

IT 資格 道 場

ORACLE MASTER Bronze DBA 26ai 学習用テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

日本オラクル「Bronze DBA Oracle AI Database 26ai Fundamentals (1Z0-185-JPN)」試験内容チェックリスト準拠
2026年10月版

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。

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 Oracleデータベース管理の概要 [1.1/1.2/1.3/1.4/1.5/1.6/1.7/1.8]

リレーショナル・データベースおよびSQLの理解 [1.1]

インスタンスおよびデータベースのコンポーネントの理解 [1.2]

マルチテナント・コンテナ・データベースの利点の理解 [1.3]

データベース・ファイルの理解 [1.4]

Oracleデータベースの管理に使用するツールの理解 [1.5]

Oracleデータベースの重要な用語の理解 [1.6]

Oracle AI Database 26aiの新機能の概要 [1.7]

Oracle AI Database 26aiの特徴と利点 [1.8]

サンプルはここまでです

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	テキストを読んだら次はこれ。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	覚えるべき要点だけを一問一答に凝縮。空き時間の反復と用語の再確認用

使い方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → 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。

この試験が問うもの

ORACLE MASTER Bronze DBA 26ai は、Oracle AI Database 26ai の管理の基本を問う試験です。試験コードは 1Z0-185-JPN で、出題範囲は公式の試験内容チェックリストの 9 大項目・28 小項目です。このテキストの章立ても、公式の大項目と同じ順番です。

項目	内容
運営	日本オラクル (Oracle Corporation)
試験コード	1Z0-185-JPN
問題数	70問
試験時間	120分
合格ライン	65%
出題形式	選択問題
対象のバージョン	Oracle AI Database 26ai

公式は大項目ごとの出題比率を公表していません。

内容は、リレーショナル・データベースと SQL の基本、インスタンスとデータベースの構造、マルチテナント・アーキテクチャ、SQL*Plus と SQL Developer、リスナーと接続、起動・停止とパラメータ、記憶域、ユーザーと権限、表とオブジェクト、バックアップとリカバリ、監視とアドバイザです。例の SQL は、手元の Oracle AI Database 26ai Free で一度実行して、出力を確かめてください。

第1章 Oracleデータベース管理の概要

[1.1/1.2/1.3/1.4/1.5/1.6/1.7/1.8]

リレーショナル・データベースおよびSQLの理解 [1.1]

表・行・列・キーの関係

リレーショナル・データベースは、データを表(テーブル)に入れて管理します。表は行と列でできています。行は1件分のデータで、列は「名前」「給与」のような項目であり、1つの列に別の種類の値を混ぜることはありません。

行を1つに決めるための列が主キー(PK)です。主キーの列には、値の重複もNULLも入りません。別の表の主キーを指す列は外部キー(FK)と呼びます。主キーを持つ側が親表、外部キーを持つ側が子表です。

例として、顧客表 CUSTOMERS (主キーは CUSTOMER_ID) と受注表 ORDERS があるとします。ORDERS の CUSTOMER_ID 列は CUSTOMERS の CUSTOMER_ID を参照するので、外部キーです。この列は ORDERS の主キーではありません。ORDERS の各行は、CUSTOMERS に実在する顧客を指していることが保証されます。ORDERS の行を区別する主キーは、ORDER_ID のような別の列が担います。

外部キーの列は、一意キーでも索引でもありません。一意キーにすると1顧客が1注文しか持てなくなり、業務に合いません。索引は検索を速めるための別の構造で、整合性を守る外部キーとは役割が違います。外部キー列はNULLを許します。顧客が未確定の注文は、CUSTOMER_ID がNULLのまま登録できます。

以降の例では、部門表 DEPT と社員表 EMP を使います。EMP の DEPTNO 列は主キーではありません。DEPT の主キーは DEPTNO で、EMP の DEPTNO

列が DEPT を参照します。

```
create table ch1_dept (deptno number(2) constraint ch1_dept_pk primary
key, dname varchar2(20) not null);
create table ch1_emp (empno number(4) primary key, ename varchar2(20)
unique,
    sal number(7,2) constraint ch1_sal_ck check (sal >= 0),
    deptno number(2) constraint ch1_emp_fk references ch1_dept(deptno));
```

DEPT に 10・20・30 の 3 部門を入れ、EMP には DEPTNO が 10・10・20・NULL・NULL の 5 人を入れます。DEPTNO が NULL の 2 人は、どの部門にも属しません。

制約の種類と違反時の挙動(違反した時のエラー)

制約は、表に入る値のルールで、種類は 5 つです。

制約	禁止すること	違反した時のエラー
NOT NULL	列に NULL を入れること	ORA-01400
UNIQUE	同じ値の重複	ORA-00001
PRIMARY KEY	重複と NULL	ORA-00001 (重複)、ORA-01400 (NULL)
CHECK	条件に合わない値	ORA-02290
FOREIGN KEY	親に無い値の挿入、子が残る親の削除	ORA-02291、ORA-02292

NOT NULL を付けた列に NULL を入れると ORA-01400 になります。UNIQUE や主キーの列に同じ値を入れると ORA-00001 です。CHECK 制約の

条件に合わない行は ORA-02290 で拒否されます。エラーになった行は、表に入りません。

```
ORA-00001: unique constraint (CH1_U.SYS_C008722) violated on table  
CH1_U.CH1_EMP columns (ENAME)  
ORA-02290: check constraint (CH1_U.CH1_SAL_CK) violated  
ORA-01400: cannot insert NULL into ("CH1_U"."CH1_DEPT"."DNAME")
```

外部キーには向きの違う 2 つのエラーがあります。子表へ、親に無い値を入れると ORA-02291 (parent key not found) です。子の行が残っている親の行を削除すると ORA-02292 (child record found) です。DEPT に存在しない部門 99 の社員を EMP へ入れるのが前者、EMP に社員がいる部門 10 を DEPT から消すのが後者です。

```
ORA-02291: integrity constraint (CH1_U.CH1_EMP_FK) violated - parent  
key not found  
ORA-02292: integrity constraint (CH1_U.CH1_EMP_FK) violated - child  
record found
```

ORA-02291 は子側への挿入や更新で出るエラーで、削除では出ません。外部キーに ON DELETE CASCADE を付けると、親の削除で子の行も一緒に消えます。ON DELETE SET NULL を付けると、子の外部キー列が NULL になります。どちらも付けていない外部キーは、自動で子を消すことも NULL にすることもなく、ORA-02292 でエラーになります。

UNIQUE 制約と NULL の関係には、特有の決まりがあります。Oracle の UNIQUE 制約は、キー列がすべて NULL の行を重複とみなしません。ENAME に UNIQUE があり NOT NULL が無い表では、ENAME が NULL の行を何行

でも入れられます。ORA-00001 は出ず、ORA-01400 も出ません。ORA-01400 は NOT NULL 列へ NULL を入れた時のエラーで、ORA-02290 は CHECK 違反のエラーです。複合の UNIQUE は、すべての列が NULL の行だけが制限の対象外です。主キーは NOT NULL を含むので、NULL は入りません。

SQL 文の分類

SQL は働きで 5 つに分けて覚えます。

分類	主な文	働き
DDL	CREATE / ALTER / DROP / TRUNCATE	オブジェクトの定義を作る・ 変える・消す
DML	INSERT / UPDATE / DELETE / MERGE	表のデータを変える
DCL	GRANT / REVOKE	権限を与える・取り上げる
TCL	COMMIT / ROLLBACK / SAVEPOINT	トランザクションを制御する
問合せ	SELECT	データを取り出す

DDL は実行の前後に暗黙のコミットが入ります。DML は COMMIT か ROLLBACK まで確定しません。DCL (GRANT など) も暗黙のコミットを起こします。SELECT は問合せとして別に扱われ、データを変えません。

トランザクションと COMMIT・ROLLBACK・SAVEPOINT

トランザクションは、まとめて確定するか、まとめて取り消す変更の単位です。COMMIT で確定し、ROLLBACK で取り消します。トランザクション全体を終えるのは、COMMIT と、TO の付かない ROLLBACK だけです。

DDL の暗黙のコミットは、表の全行の消去で差が出ます。DELETE は暗黙のコミットを起こしません。DELETE FROM emp は DML なので、同じトランザクションの ROLLBACK で行が戻ります。UPDATE と、別表への INSERT ... SELECT も DML で、COMMIT するまで他のセッションには見えず、ROLLBACK で取り消せます。TRUNCATE TABLE emp は DDL で、実行と同時に暗黙にコミットされます。次のように、ROLLBACK しても行は戻りません。

```
truncate table ch1_emp;  
rollback;  
select count(*) from ch1_emp;
```

COUNT(*)

0

暗黙のコミットは、未コミットの INSERT も一緒に確定します。INSERT の後に CREATE TABLE か CREATE INDEX を実行して ROLLBACK しても、行は残ります。実機で CREATE TABLE の後に ROLLBACK しても、先に入れた行は消えませんでした。SAVEPOINT、SELECT ... FOR UPDATE、ROLLBACK TO SAVEPOINT は、トランザクションを終えず、コミットも起こしません。SELECT ... FOR UPDATE は行をロックするだけで、その後の ROLLBACK で INSERT の行は消えます。SQL*Plus を正常終了した時も、暗黙にコミットされます。

SAVEPOINT は、取り消し位置の目印です。SAVEPOINT 自体は、変更を確定しません。INSERT A、SAVEPOINT s1、INSERT B、SAVEPOINT s2、INSERT C の順に実行して ROLLBACK TO s1 を実行すると、B と C の変更が取り消されます。A の変更は未コミットのまま残り、トランザクションは続きます。自動コミ

ットも起きません。C だけが戻って B が残る動きにもなりません。s1 より後に作った s2 は無効になるため、ROLLBACK TO s2 はエラーになります。

```
rollback to s2
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01086: savepoint 'S2' never established in this session or is  
invalid
```

ROLLBACK TO SAVEPOINT は、そのセーブポイントより後の変更とロックを手放します。

結合・副問合せ・集合演算の読み方

結合は、複数の表の行を条件でつなぐ操作です。内部結合は両方に一致する行だけを返します。外部結合は、一致しない行を捨てません。左外部結合 (LEFT JOIN) は、左の表の全行を残し、右の表に一致する行が無い時は右の表の列を NULL で補います。

社員が 1 人もいない部門を探す問合せを実行します。WHERE で右の表の列が NULL の行を選ぶと、対応する行が無い左の行だけを取り出せます。この書き方をアンチ結合と呼びます。

```
select d.deptno from ch1_dept d left join ch1_emp e on d.deptno =  
e.deptno where e.empno is null;
```

DEPTNO

30

EMP に部門 30 の行はありません。部門 10 には社員が 2 人、部門 20 には社員が 1 人います。EMPNO が NULL になる行は部門 30 だけなので、返るのは 30 の 1 行です。20 や「10 と 30」が返ることはなく、行が返らないこともあります。

副問合せは、問合せの中を書く別の SELECT です。where deptno in (select deptno from ch1_dept) のように、内側の結果を外側の条件に使います。

集合演算は、2 つの問合せの結果行を集合として組み合わせます。

演算	結果
UNION	両方の行を合わせ、重複を取り除く
UNION ALL	両方の行を合わせ、重複を残す
INTERSECT	両方にある行
MINUS	左にあって右に無い行

EMP の DEPTNO (10・10・20・NULL・NULL) と DEPT の DEPTNO (10・20・30) に UNION を使うと、結果は 10・20・30・NULL の 4 行です。UNION では、重複した行は結果に残りません。集合演算では NULL どうしも同じ値として 1 行にまとまります。UNION ALL なら 5 + 3 の 8 行が返ります。EMP だけの 5 行や、NULL を数えない 3 行にはなりません。UNION は重複排除のために並べ替えを伴います。

集合演算が NULL を同じ値とみなすのに対し、WHERE の比較は違います。
`ename = null` の条件はどの行にも一致せず、0 行になります。NULL との比較は `IS NULL` で書きます。

インスタンスおよびデータベースのコンポーネントの理解 [1.2]

インスタンスとデータベースは別なもの

インスタンスはファイルではありません。インスタンスは、メモリー構造の SGA と、バックグラウンド・プロセスの組み合わせです。データベースは、ディスク上のファイル群です。データファイル、制御ファイル、REDO ログ・ファイルがその中心になります。

インスタンスはメモリーとプロセスなので、SHUTDOWN で消えます。データベースはファイルなので、停止しても消えません。データファイルと制御ファイルの組み合わせや、オンライン REDO ログとアーカイブ REDO ログの組み合わせは、ファイルだけなのでインスタンスではありません。パスワード・ファイルとパラメータ・ファイルも、ディスク上のファイルです。起動や認証のために読まれる補助ファイルで、インスタンスの構成要素ではありません。

インスタンスとデータベースは同じものを指す言葉ではありません。STARTUP は段階的に進みます。NOMOUNT ではインスタンスだけが存在します。MOUNT で制御ファイルを開き、OPEN でデータファイルを開きます。1 つのデータベースを複数のインスタンスが開く構成は RAC です。両者を同じものとして扱うことはできません。V\$INSTANCE がインスタンスの状態を、V\$DATABASE がデータベースの情報を返します。

	インスタンス	データベース
構成	SGA とバックグラウンド・プロセス	データファイル・制御ファイル・REDO ログなどのファイル群
置き場所	メモリー	ディスク
停止すると	消える	残る

SGA の主な領域

SGA はインスタンス全体で共有するメモリーです。主な領域は 3 つあります。

- **データベース・バッファ・キャッシュ:** データ・ブロックを保持する。変更されたブロックもここに置かれる
- **共有プール:** 解析済みの SQL 文と実行計画(ライブラリ・キャッシュ)とディクショナリ・キャッシュを保持する
- **REDO ログ・バッファ:** データ・ブロックの変更を REDO レコードとして一時的に保持する

同じ SQL 文を複数のセッションが繰り返し実行する時、解析済みの SQL を保持して再利用するのは共有プールです。バッファ・キャッシュはデータ・ブロックの領域で、REDO ログ・バッファは変更の記録の領域です。解析済みの SQL は、どちらにも置かれません。同じ SQL が共有されると解析が省かれ(ソフト解析)、CPU と共有プールの競合を抑えられます。

変更の記録は REDO ログ・バッファに入ります。REDO レコードを生成してから LGWR がオンライン REDO ログ・ファイルへ書き出すまでの間、そのレコードを保持するのは REDO ログ・バッファです。バッファ・キャッシュが持つのは変更後のデータ・ブロックそのもので、REDO レコードの置き場ではありません。共有プールには変更の記録を置かず、ラージ・プールはバックアップなど

の大きなメモリーを確保する領域です。REDO ログ・バッファは SGA の中の循環バッファで、V\$SGAINFO では Redo Buffers として見えます。

実機の V\$SGAINFO には、Buffer Cache Size、Shared Pool Size、Large Pool Size、Redo Buffers、Fixed SGA Size などの項目が並びます。

PGA とサーバー・プロセス

PGA は、サーバー・プロセスごとに確保される専用メモリーです。PGA は SGA の一部ではありません。ORDER BY の並べ替えやハッシュ結合の作業領域、カーソルの状態を持ちます。SGA の一部ではなく、他のプロセスと共有もされません。バッファ・キャッシュ・共有プール・ラージ・プールは、どれも全プロセスが共有する SGA 側の領域であり、PGA には含まれません。専用サーバー接続では、セッション情報 (UGA) も PGA に置かれ、全サーバー・プロセスの PGA 合計の目標値は PGA_AGGREGATE_TARGET が決めます。

専用サーバー構成では、接続ごとに専用のサーバー・プロセスが 1 つ起動し、そのプロセスが自分専用の PGA を持ちます。100 セッションが接続していれば、サーバー・プロセスも PGA も 100 個になり、ユーザーの SQL を解析・実行して結果を返すのは、そのサーバー・プロセスです。共有サーバー構成ではディスパッチャ経由で少数のサーバー・プロセスを共有します。実機の SHARED_SERVERS は 0 で、専用サーバーしか使われません。

主なバックグラウンド・プロセス

プロセス	役割
DBWn	ダーティ・バッファ (更新済みでデータファイルに未反映のバッファ) をデータファイルへ書き出す

プロセス	役割
LGWR	REDO ログ・バッファをオンライン REDO ログ・ファイルへ書き出す
CKPT	チェックポイント情報を制御ファイルとデータファイル・ヘッダーへ記録する
SMON	起動時に必要ならインスタンス・リカバリを行う。不要な一時セグメントも回収する
PMON	異常終了したプロセスの後始末（ロックの解放など）をする
ARCn	ARCHIVELOG モードで、満杯のオンライン REDO ログをアーカイブ先へコピーする

実機の V\$BGPROCESS では、DBW0・LGWR・CKPT・SMON・PMON・MMON が動いています。ARCn は動いていません。実機は NOARCHIVELOG モードだからです。

紛らわしいのは、書き出す相手の違いです。ダーティ・バッファをデータファイルへ書くのは DBWn で、LGWR ではありません。REDO を REDO ログへ書くのは LGWR で、DBWn ではありません。CKPT が記録するのはチェックポイント情報だけで、データの書き出しは DBWn が行います。ARCn はバッファ・キャッシュへブロックを読み込みません。ブロックの読み込みはサーバー・プロセスが行い、ARCn はアーカイブを担当します。

DBWn は、バッファが不足した時、チェックポイント時、一定時間ごとなどに書き出します。コミットと同期して書く必要はありません。

SQL 文の処理の流れとコミット

SQL の処理は、解析、実行、フェッチの順に進みます。フェッチは SELECT だけの段階で、結果行をクライアントへ返します。INSERT や UPDATE のような DML にフェッチはありません。

解析では、構文と権限を検査し、共有プールに同じ SQL が既にあるかを調べます。あれば再利用でき(ソフト解析)、実行計画を作る完全な解析(ハード解析)を省けます。無ければハード解析になり、負荷が高くなります。共有プールは解析の段階から参照されます。解析は、結果行を返しません。結果行を返すのはフェッチです。バインド変数を使うと、値が違っても同じ SQL として共有されやすくなります。

COMMIT を発行すると、LGWR がそのトランザクションの REDO レコードをオンライン REDO ログ・ファイルへ書き出します。書き出しが終わった時点で COMMIT は完了します。データファイルへの書き出しは COMMIT の完了条件ではありません。変更されたデータ・ブロックは、DBWn が後でまとめて書き出します。この順序をログ先行書き込みと呼びます。コミット時に同期して書かれるのは REDO だけで、データファイルへの書き出しはコミットの時点で待ちません。

アーカイブ REDO ログは、満杯になったオンライン REDO ログを ARCn がコピーしたものです。COMMIT の時に書かれるのは、オンライン REDO ログです。REDO ログ・バッファも、COMMIT の完了では破棄されず、循環して再利用されます。LGWR は COMMIT 時のほか、バッファが 3 分の 1 まで埋まった時、DBWn が書く前、3 秒ごとにも書き出します。停電しても REDO が残っていれば、起動時のインスタンス・リカバリでコミット済みの変更を再実行できるので、変更は失われません。

マルチテナント・コンテナ・データベースの利点の理解 [1.3]

CDB\$ROOT・PDB\$SEED・PDB

マルチテナント構成のデータベースをコンテナ・データベース (CDB) と呼び、CDB の中に入るデータベースをプラガブル・データベース (PDB) と呼びます。CDB は、ルート 1 つ、シード 1 つ、ユーザー PDB 0 個以上で構成されます。

- **CDB\$ROOT**: Oracle が管理するメタデータと共通ユーザーを持つルート。業務データは置かない
- **PDB\$SEED**: 新しい PDB を作る時の雛形で、READ ONLY で開かれている
- **PDB**: 業務データを入れる。実機では FREEPDB1 が該当する

実機の V\$PDBS には、次の 2 行が出ます。

CON_ID	NAME	OPEN_MODE
2	PDB\$SEED	READ ONLY
3	FREEPDB1	READ WRITE

業務データを格納して READ WRITE で開かれているのは FREEPDB1 です。PDB\$SEED は新しい PDB を作る時のテンプレート (雛形) で、削除した PDB のデータファイルの退避先やバックアップ領域ではありません。CDB 全体の共通ユーザーとメタデータを保持するのは、ルートの CDB\$ROOT です。CDB\$ROOT (CON_ID 1) は V\$PDBS には出ず、V\$CONTAINERS には出ます。

PDB を使う利点

PDB は、アプリケーションから見ると独立した従来型のデータベースに見えます。複数の PDB が 1 つの CDB のインスタンス (SGA とバックグラウンド・プロセス) を共有するので、データベースごとにインスタンスを持つ場合より、メモリとプロセスの使用量を抑えられます。インスタンスは CDB に 1 つで、PDB ごとには起動しません。部署ごとに別サーバーで動かしていた 10 個の小さなデータベースを 1 つの CDB に統合すれば、サーバー台数とメモリ消費が減ります。

PDB ごとに独立した SGA が割り当てられることはなく、ほかの PDB の負荷の影響を完全に避けられるわけでもありません。Oracle ホームも CDB で 1 つで、PDB ごとには持ちません。データファイルと表領域は PDB ごとに別で、1 つの表領域へ統合されることもありません。Resource Manager を使うと、PDB ごとの CPU などを制限できます。

パッチ適用の利点は、CDB に 1 回適用すれば配下の PDB が同じ版で動く点です。40 個の PDB があっても、台数分の作業を繰り返す必要はありません。ただし、1 つの CDB 内で PDB ごとに別の版にはできません。ソフトウェアの更新は CDB 側に必要で、PDB のデータファイルだけが対象ではありません。適用中は再起動などで接続が中断することがあります。適用後に PDB を開いたら、ディクショナリの更新処理 (datapatch) が必要です。パッチ済みの別の CDB へ PDB を移せば、PDB ごとに適用時期をずらせます。バックアップと監視も、CDB 単位でまとめて扱えます。PDB ごとの手順は要りません。

プロビジョニングも速くなります。新しい PDB は、シードやクローンから短い手順で作れるからです。

共通ユーザーとローカル・ユーザー

CDB のユーザーは 2 種類です。共通ユーザーは CDB のすべてのコンテナ (シードを含む) に同じ名前で存在します。ローカル・ユーザーは、1 つの PDB の外には存在しません。共通ユーザーの名前は、既定の接頭辞 C## で始めます。SYS と SYSTEM も共通ユーザーで、DBA_USERS の COMMON 列は YES です。

共通ユーザーは PDB 内では作れません。CDB\$ROOT で作り、CONTAINER=ALL (既定) で全コンテナに存在させます。ルートで作った C##APP1 は、PDB (FREEPDB1) に接続して DBA_USERS を見ても、COMMON 列が YES で表示されます。CDB_USERS を見ると、C##APP1 と SYS は COMMON が YES で、PDB 内で作ったローカル・ユーザーは NO です。PDB 側で同名のローカル・ユーザーを作り直す必要はありません。SYS と SYSTEM だけが共通ユーザーなのではなく、ルートで作ったユーザーはすべて該当します。COMMON が NO になるのは、PDB 内で作ったローカル・ユーザーです。

USERNAME	COM
SYS	YES
C##CH1_APP1	YES
CH1_U	NO

名前の決まりを破ると、場所ごとに別のエラーになります。PDB に SYS で接続して `CREATE USER c##app2 IDENTIFIED BY x;` を実行すると、ORA-65094 (invalid local user or role name) でエラーになります。PDB 内で共通ユーザーは作れません。PDB 内で作れるのはローカル・ユーザーだけで、C## で始まる名前は共通ユーザー用だからです。PDB 内で共通ユーザーが作れるわけではなく、成功してローカル・ユーザーになることもありません。逆に、ル

ートで C## を付けない名前を使うと ORA-65096 です。PDB 内で CONTAINER=ALL を付けた共通 DDL は、ORA-65050 (共通 DDL はルートのみ) になります。

```
ORA-65094: invalid local user or role name
ORA-65096: common user or role name must start with prefix C##
ORA-65050: Common DDLs only allowed in root.
```

ORA-65096 は ORA-65094 の逆で、ルートで接頭辞に合わない名前を使った時のエラーです。PDB でログインするには、PDB 側で CREATE SESSION を付与しなければなりません。

PDB の作成方法

PDB の作り方は 3 つあります。

方法	内容
シードから	PDB\$SEED を雛形に、管理者ユーザー付きの空の PDB を新規作成する
クローン	既存の PDB (ローカルまたはリモート) をデータごと複製する
アンプラグとプラグ	PDB を XML マニフェストと共に切り離し、別の CDB へ取り込む

シードから作る PDB は空の雛形で、本番のデータもオブジェクトも入りません。実機でも `CREATE PLUGGABLE DATABASE ... ADMIN USER` で作成できます。本番 PDB のデータを含めて検証用 PDB を作りたい時は、`CREATE PLUGGABLE DATABASE ... FROM 本番PDB名` でクローンを作ります。構文は FROM 句です。`ALTER PLUGGABLE DATABASE ... CLONE` という文はありません。アンプラグは元

の PDB を CDB から切り離す操作で、元の場所で開いたままコピーを作るものではありません。

アンプラグは PDB をクローズした状態で行い、メタデータを XML マニフェスト・ファイルへ書き出します。プラグは、その XML ファイルを指定した `CREATE PLUGGABLE DATABASE ... USING` で、別の CDB へ取り込む操作です。アンプラグしても、データファイルはその場で削除されません。再プラグに使えます。アンプラグ後の PDB は CDB に残りますが開けないので、`DROP PLUGGABLE DATABASE ... KEEP DATAFILES` でデータファイルを残して切り離します。プラグ時のデータファイルの扱いは、COPY (既定)、MOVE、NOCOPY から選べます。プラグ先は、同じ版でなければならないわけではありません。新しい版の CDB へ取り込んで、そこでアップグレードする方法があります。プラグは、PDB\$SEED からの新規作成ではありません。XML の記述に従って既存のデータファイルを取り込む操作です。

26ai は CDB だけ

Oracle AI Database 26ai は、マルチテナントの CDB 構成だけをサポートします。非 CDB 構成は使えません。新しく作成するデータベースは CDB になり、DBCA で 非 CDB を 選 ぶ こ と は で き ま せ ん 。 実 機 の `ENABLE_PLUGGABLE_DATABASE` は TRUE で、`V$DATABASE` の CDB 列も YES です。

19c の非 CDB を 26ai へ移行する場合は、PDB へ変換して CDB に取り込みます。19c の PDB を非 CDB に戻す必要はありません。CDB が唯一のアーキテクチャになったのは 21c 以降で、23ai と 26ai もその流れを引き継ぎます。アプリケーションのデータは PDB に置き、ルートには置きません。PDB が非推奨になったわけでもありません。

データベース・ファイルの理解 [1.4]

データファイル・制御ファイル・REDO ログ・ファイル

データベースは、役割の違うファイルで構成されます。

ファイル	役割
データファイル	表や索引などのデータそのものを格納する
制御ファイル	データベース名、データファイルと REDO ログの場所、チェックポイント情報などの物理構造を記録する
オンライン REDO ログ・ファイル	データベースに加えた変更の記録 (REDO) を保持する。グループを循環して使う
一時ファイル	ソートなどの作業用で、コミット済みの変更の記録は持たない

制御ファイルは MOUNT の段階で最初に開かれます。STARTUP は、NOMOUNT (パラメータ・ファイルを読んでインスタンスを起動)、MOUNT (制御ファイルを開く)、OPEN (データファイルと REDO ログを開く) の順に進みます。制御ファイルが失われると MOUNT できません。データファイルは表などのデータそのもの、パスワード・ファイルは特権ユーザーの認証用、パラメータ・ファイルはインスタンスの設定です。MOUNT で最初に読むのは、そのどれでもなく制御ファイルです。

サーバーの電源が突然落ち、コミットした変更がデータファイルに書かれていなくても、復旧に使うのはオンライン REDO ログ・ファイルです。再起動時のインスタンス・リカバリで、REDO からコミット済みの変更を再適用します。制御ファイルは構造情報だけで、変更の履歴は持ちません。実機の V\$LOGFILE には redo01.log と redo02.log が並び、V\$LOG ではグループ 1 が CURRENT、

グループ 2 が INACTIVE などの状態になります。グループが満杯になって次のグループへ移ることを、ログ・スイッチと呼びます。

パラメータ・ファイル (SPFILE と PFILE)

パラメータ・ファイルは 2 種類しかありません。

	SPFILE	PFILE
形式	バイナリ	テキスト
置き場所	通常はサーバー側	任意
変更の仕方	ALTER SYSTEM で書き込む。エディタでは直接編集しない	エディタで編集する

SPFILE はサーバー側のバイナリ・ファイルです。ALTER SYSTEM で SCOPE=SPFILE か BOTH を指定すれば、インスタンスを止めずに、変更を次回起動後も保持できます。SPFILE をエディタで直接編集することはありません。PFILE はテキスト・ファイルで、エディタで編集します。

起動時に読まれるのは、SPFILE と PFILE のどちらか 1 つです。両方を読んで値を合算したり、PFILE の値を優先したりはしません。既定の検索順は spfile.ora、spfile.ora、init.ora です。STARTUP PFILE=... と指定すれば、PFILE を明示して起動できます。CREATE PFILE FROM SPFILE でテキストの PFILE を書き出し、CREATE SPFILE FROM PFILE で SPFILE を作れます。

SQL*Plus で SHOW PARAMETER spfile を実行した時に、パスが表示されたなら、SPFILE で起動しています。PFILE で起動した場合は、パスが表示されません。

```
spfile string
```

```
/opt/oracle/product/26ai/dbhomeFree/dbs/spfileFREE.ora
```

アーカイブ REDO ログ・パスワード・ファイル・アラート・ログ

アーカイブ REDO ログは、満杯になったオンライン REDO ログのコピーです。ARCHIVELOG モードでは、満杯のグループを ARCn がアーカイブ先へコピーしてから、そのグループを再利用します。V\$DATABASE の LOG_MODE が NOARCHIVELOG の時は、満杯のグループはアーカイブされずに上書きされ、再利用されます。SQL*Plus の ARCHIVE LOG LIST でも、このモードを確かめられます。

Database log mode	No Archive Mode
Automatic archival	Disabled

アーカイブされないからといって、REDO が書かれないわけではありません。NOARCHIVELOG でも、オンライン REDO ログは常に書かれ、インスタンス・リカバリにも使えます。アーカイブだけに変更が書かれることはありません。ARCn によるコピーは ARCHIVELOG モードの動作です。NOARCHIVELOG ではクローズ(コールド)バックアップしか使えず、直前のバックアップまでしか戻せません。

パスワード・ファイルは、SYSDBA などの管理権限を持つユーザーが、データベースが開いていない状態でもリモートから認証されるために使われます。REMOTE_LOGIN_PASSWORDFILE が EXCLUSIVE の時に使われます。名前は orapw で、実機の dbs ディレクトリには orapwFREE があります。作成には ORAPWD コマンドを使います。アラート・ログは診断メッセージを時系列に残すテキストで、認証にも復旧にも使いません。場所は V\$DIAG_INFO の Diag

Trace と Diag Alert で確認できます。制御ファイルは物理構造の記録で、パスワードを保持しません。

制御ファイルと REDO ログの多重化

制御ファイルは、複数のコピーを作って別のディスクに置くことが推奨されます。多重化した制御ファイルは同じ内容に保たれ、データベースはそれらを同時に更新します。世代違いの内容が保持されることも、多重化した数だけデータファイルのコピーが作られることもありません。実機の V\$CONTROLFILE には、control01.ctl と control02.ctl の 2 つが返ります。

CONTROL_FILES は静的パラメータです。ALTER SYSTEM SET CONTROL_FILES ... SCOPE=MEMORY は ORA-02095 でエラーになります。追加するには、CONTROL_FILES を SPFILE へ変更し、インスタンスを停止してファイルをコピーし、再起動します。どれか 1 つが欠けると、インスタンスは起動できません。内容が同じ限り、残ったコピーから復旧できます。稼働中のインスタンスが制御ファイルを 1 つ失うと、次にそのファイルへ書き込む時にインスタンスが停止します。残りのコピーがあっても、動き続けることはありません。

データベースには、オンライン REDO ログのグループが最低 2 つ必要です。オンライン REDO ログは、グループ内に複数のメンバーを作って別のディスクに置くと多重化できます。LGWR は同じ内容をグループ内の全メンバーへ同時に書き込みます。1 つのメンバーを失っても、残りのメンバーがあればグループは使えなくなりません。メンバーごとに内容が分かれることはなく、メンバーへの書き込みを ARCn が行うこともありません。メンバーを足してもグループの数は変わらず、同じグループのメンバーは同じサイズ・同じ内容です。追加は ALTER DATABASE ADD LOGFILE MEMBERで行います。実機の V\$LOG では MEMBERS が 1 のグループしかなく、多重化されていません。実運用では少なくとも 2 メンバーが推奨されます。

Oracleデータベースの管理に使用するツールの理解 [1.5]

ツールと用途の対応

管理に使う主なツールは、用途で覚えます。

ツール	用途
SQL*Plus	SQL と管理コマンドを対話的に実行する
SQL Developer	GUI の開発・管理ツール
RMAN	バックアップ、リストア、リカバリ
lsnrctl	リスナーの起動・停止・状態確認
DBCA	データベースの作成・削除・構成変更
ORAPWD	パスワード・ファイルの作成
SQL*Loader	外部ファイルのデータを表へ取り込む

データベース全体のバックアップを取り、障害時に復旧するコマンドライン・ツールは RMAN (Recovery Manager) です。lsnrctl はバックアップを担当せず、リスナーを制御します。DBCA はバックアップのツールではありません。DBCA はデータベースを作成するツールで、ORAPWD はパスワード・ファイルを作るツールです。

リスナーが起動しているかを確認し、必要なら停止するコマンドは lsnrctl (リスナー制御ユーティリティ) です。lsnrctl start、lsnrctl stop、lsnrctl status、lsnrctl services がよく使う操作です。RMAN や DBCA、SQL*Loader では、リスナーは操作できません。実機で lsnrctl status を実行すると、バージョン、起動時刻、待ち受けアドレス (tcp のポート 1521 など)、登録されたサービスの一覧が表示されます。リスナーが止まっても、既存のセッションは継続します。

OS 認証と SYSDBA・SYSOPER

OS 認証は、データベース・サーバー上で、OS ユーザーが OSDBA グループ (Linux では通常 dba) に属していれば、パスワードなしで AS SYSDBA 接続できる方式です。OSOPER グループが SYSOPER に対応します。

```
sqlplus / as sysdba
```

OS 認証は、ネットワーク経由の接続ではありません。ローカルのサーバー上で行う接続方式で、ネットワーク経由の接続とは別です。パスワード・ファイルは使わず、OS のグループで判断します。OS ユーザーと同名のデータベース・ユーザーも、パスワード・ファイルへの登録も要りません。パスワードを指定しないので、インスタンスが停止している状態でも AS SYSDBA で接続して STARTUP を実行できます。パスワード・ファイルが使われるのは、リモートから SYSDBA で接続する時です。

AS SYSDBA で接続したセッションは SYS スキーマとして動きます。SYSDBA は権限の名前で、スキーマ名は SYS です。SYSOPER 接続は SYS スキーマになりません。AS SYSOPER で接続すると PUBLIC スキーマとして動きます。

```
SQL> connect / as sysdba
SQL> show user
USER is "SYS"
SQL> connect / as sysoper
SQL> show user
USER is "PUBLIC"
```

AS SYSDBA で接続すると SHOW USER は SYS を、AS SYSOPER で接続すると PUBLIC を返します。

権限	SHOW USER	できること
SYSDBA	SYS	SYSOPER の権限に加え、 CREATE DATABASE、リカバリ、ユーザー管理など
SYSOPER	PUBLIC	STARTUP / SHUTDOWN、 ALTER DATABASE OPEN・ MOUNT、バックアップなどの運用

CREATE DATABASE のようなデータベースの作成は、SYSDBA にしか認められていません。SYSOPER の権限は、起動・停止やバックアップなどの運用に限られます。

Oracleデータベースの重要な用語の理解 [1.6]

記憶域の包含関係

論理的な記憶域は、大きい単位から順に、表領域、セグメント、エクステント、データ・ブロックです。

単位	内容
表領域	複数のセグメントを持つ論理的な入れ物
セグメント	表や索引などのオブジェクトに対応する。1つ以上のエクステントで構成される
エクステント	連続する複数のデータ・ブロック
データ・ブロック	最小の単位。サイズは DB_BLOCK_SIZE (既定 8 KB)

エクステントはセグメントを構成する単位なので、セグメントより小さく、ブロックより大きい単位です。表領域の方がセグメントより大きく、ブロックの方がエクステントより小さい単位です。スキーマは記憶域の階層に入りません。スキーマは、ユーザーが所有するオブジェクトの集まりです。記憶域の階層とは別の論理構造で、この並びには入りません。

セグメントの大きさは USER_SEGMENTS で調べます。BYTES は BLOCKS にブロック・サイズを掛けた値です。実機で小さな表を作ると、USER_SEGMENTS には次の 1 行が出ます。

SEGMENT_NAME	SEGMENT_TYPE	EXTENTS	BLOCKS	BYTES
CH1_T	TABLE	1	8	65536

BLOCKS が 8 で 1 ブロックが 8 KB (8192 バイト) なので、 $8 \times 8192 = 65536$ バイトです。8192 は 1 ブロック分、524288 は 64 ブロック分、1048576 は 128 ブロック分の大きさで、このセグメントとは合いません。ローカル管理表領域では、小さなセグメントの最初のエクステントは 64 KB (8 KB ブロックで 8 ブロック) です。

SID・サービス名・DB_NAME

名前の種類は、指すものが違います。

名前	指すもの
DB_NAME	データベース自体の名前。制御ファイルに記録され、起動時に参照される。8 文字以内
INSTANCE_NAME / ORACLE_SID	インスタンスを識別する名前。ORACLE_SID は OS の環境変数
サービス名 (SERVICE_NAMES)	クライアントが接続時に指定する名前

名前	指すもの
DB_UNIQUE_NAME	データベースを一意に識別する名前。Data Guardなどで区別に使う

データベース自体の名前を表すのは DB_NAME です。INSTANCE_NAME はインスタンスの名前で、SERVICE_NAMES は接続に使うサービスの名前です。LOCAL_LISTENER はリスナーの場所を指定するパラメータで、名前ではありません。実機では次の 3 つがたまたま同じ値です。

db_name	string	FREE
instance_name	string	FREE
service_names	string	FREE

役割は別です。DB_NAME は 1 つのデータベースにつき 1 つで、インスタンスごとに別の値は持てません。RAC のように 1 つのデータベースを複数のインスタンスが開く構成では、INSTANCE_NAME がインスタンスごとに異なります。サービス名は、1 つのデータベースに複数定義できます。SERVICE_NAMES は、データファイルの保存先ディレクトリの名前を決めません。ORACLE_SID と DB_NAME は、異なる値にしても起動できます。

CDB では、V\$SERVICES でサービスと接続先のコンテナを確認できます。

NAME	PDB
SYS\$BACKGROUND	CDB\$ROOT
SYS\$USERS	CDB\$ROOT
free	CDB\$ROOT
freeXDB	CDB\$ROOT
freepdb1	FREEPDB1

freepdb1 サービスを指定して接続すると、PDB の FREEPDB1 につながります。free サービスは PDB 列が CDB\$ROOT なので、CDB\$ROOT につながります。FREEPDB1 ではありません。SYS\$USERS は、サービス名を使わない接続（OS 認証など）に使われる既定のサービスで、FREEPDB1 専用ではありません。freeXDB は XML DB 用のサービスで、データファイルの名前ではありません。SYS\$BACKGROUND は、バックグラウンド・プロセス用の内部サービスです。PDB を作成すると、その PDB 名のサービスが自動で作られます。CDB\$ROOT へ接続したい時は、CDB のサービス（この環境では free）を使います。

Oracle AI Database 26aiの新機能の概要 [1.7]

AI Vector Search と VECTOR 型

AI Vector Search は、文章や画像の意味を数値の並び（ベクトル）に変えて保存し、意味の近いデータを探す機能です。ベクトルは文字列型の列には入れません。ベクトルを入れる列のデータ型が VECTOR 型で、`VECTOR(次元数, 要素の型)` で定義します。

```
create table ch1_items (id number, v vector(3, float32));
```

VECTOR(3, FLOAT32) の列は、3 次元のベクトルだけを受け付けます。'[1,0,0]' のような文字列は VECTOR 型へ変換されて挿入できます。2 次元の値を入れようとすると、次元数の不一致で ORA-51803 になります。

```
insert into ch1_items values (4, '[1,2]')
```

```
ORA-51803: Vector dimension count must match the dimension count
specified in the column definition (expected 3 dimensions, specified 2
dimensions).
```

文字列から VECTOR 型への変換はできるので、失敗の原因は次元数の不一致しかありません。DESCRIBE では VECTOR(3, FLOAT32, DENSE) と表示されます。

類似検索は、VECTOR_DISTANCE 関数で距離を求めて並べます。距離が小さいほど似ていて、距離が大きい行を似ているとは読みません。ID が 1・2・3 の行にそれぞれ [1,0,0]・[0,1,0]・[0.9,0.1,0] を入れ、[1,0,0] に近い 2 行を探します。

```
select id from ch1_items order by vector_distance(v, '[1,0,0]', cosine)
fetch first 2 rows only;
```

ID

1

3

ORDER BY の既定は昇順なので、距離の小さい行から並びます。基準と同じ ID 1 の距離は 0 で、ID 3 の向きが次に近く、ID 2 は基準と直交するので最も遠い行です。コサイン距離はベクトルの向きの違いを測り、同じ向きなら 0、直交なら 1 です。実機の距離は ID 3 が約 0.0061、ID 2 が 1 でした。距離の指標は COSINE、EUCLIDEAN、EUCLIDEAN_SQUARED、DOT、MANHATTANなどを指定できます。実機でベクトル [1,2] と [4,6] の距離を測ると、EUCLIDEAN は 5、EUCLIDEAN_SQUARED は 25、MANHATTAN は 7、DOT は -16 になります。MANHATTAN は成分ごとの差の絶対値の合計 ($|1-4| + |2-6| = 7$) で、EUCLIDEAN_SQUARED は差の 2 乗の合計、DOT は内積に負号を付けた値です。ベクトル索引が無くても、エラーにはなりません。厳密な検索として実行されます。FETCH APPROXIMATE FIRST ... を使うと、ベクトル索引による近似検索になります。

開発者向けの機能

JSON リレーショナル二面性ビューは、正規化された複数の表のデータを、1 つの JSON ドキュメントとして公開するビューです。二面性ビュー自体はデータを持ちません。同じデータを JSON (ドキュメント) と表 (リレーショナル) の両方の見方で扱えます。JSON で読み書きした内容は、元の表へ分解されて反映されます。整合性は表の制約が保ちます。JSON 型の列を持つ表は JSON を格納するだけで、表とドキュメントを結ぶビューにはなりません。マテリアライズド・ビューの高速リフレッシュは更新方式で、JSON の読み書きとは別です。AI Vector Search は類似検索の機能です。

SQL ドメインは、データ型、制約、既定値、表示形式などの属性に名前を付けて定義し、複数の表の列で再利用する機能です。同じ検証を何度も書かずに済み、定義を 1 か所に変更できます。例を実機で実行しました。

```
create domain ch1_d_age as number(3) check (ch1_d_age between 0 and 150);  
  
create table ch1_person (age ch1_d_age);  
  
insert into ch1_person values (200);
```

```
ORA-11534: check constraint (CH1_U.SYS_C008724) involving column AGE  
due to domain constraint CH1_U.SYS_DOMAIN_C0043 of domain  
CH1_U.CH1_D_AGE violated
```

SQL ドメインは、PDB を束ねるグループではありません。PDB の集まりは CDB です。表領域を作る時にデータファイルの場所を指定する仕組みでもなく、ホスト名からサービスを探す名前解決の仕組みでもありません。CREATE DOMAIN の実行には権限が必要で、権限のないユーザーは ORA-01031 になります。

23ai から 26ai への名称変更と提供形態

Oracle AI Database 26ai は、Oracle Database 23ai に代わる長期サポート・リリースで、製品名が Oracle AI Database になりました。[^1-1]23ai からは、リリース・アップデートの適用で移行できます。データベースのアップグレードも、アプリケーションの再認定も要らないとされています。26ai は 23ai の更新版なので互換性は保たれ、データを新しい構造へ作り直す必要はありません。AI Vector Search も 26ai の主要な機能として含まれ、別の製品として提供されることはありません。SQL と PL/SQL の既存の API も維持され、アプリケーションの再設計は必須ではありません。

実機で使っている Free 版は、イメージ名が 23ai Free のままでも、バージョンは 23.26.3 で、ソフトウェアの配置先は 26ai です。

26ai には、開発や学習に使える無償の Free 版があります。Free 版は CPU とメモリー、ユーザー・データの容量に上限があり、Enterprise Edition の全機能を制限なく使える版ではありません。商用の本番利用には、サポート付きのエディションを使います。26ai はクラウドとオンプレミスの両方で利用できます。Oracle Cloud Infrastructure だけの製品でも、Exadata 専用の製品でもなく、Exadata 以外の汎用のサーバーにも導入できます。

Oracle AI Database 26aiの特徴と利点 [1.8]

AI データ処理をデータベース内で行う利点

業務データを Oracle に持つ企業が商品説明の意味検索を追加する時、毎晩データを外部のベクトル専用データベースへコピーする案があります。外部へコピーする案では、権限の管理を二重にしなければなりません。データベース内で AI Vector Search を使う案なら、データを複製・移動せずに、既存のアクセス制御、暗号化、トランザクションのもとで検索できます。外部へのコピーでは、同期の遅れ、権限の二重管理、データの漏えい経路の増加が課題になります。

ベクトル列は他の列と同様に、暗号化と監査の対象です。ベクトルが対象から外れて処理が軽くなることはなく、別のデータベース製品への自動複製が行われるわけでもありません。AI Vector Search は SQL の一部として使えるので、SQL と組み合わせられないという説明も誤りです。既存のバックアップ、高可用性、監査の仕組みも、そのまま効きます。

商品表の説明文をベクトルに変換して同じデータベースに保存し、顧客の質問に近い商品を探して生成 AI への入力に使う構成を RAG と呼びます。RAG は、検索 (Retrieval) で関連データを集め、生成 AI の回答 (Augmented Generation) に使う方式です。ベクトルの検索に、外部のプログラムは要りません。業務データとベクトルを同じデータベースに置くと、類似検索の結果を、

価格や在庫などの条件や表の結合と、1つの SQL で組み合わせて取り出せます。VECTOR_DISTANCE と通常の WHERE 条件を併用できるからです。検索はデータベース内で SQL として実行でき、データベースの外のプログラムでしか実行できないわけではありません。ベクトルの列も他の列と同じ仕組みで権限が管理され、ベクトルを含む表も通常の表と同じトランザクションの対象です。同じ表に VECTOR 列を持たせれば、更新と同時に整合した検索対象になります。

セキュリティと可用性の特徴

26ai のセキュリティには、Oracle SQL Firewall があります。データベース内に組み込まれ、アプリケーションの接続元と SQL を学習して許可リストを作ります。SQL Firewall は、データを暗号化しません。承認した SQL とその実行元の組み合わせだけを実行させ、許可リストに無い SQL は、違反として記録し、ブロックもできます。アプリケーションの SQL インジェクションへの備えになります。

機能	守るもの
Oracle SQL Firewall	実行する SQL (許可リスト方式)
Transparent Data Encryption	保存されたデータ(暗号化)
Data Redaction	問合せ結果の表示(一部を隠す)
統合監査	操作の履歴(記録)

Transparent Data Encryption は保存データを暗号化しますが、実行する SQL は制限しません。Data Redaction は問合せ結果の一部を隠す機能です。統合監査は操作を記録する機能で、許可リストに基づいて SQL を拒否しません。

可用性では、Globally Distributed Database が拠点をまたいでデータを分散して持ちます。データを複数のシャードへ分散し、Raft ベースのレプリケーションで、障害時に自動でフェイルオーバーします。Raft は多数決のコンセンサスに基づくレプリケーション方式で、リーダーに障害が起きると新しいリーダーが選ばれます。Transparent Data Encryption は暗号化、Oracle SQL Firewall は SQL の許可リスト、SQL ドメインは列定義の再利用の機能で、どれもフェイルオーバーとは関係がありません。Data Guard や RAC のような従来の可用性の機能も、26ai に引き続き含まれます。

19c の機能との対応

19c と 26ai は、基本構造が共通です。表領域、データファイル、REDO ログ、SGA と PGA の考え方は変わりません。制御ファイルと REDO ログも 26ai で必須のファイルで、メモリーだけで運用されることはありません。表領域とデータファイルも従来どおり使われ、セグメントはその表領域の中に作られます。リスナーとサービス名による接続も 26ai で使えます。実機の V\$CONTROLFILE と V\$LOGFILE には従来どおりのファイルが並び、リスナーは lsnrctl で従来どおり管理できます。

SQLPlus は 26ai でも使え、廃止されて SQL Developer だけになったわけではありません。実機の SQLPlus で、従来どおりの SQL を実行できます。専用ハードウェアも必須ではありません。26ai は汎用の x86-64 サーバーにも導入できます。

19c から 26ai へは、AutoUpgrade を使って直接アップグレードできます。19c で使っていた SQL や PL/SQL の多くは、26ai でもそのまま動きます。26ai では、そこへ AI Vector Search などの機能が追加されました。19c にはなかった VECTOR 型が、26ai では標準で使えます。

違いが出るのは、データベースの構成です。26ai は CDB だけをサポートするので、19c の非 CDB は、26ai へ移行する際に PDB へ変換して CDB に取り込みます。19c の PDB を非 CDB に戻す方向の変換は求められません。

SAMPLE

サンプルはここまでです

続き(第2章～巻末)は、ORACLE MASTER Bronze DBA 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。

SAMPLE

ORACLE MASTER Bronze DBA 26ai 学習用テキスト(無料サンプル)

版

2026年10月版(日本オラクル「Bronze DBA Oracle AI Database 26ai Fundamentals (1Z0-185-JPN)」試験内容チェックリスト準拠)

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登録商標です。ORACLE MASTER は日本オラクル株式会社の登録商標です。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 の公式文書の内容を基に作成したもので、Oracle Corporation および日本オラクル株式会社との提携・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。