTER。
3. 初期化パラメータ `PDB_LOCKDOWN` にプロファイル名を設定する。

```
CREATE LOCKDOWN PROFILE hr_prof;
ALTER LOCKDOWN PROFILE hr_prof DISABLE STATEMENT = ('ALTER SYSTEM');
ALTER LOCKDOWN PROFILE hr_prof ENABLE  STATEMENT = ('ALTER SYSTEM')
CLAUSE = ('FLUSH SHARED_POOL');
ALTER LOCKDOWN PROFILE hr_prof DISABLE FEATURE   = ('XDB_PROTOCOLS');
ALTER SYSTEM SET PDB_LOCKDOWN = hr_prof;
```

`ENABLE ... CLAUSE` は、`DISABLE` で止めた文の一部の句だけを許す書き方です。`CREATE LOCKDOWN PROFILE 新 FROM 元` は、作った時点の元の規則を写すだけで、元の後の変更は追いません。`INCLUDING 元` は、元の `DISABLE STATEMENT`

SAMPLE

の規則と、元の後の変更も受け継ぎます。規則が食い違うと、元（ベース）の規則が勝ちます。Oracle は、PRIVATE_DBAAS・SAAS・PUBLIC_DBAAS の 3 つの空のプロファイルを用意しています。PUBLIC_DBAAS が最も制限が強く、SAAS は PRIVATE_DBAAS より強いです。[23]

`PDB_LOCKDOWN` は、PDB・アプリケーション・ルート・CDB のどこにも設定できます。アプリケーション・ルートに設定すると、その下のアプリケーション PDB のすべてに効きます。CDB に設定すると全 PDB に効きます。再起動は要りません。設定した瞬間から効きます。PDB 自身の設定が、ルートの設定に優先します。PDB が設定を持たなければ、アプリケーション・ルート、さらに CDB ルートの値を引き継ぎます。PDB が CDB のプロファイルを指していれば、ルートのほかの設定は参照されません。CDB のプロファイルとアプリケーションのプロファイルが衝突した時は、CDB の規則が勝ちます。CDB 共通ユーザーが設定できるのは CDB ルートで作ったプロファイルだけです。アプリケーション共通ユーザーが設定できるのは、アプリケーション・ルートで作ったプロファイルだけです。値を空にすると無効になります。[23]

プロファイルは `DBA_LOCKDOWN_PROFILES`（CDB 全体は `CDB_LOCKDOWN_PROFILES`）で見ます。PDB のローカル・ユーザーは、動的ビュー `V$LOCKDOWN_RULES` で規則を見ます。削除は DROP LOCKDOWN PROFILE システム権限で、ルートから流します。実機で、PDB の中の ALTER SYSTEM を止めると、止めた本人も閉め出されます。[23]

SAMPLE

```
SQL> select profile_name, rule_type, rule, clause, status from
cdb_lockdown_profiles where profile_name='T1_PROF';
T1_PROF    FEATURE     XDB_PROTOCOLS                DISABLE
T1_PROF    STATEMENT   ALTER SYSTEM                 DISABLE
T1_PROF    STATEMENT   ALTER SYSTEM  FLUSH SHARED_POOL   ENABLE
SQL> alter session set container=t1_arch;
SQL> alter system set pdb_lockdown=t1_prof;
System altered.
SQL> alter system set open_cursors=222;
ORA-01031: insufficient privileges
SQL> alter system flush shared_pool;
System altered.
SQL> alter system set pdb_lockdown='';
ORA-01031: insufficient privileges
```

PDB の中から `PDB_LOCKDOWN` も戻せなくなります。プロファイルをルートで削除すれば、PDB で値を空にして戻せます。

統合監査と共通・ローカルの監査方針 [1.4]

統合監査の方針は、3 種類あります。[24]

種類	作る場所	対象	含められる物
ローカル監査方針	CDB ルート・アプリケーション・ルート・PDB のどこでも	そのコンテナだけ。共通・ローカルのユーザーに有効	ローカルの物と共通の物の両方（オブジェクトも）

SAMPLE

種類	作る場所	対象	含められる物
CDB 共通監査方針	CDB ルートだけ	全 PDB。共通ユーザーにだけ有効	アクション・システム権限・共通ロール・共通オブジェクトだけ
アプリケーション共通監査方針	アプリケーション・ルートだけ	アプリケーション・コンテナ。アプリケーション共通ユーザーにも CDB 共通ユーザーにも有効	アクション・システム権限・アプリケーション共通のオブジェクトとロール

方針は `CONTAINER` 句で範囲を決めます。

```
CREATE AUDIT POLICY dict_updates
  ACTIONS UPDATE ON SYS.USER$, DELETE ON SYS.USER$
  CONTAINER = ALL;          -- 共通方針
```

- `CONTAINER = CURRENT` は、今の PDB のローカル。既定の範囲もこれ（CDB でもアプリケーションでも）。ルートで省くと `CURRENT` で、ルートだけに効く。
- `CONTAINER = ALL` は共通方針。ルートで使う。PDB で使うとエラー。
- `CONTAINER` 句があるのは `CREATE AUDIT POLICY` だけ。`ALTER`・`DROP` には無いので、範囲を変えるには、削除して作り直す。
- 方針を有効にする `AUDIT POLICY` 文には `CONTAINER` 句を使えない。
- 共通方針は、共通ユーザーにしか有効にできない。ローカル・ユーザーに全コンテナで効かせようとするとエラー。
- ローカル・ユーザーのためのローカル方針は、AUDIT_ADMIN ロールを持つローカル・ユーザーと共通ユーザーが有効にできる。

SAMPLE

- CDB 共通方針を作って管理できるのは、AUDIT_ADMIN ロールを付与された共通ユーザーだけ。作る場所はルートだけ。
- 方針の名前は 128 バイトまでで、ASCII 文字だけ。CDB 共通方針の名前は、COMMON_USER_PREFIX（既定 C##）で始める。アプリケーション共通方針は、アプリケーション・ルートの COMMON_USER_PREFIX（既定は空）に従う。

実機の結果です。この環境は COMMON_USER_PREFIX が空だったので、接頭辞の無い名前でも共通になります。接頭辞の既定の C## を使うなら、先頭に付けます。[24]

```
SQL> create audit policy c##t1_pol actions create table container=all;
SQL> create audit policy t1_pol_all actions create table container=all;
-- 接頭辞なし
SQL> create audit policy t1_pol_cdb actions create table
container=current;
SQL> select policy_name, common from audit_unified_policies where
policy_name like '%T1_POL%' group by policy_name, common;
C##T1_POL     YES
T1_POL_ALL    YES
T1_POL_CDB    NO
SQL> alter session set container=t1_arch;
SQL> create audit policy t1_pol_loc2 actions drop table container=all;
ORA-65040: Operation is not allowed from within a pluggable database.
```

監査の記録は、共通方針でも、イベントが起きた PDB に書かれます。共通の設定はルートの SYS スキーマ、ローカルの設定は PDB の SYS スキーマに置

SAMPLE

かれます。証跡は、その CDB か PDB の SYS か AUDSYS スキーマに保存されます。証跡の見方は、CDB 全体ならルートで `CDB_UNIFIED_AUDIT_TRAIL`、PDB なら `UNIFIED_AUDIT_TRAIL` です。監査の初期化パラメータは CDB の単位にあり、PDB ごとにはありません。OS や XML の監査証跡は、`AUDIT_FILE_DEST`（非推奨）の下の PDB ごとのサブディレクトリです。[24]

TDE の鍵の管理：統一モードと分離モード [1.4]

Transparent Data Encryption（TDE）は、データを暗号化して保存する仕組みです。マルチテナントでは、キーストア（TDE ウォレットか外部のキーストア）を CDB 全体で持つか、PDB ごとに持つかを選びます。[25]

項目	統一モード	分離モード
キーストア	CDB ルートに 1 つ。統一モードの PDB が共有する	PDB ごとに、自分のキーストアとパスワード
鍵の管理	ルートから行う。ただし PDB に固有のマスター暗号化鍵が必要	PDB の中だけで行う
キーストアの種類	ルートで決めた 1 種類	PDB ごとに違う種類にできる
位置づけ	以前のリリースの管理に近い	ルートの設定を上書きする

両方を混ぜて使えます。多くの PDB は統一モードにして、別の種類のキーストアが要る PDB だけを分離モードにします。外部のキーストアは、1 つの CDB で 1 種類しか混ぜられません（サーバーが読める PKCS#11 のベンダー・ライブラリが 1 つだけのため）。分離モードの PDB では、ルート側のキーストアは使えなくなり、その PDB の場所は `WALLET_ROOT/PDB の GUID/tde` です。[25][26]

SAMPLE

最初に一度、初期化パラメータで設定します。

- `WALLET_ROOT`：キーストアのディレクトリ。静的なので、設定後に再起動する。
- `TDE_CONFIGURATION`：キーストアの種類（`FILE`＝TDE ウォレット、Oracle Key Vault）。動的。

どのモードかは `V$ENCRYPTION_WALLET.KEYSTORE_MODE` に出ます。PDB の中では UNITED か ISOLATED、ルートからは全コンテナを見られます。[25]

統一モードでの流れは、ルートでキーストアを作って開き、各 PDB でキーストアを開いて、その PDB のマスター暗号化鍵を作る、です。ルートで `ADMINISTER KEY MANAGEMENT ... CONTAINER = ALL` を流すと、ルートと統一モードの PDB にだけ効き、分離モードの PDB には効きません。分離モードの PDB では、`CONTAINER = ALL` で将来の鍵を全 PDB に作ることと、ルートのキーストアの鍵を PDB のキーストアへ移すことができません。統一モードのキーストアから分離モードのキーストアへ鍵を動かすには、その PDB に接続して `ADMINISTER KEY MANAGEMENT ISOLATE KEYSTORE ... FROM ROOT KEYSTORE` を流します。クローンが元の鍵を使っている時は、`FORCE ISOLATE KEYSTORE` で、動かさず写します。PDB$SEED を分離モードにするのは勧められません。後の PDB の作成やアップグレードに響くからです。[26][27]

26ai では、sqlnet.ora の `ENCRYPTION_WALLET_LOCATION` による設定は使えません。19c と 21c から 26ai へのアップグレードは、元のデータベースに WALLET_ROOT と TDE_CONFIGURATION が無いとエラーになります。TDE の PKI 鍵も、26ai では使えません。キーストアを探す順は 3 つです。1 つ目は WALLET_ROOT の場所で、TDE_CONFIGURATION の KEYSTORE_CONFIGURATION が設定済みの時に使います。2 つ目は sqlnet.ora の設定、3 つ目は `$ORACLE_BASE/admin/db_unique_name/wallet` で

SAMPLE

す。DBCA は、`-configureTDE true` で TDE を設定でき、ソフトウェア・ウォレットだけに対応します。[252]

アプリケーション・コンテナの共通の方針 [1.4]

アプリケーション・コンテナでも、共通ユーザー・ロール・プロファイル・監査方針は、CDB と同じ考え方で作ります。違いは範囲です。アプリケーション・ルートで作った物は、そのアプリケーション・コンテナの中だけが対象です。[824]

- **アプリケーション共通ユーザーと権限**: アプリケーション・ルートで `CONTAINER=ALL` を付けて作る。権限の付与も同じ。CDB 共通ユーザーと同様に、アプリケーション・コンテナのコンテナ間を移れる。
- **アプリケーション共通ロール**: アプリケーション・ルートで作る。名前の接頭辞は、アプリケーション・ルートの COMMON_USER_PREFIX（既定は空）に従う。
- **共通プロファイルの確認**: 共通ユーザー・共通ロール・共通プロファイルかどうかは、DBA_USERS・DBA_ROLES・DBA_PROFILES の COMMON 列で見る。
- **アプリケーション共通監査方針**: アプリケーション・ルートで `CONTAINER = ALL` を付けて作る。監査するオブジェクトは、アプリケーション共通オブジェクトでなければならない。アプリケーションのインストールやアップグレードの作業を監査したい時は、`BEGIN ... END` の間に `CREATE AUDIT POLICY` と `AUDIT POLICY` を入れる。パッチとアンインストールも同じ。ローカルに監査する時は、同期（SYNC）してから効く。

PDB を元にアプリケーション・ルートを作ると、PDB にあったローカルの統合監査方針はアプリケーション・ルートに入ります。共通方針にするには、アプリケーション・ルートで方針を削除して、`CONTAINER = ALL` で作り直します。PDB

SAMPLE

にあったユーザー・ロール・プロファイルは、アプリケーション・ルートを最初に開いた時に共通として印が付きます。`DBMS_PDB.SET_USER_EXPLICIT`・`SET_ROLE_EXPLICIT`・`SET_PROFILE_EXPLICIT` で、アプリケーションに結び付けます。結び付けた後の DDL は、アプリケーションの `BEGIN ... END` の外では流せません。[924]

以前のリリースの PDB をプラグインしてアップグレードする [1.5]

以前のリリースの PDB を新しいリリースの CDB に差し込んで開くと、PDB が自動でアップグレードされます。これをリプレイ・アップグレード（PDB を開く時のアップグレード）と呼びます。PDB と CDB ルートの版が食い違うと、PDB を開いた時に、キャプチャ表に記録された文が再実行されます。キャプチャ表は、移し先のリリースの CDB の CDB$ROOT にあります。[28]

アプリケーションの同期と同じ仕組みです。[142928]

- 既定で有効。データベース・プロパティ `UPGRADE_PDB_ON_OPEN` と `PDB_UPGRADE_SYNC` が true であること。どちらかが false なら、従来のアップグレードが先に要る。
- 無効にするのは `ALTER DATABASE UPGRADE SYNC OFF`（再び有効にするのは `ON`）。CDB ルートに流すと、どの PDB も使わない。ルートが ON で、ある PDB だけ OFF なら、その PDB だけ使わない。
- 無効にした PDB は、AutoUpgrade でアップグレードする。
- 問題があれば、互換性違反（警告かエラー）が記録され、`PDB_PLUG_IN_VIOLATIONS` で見られる。
- 一度に差し込んで上げる PDB の数は、CPU_COUNT の 1/4 までが目安。従来の並列アップグレードは 1/2 まで。

SAMPLE

- アップグレードの後は、datapatch でパッチを当て、AutoUpgrade でアップグレード後の検査をする。

AutoUpgrade は、既定ではリプレイ・アップグレードを使わず、従来の並列アップグレードのスクリプトを使います。リプレイを使わせるには、構成ファイルに `接頭辞.replay=yes` を書きます。[29]

AutoUpgrade は、アンプラグ・プラグのアップグレードを自動で行います。以前のリリースの CDB から PDB をアンプラグして新しいリリースの CDB に差し込み、アップグレードします。手順は 2 つです。[30]

1. 構成ファイルを作り、`java -jar autoupgrade.jar -config ファイル -mode analyze` で準備ができているかを確かめる。
2. `-mode deploy` で、PDB を新しいリリースへ移し、事前の修正（prefixups）を流して、アップグレードまで実行する。

源と移し先の CDB には条件があります。

- 移し先の CDB を先に作って、開いておく。
- エンディアンが同じ。
- 移し先が、源のコンポーネントをすべて含む。
- 文字セットが互換。
- 源のリリースが、移し先へ直接アップグレードできる。
- 両方の CDB で OS 認証が有効。
- APEX のインストール形式が一致する。
- 差し込む非 CDB に、保証付きリストア・ポイントが無い。

AutoUpgrade は Data Guard 構成へも差し込めます。PDB は `STANDBYS=NONE` で作られ、後でスタンバイのデータ・ファイルをリカバリして復

SAMPLE

元します。源 PDB で `SAVE STATE` を流してあれば、その状態も移し先に引き継がれます。構成ファイルの `keep_pdb_save_state` で、保つか捨てるかを決めます。[30]

AutoUpgrade がこの手順を始めた後は、元に戻す選択肢がありません。フラッシュバック・データベースも、PDB の変換をまたいでは使えません。保険はバックアップだけです。[30]

非 CDB を PDB に変える [1.5]

非 CDB の構成は、21c 以降サポートされません。以前のリリースの非 CDB は、PDB に変えて 26ai の CDB に入れます。変えると、ライセンスを追加せずに、ユーザーが構成できる PDB を CDB に 3 つまで持てます。4 つ以上なら、マルチテナントのライセンスが要ります。[29][31]

変換には 2 つの道があります。

道	手順
手作業（リプレイ・アップグレード）	非 CDB を読取り専用にして `DBMS_PDB.DESCRIBE` で XML を作る → `CHECK_PLUG_COMPATIBILITY` → 非 CDB を停止 → `CREATE PLUGGABLE DATABASE ... USING 'XML'`（`COPY` なら元を残せる）→ PDB を `OPEN`（読み書き）
AutoUpgrade	構成ファイルに非 CDB の情報を書き、`target_cdb` に差し込み先の CDB の SID を書く。analyze → deploy

手作業の道では、`ALTER PLUGGABLE DATABASE OPEN` の時に、アップグレードと、非 CDB の辞書から PDB の辞書への変換の両方が起きます。従来の

SAMPLE

`noncdb_to_pdb.sql` のようなスクリプトは流しません。進み具合は `DBA_REPLAY_UPGRADE_ERRORS`（エラー）と `DBA_APPLICATIONS`（完了）で見ます。設定は `UPGRADE_PDB_ON_OPEN` と `CONVERT_NONCDB_ON_OPEN` の 2 つで、どちらもデータベース・プロパティです。`USER_TABLESPACES` 句で、複数の利用者のデータを持っていた非 CDB を、表領域ごとに別の PDB へ分けることもできます。句には PDB に含める表領域を並べます（`ALL EXCEPT` で除く側を書くこともできます）。非 CDB をクローンして PDB を作り、元の非 CDB のバックアップでその PDB を復元したい時は、クローンの前に `DBMS_PDB.EXPORTRMANBACKUP` を流します。このプロシージャを流さないと、バックアップのメタデータは辞書に取り込まれません。アンプラグした PDB には、アンプラグの時に自動で書き出されるため、流す必要がありません。[143132]

AutoUpgrade の道には、決まりがあります。

- 作る PDB は元の非 CDB の名前を使い続け、名前は変えられない。
- 差し込み先の CDB は、先に作っておく。
- 移し先のオプションは、元と同じか、それを含む上位の集合。
- 元の非 CDB と移し先の CDB で、エンディアンが同じ。文字セットは互換。OS 認証が有効。
- 元のリリースと OS が、移し先のリリースへ直接アップグレードできる。
- 元の COMPATIBLE は 12.2.0 以上。それより低くても、変換の途中で 12.2.0 に設定される。
- 元のデータ・ファイルは、既定では写さずに使う。差し込みで写して元を残し、巻き戻しに使いたければ、`target_pdb_copy_option` を設定する。値は `file_name_convert=(...)` で、OMF や ASM なら `file_name_convert=none`。非 CDB の変換では、この値に SID の接尾辞は要らない（複数の PDB を

SAMPLE

同じ作業で再配置やアンプラグ・プラグする時は、PDB ごとに SID を接尾辞に付ける)。

- 始めた後は、非 CDB の状態に戻す選択肢がない。始める前に完全バックアップを取り、よく試験する。

非 CDB を 1 回の作業で変えてアップグレードするほか、元のリリースで PDB に変えてから、新しい CDB に差し込み、差し込みの時にアップグレードする道もあります。[31]

PDB の転送の使い分け [1.5]

PDB を別の CDB や別の場所へ移す方法を、元の PDB の扱いと停止時間で比べます。

方法	元の PDB	停止	向く場面
クローン(ローカル・リモート)	残る	ホット・クローンなら元は動いたまま(ARCHIVELOG とローカル UNDO が要る)	開発・検証用の複製を作る
リフレッシュ可能クローン	残る	更新の間だけ閉じる	本番の複製を、定期的に最新へ更新して持つ。切替え(SWITCHOVER)にも使える
アンプラグとプラグ	CDB から切り離される	元の PDB を閉じる間、停止する	ファイルを持ち運ぶ。ネットワークがつながらない。保管する

SAMPLE

方法	元の PDB	停止	向く場面
再配置（RELOCATE）	移動先の `OPEN` で閉じる	ほとんど無し	停止時間を抑えた移動。元は閉じられ、`MAX` ではトゥームストーン PDB が残る
プロキシ PDB	元に残る	無し	別の CDB の PDB を、リンク越しに参照する

ネットワークで直接つながらない時は、ファイルを運ぶアンプラグとプラグが残ります。まとめて運ぶなら .pdb アーカイブです。プロキシ PDB に、サービスは関連付けません。RMAN の `DUPLICATE` でも、PDB を別の CDB にコピーできます。以前のリリースの PDB は差し込む時にアップグレードされますが、移し先が古いリリースだと差し込めません。自動のダウングレードが無いからです。[612144]

SAMPLE

サンプルはここまでです

続き(第2章〜巻末)は、ORACLE MASTER Gold DBA Oracle AI Database 26ai 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

Footnotes

1. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/understanding-non-cdb-to-pdb-upgrades-autoupgrade.html（Understanding Non-CDB to PDB Upgrades with AutoUpgrade） ↩

2. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/dbca-command.html （createDatabase・createPluggableDatabase）と https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/dbca-overview.html （DBCA Command-Line Syntax Overview・About DBCA Templates） ↩ ↩2 ↩3

3. https://docs.oracle.com/en/database/oracle/oracle-database/26/ladbi/response-files/running-configuration-assistants-using-response-files.html（Running Oracle DBCA Using Response Files） ↩

4. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/relocating-a-pdb.html（About PDB Relocation・Stages of PDB Relocation） ↩ ↩2 ↩3 ↩4 ↩5 ↩6

SAMPLE

5. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/creating-and-configuring-an-oracle-database.html（About CDB Creation with SQL Statements・Creating a CDB Without Using Oracle Managed Files） ↩ ↩2 ↩3

6. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/creating-a-pdb-from-scratch.html（Creating a PDB） ↩ ↩2 ↩3 ↩4 ↩5

7. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/overview-of-pdb-creation.html（PDB File Locations・Table 6-3）と https://docs.oracle.com/en/database/oracle/oracle-database/26/sqlrf/CREATE-PLUGGABLE-DATABASE.html（NO DATA・Restrictions on the NO DATA Clause） ↩ ↩2 ↩3 ↩4 ↩5

8. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/creating-application-containers1.html（About Creating an Application Container） ↩ ↩2

9. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-application-containers-with-sql-plus.html（Application Maintenance・Application Synchronization・Creating Application Common Objects・Migrating an Existing Application） ↩ ↩2 ↩3 ↩4 ↩5 ↩6 ↩7 ↩8 ↩9 ↩10 ↩11 ↩12 ↩13

10. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/creating-application-containers1.html

SAMPLE

（Preparing for Application Containers・Creating an Application Container）↩ ↩2

11. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/removing-a-pdb.html（Unplugging a PDB from a CDB・Dropping a PDB）↩

12. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/cloning-a-pdb.html（About Cloning a PDB・Cloning a Remote PDB・About Refreshable Clone PDBs）↩ ↩2 ↩3 ↩4 ↩5 ↩6 ↩7 ↩8 ↩9 ↩10 ↩11

13. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/removing-a-pdb.html（Unplugging a PDB・Dropping a PDB）↩ ↩2 ↩3 ↩4

14. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/plugging-in-a-pdb.html（About the XML File and Archive File・Plugging In an Unplugged PDB）↩ ↩2 ↩3 ↩4 ↩5 ↩6 ↩7

15. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-pdb-snapshots.html（About PDB Snapshot Carousel・Setting the Maximum Number of Snapshots）↩ ↩2

16. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-a-cdb-with-sql-plus.html（Starting Up a CDB）↩

SAMPLE

17. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-pdbs-with-sql-plus.html（About the Open Mode of a PDB・Preserving or Discarding the Open Mode of PDBs When the CDB Restarts） ↩ ↩2 ↩3 ↩4 ↩5

18. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-pdbs-with-sql-plus.html （Managing Services for PDBs） ↩ ↩2 ↩3

19. https://docs.oracle.com/en/database/oracle/oracle-database/26/multi/administering-a-cdb-with-sql-plus.html（About the Current Container・Switching to a Container Using the ALTER SESSION Statement・Modifying a CDB at the System Level・Setting the Undo Mode in a CDB Using ALTER DATABASE） ↩ ↩2 ↩3 ↩4 ↩5 ↩6 ↩7 ↩8

20. https://docs.oracle.com/en/database/oracle/oracle-database/26/admin/managing-resources-with-oracle-database-resource-manager.html （ Managing Resources with Oracle Database Resource Manager・CDB Resource Plans） ↩ ↩2 ↩3 ↩4 ↩5 ↩6

21. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbseg/managing-commonly-and-locally-granted-privileges.html（About Commonly and Locally Granted Privileges） ↩ ↩2

22. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbseg/managing-common-roles-and-local-roles.html

SAMPLE

（About Common Roles and Local Roles）↩

23. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbseg/restricting-operations-pdbs-using-pdb-lockdown-profiles.html（About PDB Lockdown Profiles・Creating a PDB Lockdown Profile・Enabling or Disabling a PDB Lockdown Profile）↩ ↩2 ↩3 ↩4 ↩5

24. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbseg/auditing-multitenant-deployment.html（About Local, CDB Common, and Application Common Audit Policies・Configuring a Local Unified Audit Policy or Common Unified Audit Policy）↩ ↩2 ↩3 ↩4 ↩5

25. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbtde/introduction-to-transparent-data-encryption.html（Transparent Data Encryption in a Multitenant Environment・Transparent Data Encryption Keystore Search Order）↩ ↩2 ↩3 ↩4

26. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbtde/configuring-isolated-mode2.html（About Configuring Isolated Mode・Operations That Are Not Allowed in an Isolated Mode PDB）↩ ↩2

27. https://docs.oracle.com/en/database/oracle/oracle-database/26/dbtde/configuring-united-mode2.html（Operations That Are Allowed in United Mode）↩

SAMPLE

28. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/choose-an-upgrade-method-for-oracle-database.html ↩ ↩2

29. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/adopting-non-cdb-pbs-using-capture-upgrade.html（Upgrading PDBs Using Replay Upgrade・How to Disable or Enable Replay Upgrade） ↩ ↩2 ↩3

30. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/understanding-pdb-unplug-plug-upgrades-autoupgrade.html（Understanding Unplug-Plug Upgrades with AutoUpgrade） ↩ ↩2 ↩3

31. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/understanding-non-cdb-to-pdb-upgrades-autoupgrade.html（Requirements for Source Non-CDB and Target CDB） ↩ ↩2 ↩3

32. https://docs.oracle.com/en/database/oracle/oracle-database/26/upgrd/adopting-non-cdb-pbs-using-capture-upgrade.html（Adopting and Upgrading a Non-CDB as a PDB with Replay Upgrade） ↩

SAMPLE

ORACLE MASTER Gold DBA Oracle AI Database 26ai 学習用テキスト(無料サンプル)

版

2026年11月版(Oracle Japan「Oracle AI Database 26ai Administration Professional 1Z0-183-JPN」試験内容チェックリスト（5分野・35細目）準拠)

発行

IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ

info@shikaku-doujou.com

Oracle、Java、MySQL および NetSuite は、Oracle、その子会社および関連会社の米国およびその他の国における登録商標です。ORACLE MASTER は、Oracle が日本で提供する認定資格の名称です。記載されているその他の会社名・製品名・サービス名は、各社の商標または登録商標です。

本教材は公式の試験内容チェックリストと Oracle AI Database 26ai の公式文書、Oracle AI Database 26ai Free で実行した結果を基に作成したもので、Oracle およびその子会社・関連会社との提携・承認関係はありません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。

SAMPLE