

IT 資格 道 場

ORACLE MASTER Silver 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 Associate 1Z0-182-JPN」
試験内容チェックリスト(13分野・34細目) 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 Databaseアーキテクチャ [1.1/1.2/1.3]

- インスタンスとデータベースの関係 [1.1]
- マルチテナント構成:CDB・ルート・シード・PDB [1.1]
- 接続とセッション、専用サーバーと共有サーバー [1.1]
- SGA の構成要素 [1.2]
- PGA の構成と用途 [1.2]
- 主なバックグラウンド・プロセス [1.2]
- 書込みの契機:コミット、チェックポイント、高速コミット [1.2]
- CDB とプロセス・メモリーの共有 [1.2]
- 物理ファイル [1.3]
- 論理構造の階層 [1.3]
- 制御ファイルと REDO ログの多重化 [1.3]
- CDB と PDB の物理ファイル [1.3]

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 Silver DBA Oracle AI Database 26ai (試験番号 1Z0-182-JPN、試験名は Oracle AI Database 26ai Administration Associate) は、データベース管理者の仕事を、仕組みの理解と操作の選び方で問う試験です。[^m-1]データベースの構造、起動と停止、接続、ユーザーと権限、記憶域、UNDO、データの移動、PDB、監査、自動メンテナンス、性能管理が範囲で、SQL の書き方は含みません。

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

公式の試験内容チェックリストは13分野に分かれていて、このテキストも同じ順番で13章に並べました。出力を載せた例は、Oracle AI Database 26ai Free (バージョン 23.26.3) で実際に実行した結果です。SQL や管理コマンドの下にある出力は、そのときの画面そのままです。

第1章 Oracle Databaseアーキテクチャ [1.1/1.2/1.3]

この章では、Oracle Database がメモリー・プロセス・ファイルの 3 つで組み立てられていることを見ます。後の章の管理操作は、この 3 つのどこに作用するかで説明されます。例は Oracle AI Database 26ai Free (バージョン 23.26.3) で実際に実行した結果。

インスタンスとデータベースの関係 [1.1]

インスタンスは、SGA というメモリー領域とバックグラウンド・プロセスの組で、データベースは、ディスク上のファイルの組です。両者は別のもので、インスタンスはデータベースなしでも存在でき、データベースもインスタンスなしで存在できます。ただし、同時に開ける相手は決まっています。1 つのインスタンスが開けるのは 1 つのデータベースだけで、開いたデータベースを別のものに付け替えることもできません。停止後に別のインスタンスを起動して、同じファイルを開き直します。[^c1-1]

1 つのデータベースを 1 つのインスタンスで開く構成が、シングル・インスタンスで、データベースとインスタンスは 1 対 1 になります。1 つのデータベースを複数のホストの複数のインスタンスで同時に開く構成は Oracle RAC (Real Application Clusters) で、1 対多です。両者を同時に使うことはできません。RAC の管理操作は範囲外なので、この試験では構成の名前と 1 対多の関係が問われます。

Oracle Restart は、RAC を使わない単一サーバーで使う仕組みで、Grid Infrastructure の一部です。インスタンス・リスナー・ASM などが障害で止まった時やホストを再起動した時に、依存関係の順に自動で起動し直します。クラ

スタ化はしないので、複数のインスタンスで負荷を分けることはありません。
単一サーバーの自動復旧のための仕組みです。[^c1-11]

インスタンスの起動は 3 段階で進みます。NOMOUNT では、インスタンス (SGA とバックグラウンド・プロセス) だけが存在し、データベースとはまだ結び付いていません。MOUNT では、初期化パラメータ CONTROL_FILES に書かれた制御ファイルを読み、データ・ファイルと REDO ログの名前と場所を知ります。OPEN でデータ・ファイルと REDO ログが開かれ、一般の利用者が接続できます。

動的パフォーマンス・ビューは、段階によって見られるものが違います。V\$INSTANCE と V\$BGPROCESS は NOMOUNT から問い合わせられます。V\$DATAFILE のようにファイルの情報を持つビューは、制御ファイルを読む MOUNT まで進まなければ問い合わせられません。

```
SELECT banner_full FROM v$version;
```

```
BANNER_FULL
```

```
-----  
-----
```

```
Oracle AI Database 26ai Free Release 23.26.3.0.0 - Develop, Learn, and  
Run for Free  
Version 23.26.3.0.0
```

Free は、開発・学習・実行を無償で行える提供形態です。この教材の例は、すべて Free の実行結果。製品名は 26ai ですが、内部のリリース番号は 23.26.x です。23ai の名前で出た製品が 26ai に改称されたので、名前と番号が合わなくても別の製品ではありません。

マルチテナント構成:CDB・ルート・シード・PDB [1.1]

マルチテナント構成のデータベースを、マルチテナント・コンテナ・データベース (CDB) と呼びます。1 つの CDB の中には、CDB\$ROOT (ルート)、PDB\$SEED (シード)、0 個以上のプラガブル・データベース (PDB) が入っています。利用者の目に見えるデータベースは PDB です。アプリケーションからは独立したデータベースに見えますが、実体は CDB の中に置かれた、スキーマ・オブジェクトの持ち運べる集まりです。[^c1-4]

コンテナ	CON_ID	役割	読み書き
CDB\$ROOT	1	Oracle 提供のメタデータと共通ユーザーを持つ	読み書き
PDB\$SEED	2	新しい PDB を作る時の雛形	読取り専用で変更できない
PDB	3 以上	利用者のデータを入れるデータベース	開き方による

CON_ID が 0 の行は、特定のコンテナではなく、CDB 全体に関わる行を指します。CDB\$ROOT で V\$ ビューや CDB_ ビューを問い合わせると、各行の CON_ID でコンテナを見分けられます。

```
SELECT con_id, name, open_mode FROM v$containers ORDER BY con_id;
```

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

26ai がサポートするのは、マルチテナント構成のデータベースだけです。非 CDB (マルチテナントでない従来型のデータベース) は 26ai では使えません。旧版の非 CDB を 26ai に移す時は、PDB として CDB に取り込みます。[^c1-8] 試験では、26ai のデータベースは CDB 前提で動く、という事実が問われます。

SQL*Plus で今いるコンテナを知るには、SHOW CON_NAME と SHOW CON_ID を使います。コンテナを切り替えるのは、共通ユーザーが実行する ALTER SESSION SET CONTAINER で、切り替え先は CDB\$ROOT・PDB\$SEED・PDB のどれかです。接続は切れませんが、1 つのトランザクションが複数のコンテナにまたがることはできません。[^c1-9]

```
SQL> alter session set container=FREEPDB1;
```

```
SQL> show con_name
```

```
CON_NAME
```

```
-----
```

```
FREEPDB1
```

```
SQL> show con_id
```

```
CON_ID
```

```
-----
```

```
3
```

アプリケーション・コンテナは任意の構成です。アプリケーション・ルートと、その配下のアプリケーション PDB からなり、複数の PDB で共通のアプリケーション定義(表やデータ)を共有する時に使います。アプリケーション・ルートに置いた定義を、配下の PDB がまとめて引き継ぎます。PDB を並べるだけの CDB と違い、アプリケーションの共通部分を 1 か所で保守する入れ物です。

CDB を動かすインスタンスは 1 つで、PDB ごとのインスタンスはありません。PDB は個別に開いたり閉じたりできますが、メモリーとバックグラウンド・プロセスは CDB 全体で 1 組です。ルートを先に開かないと、PDB は開けません。CDB を開いた直後は、ルートが開き、PDB はマウント状態にあります。シードの PDB\$SEED は読取り専用で開いている。

接続とセッション、専用サーバーと共有サーバー [1.1]

接続とセッションは別のものです。接続は、クライアントのプロセスとインスタンスの間の物理的な通信路で、セッションは、インスタンスのメモリーの中にある、ログインした利用者の状態を表す論理的な単位です。1 つの接続の上に、セッションを 0 個、1 個、または複数持てます。セッションどうしは独立しているので、あるセッションの COMMIT が、別のセッションのトランザクションに影響することはありません。DISCONNECT が終わらせるのはセッションで、接続はそのまま残ります。[^c1-2]

接続を作る側のクライアントのプロセスが、ユーザー・プロセスです。SQL*Plus やアプリケーションがこれにあたり、インスタンスの外で動きます。ユーザー・プロセスの SQL を受けて処理するのが、インスタンス側のサーバー・プロセスで、構成は 2 通りに分かれる。専用サーバーは、クライアント 1 つにサーバー・プロセスを 1 つ付ける構成で、クライアントが 20 なら、サーバー・プロセスも 20 になります。既定はこちらで、Free のセッションも、SERVER 列

はすべて DEDICATED です。サーバー・プロセスを複数のクライアントで共有することはありません。

```
SELECT username, server, type FROM v$session WHERE username IS NOT NULL;
```

USERNAME	SERVER	TYPE
SYS	DEDICATED	USER
SYS	DEDICATED	BACKGROUND

共有サーバーでは、クライアントはサーバー・プロセスに直接つながらず、ディスパッチャに接続します。処理の流れは次のとおり。

1. ディスパッチャが要求を、SGA のラージ・プールにあるリクエスト・キューに置く
2. 待機中の共有サーバー・プロセスの最初の 1 つが、要求を取り出して処理する
3. 結果をディスパッチャの応答キューに置く
4. ディスパッチャが結果をクライアントに返す

クライアント 20 台がディスパッチャ 1 つにつながり、少数の共有サーバー・プロセスで処理されるので、接続数が多い環境でプロセス数を抑えられます。

2 つの構成は、メモリーの置き場所にも違いが出ます。セッション情報を持つ UGA は、専用サーバーならサーバー・プロセスの PGA にあり、SGA は使いません。共有サーバーでは、どの共有サーバー・プロセスもセッションのデータを読めなければならないので、UGA は SGA に置かれます。置き場所は、ラージ・

プールがあればそこで、なければ共有プールです。共有サーバーでは、ラージ・プールの使用量が増える点に注意する。専用サーバーでは、この UGA はサーバー・プロセスごとの PGA にあるので、SGA は増えません。

共有サーバーは、初期化パラメータ SHARED_SERVERS と DISPATCHERS で設定します。Free は SHARED_SERVERS が 0 なので、共有サーバー・プロセスは 1 つも起動していません。SHARED_SERVERS を 0 にすると共有サーバーは無効になり、DISPATCHERS に値が入っていても、新しいクライアントは共有モードで接続できません。

場面で確かめます。同時接続が数千あり、各接続はほとんどの時間アイドルで、たまに短い SQL を流すだけのアプリケーションがあるとします。専用サーバーでは接続数と同じ数のサーバー・プロセスが居座るので、プロセス数とメモリーが膨らみます。共有サーバーなら、少数の共有サーバー・プロセスで足りるため、プロセスが膨らみません。反対に、長時間かかるバッチ処理は共有サーバー・プロセスを長く占有するので、共有サーバーには向きません。バッチ処理は専用サーバーで接続します。

SGA の構成要素 [1.2]

SGA (System Global Area) は、インスタンスの共有メモリーで、同じインスタンスのサーバー・プロセスとバックグラウンド・プロセスのすべてが共有します。インスタンスの起動時に割り当てられ、停止時に解放される。[^c1-3]

要素	置くもの	備考
データベース・バッファ・キャッシュ	データ・ファイルから読んだブロックの写し	変更されたバッファ(ダーティ・バッファ)は DBWn が書く

要素	置くもの	備考
共有プール	解析済みの SQL・PL/SQL、データ・ディクショナリの情報	ライブラリ・キャッシュ、データ・ディクショナリ・キャッシュ、サーバー結果キャッシュなどからなる
REDO ログ・バッファ	変更の記録 (REDO エントリ)	循環バッファ。LGWR がオンライン REDO ログへ書く。大きさは LOG_BUFFER
ラージ・プール	共有サーバーの UGA、RMAN の入出力バッファ、パラレル実行のメッセージ用など	任意。大きな割当て用
Java プール	Java のコードとデータ (JVM のセッション固有の部分)	
Streams プール	バッファ付きキューのメッセージ用。Oracle GoldenGate、XStream、Advanced Queuing、Data Pump が使う	大きさは STREAMS_POOL_SIZE。既定は 0
固定 SGA	内部の管理情報	大きさは Oracle が決め、利用者は変えられない
インメモリー列ストア	表などを列形式にした写し	任意。In-Memory オプションの機能

共有プールのライブラリ・キャッシュは、同じ SQL 文の再解析を省きます。サーバー結果キャッシュには、SQL の結果セットと PL/SQL ファンクションの結果が入ります。ラージ・プールには LRU のリストがありません。[^c1-10]

Free で SGA の内訳を見ます。Streams プールとインメモリー領域は 0 です。Streams プールは自動共有メモリー管理が必要に応じて大きさを決め、インメモリー領域は INMEMORY_SIZE (Free は 0) を設定しない限り確保されません。

```
SELECT name, bytes/1024/1024 mb FROM v$sgainfo
WHERE name IN ('Fixed SGA Size','Redo Buffers','Buffer Cache
Size','Shared Pool Size',
               'Large Pool Size','Java Pool Size','Streams Pool
Size','In-Memory Area Size')
ORDER BY name;
```

NAME	MB
-----	-----
Buffer Cache Size	944
Fixed SGA Size	4.79359436
In-Memory Area Size	0
Java Pool Size	128
Large Pool Size	16
Redo Buffers	4.3203125
Shared Pool Size	416
Streams Pool Size	0

メモリーの管理方式は 3 つあります。

- 自動メモリー管理: SGA と PGA の合計を MEMORY_TARGET で決め、Oracle が配分を自動調整する

- 自動共有メモリー管理: SGA の合計を SGA_TARGET で決め、SGA の各要素の配分を自動調整する
- 手動管理: 各領域の大きさを個別に指定する

Free は SGA_TARGET が 1610612736 バイト (約 1.5GB) で、MEMORY_TARGET は 0 です。自動共有メモリー管理にあたります。

PGA の構成と用途 [1.2]

PGA (Program Global Area) は、サーバー・プロセスごとに割り当てられる、非共有のメモリーです。そのプロセスだけが使い、ほかのプロセスとは共有しません。SGA がインスタンスに 1 つなのに対し、PGA はプロセスの数だけあります。

PGA に入るのは、SQL 作業領域とセッション情報などです。SQL 作業領域は、ソート、ハッシュ結合、ビットマップのマージなど、メモリーを多く使う処理の作業場所です。プライベート SQL 領域には、バインド変数の値や実行の状態が入ります。SGA にある共有 SQL 領域 (解析済みの SQL 文) は別の領域で、ここには含まれません。同じ SQL 文でも、バインド変数の値はセッションごとに PGA へ持つので、セッション間で共有はされません。

セッション情報を持つ UGA (User Global Area) の置き場所は、専用サーバーなら PGA、共有サーバーなら SGA と決まっています。PGA 全体の目標は PGA_AGGREGATE_TARGET で決め、Free の値は 536870912 バイト (512MB) です。

SGA と PGA の違いを、場面で確かめます。専用サーバーで接続した 100 セッションが、それぞれ大きなソートを実行しているとします。ソートの作業領域は、各セッションのサーバー・プロセスの PGA に取られるので、SGA の大きさ

は変わりません。足りなくなるのは PGA の合計の側です。作業領域に収まらないソートは、一時表領域の一時セグメントへ書き出されます。

主なバックグラウンド・プロセス [1.2]

インスタンスの起動とともに、バックグラウンド・プロセスが起動します。主なものの役割を次に示します。[^c1-2]

プロセス	役割
DBWn	ダーティ・バッファをデータ・ファイルへ書く。 コミットでは書かない
LGWR	REDO ログ・バッファをオンライン REDO ログへ書く
CKPT	チェックポイントで制御ファイルとデータ・ファイル・ヘッダーを更新し、DBWn に書込みを指示する。自分ではブロックを書かない
SMON	インスタンス・リカバリ、使われなくなった一時セグメントの掃除
PMON	異常終了したサーバー・プロセスやディスパッチャの後始末 (PMON グループの CLMN が補助する)
LREG	リスナーへのインスタンスとディスパッチャの登録
MMON	AWR のスナップショット取得、メトリックの計算、しきい値を超えたアラート
MMNL	ASH バッファのデータをディスクへ書く (バッファが満杯になった時)

プロセス	役割
ARCN	ARCHIVELOG モードで、ログ・スイッチ後のオンライン REDO ログをアーカイブ REDO ログへ写す

```
SELECT name, description FROM v$bgprocess
WHERE name IN
('DBW0','LGWR','CKPT','SMON','PMON','LREG','MMON','MMNL','ARC0') ORDER
BY name;
```

NAME	DESCRIPTION
-----	-----
ARC0	Archival Process 0
CKPT	checkpoint
DBW0	db writer process 0
LGWR	Redo etc.
LREG	Listener Registration
MMNL	Manageability Monitor Process 2
MMON	Manageability Monitor Process
PMON	process cleanup
SMON	System Monitor Process

まぎらわしいのは PMON と LREG です。リスナーへの登録は、旧版では PMON が行う仕事でした。12c 以降は LREG が受け持ちます。インスタンスの起動時に LREG がリスナーの有無を確かめ、動いていなければ定期的に接続

を試みます。リスナーを後から起動しても、LREG の再試行で、サービスが登録されます。PMON が登録を行うという選択肢は、古い知識の誤りです。

ARCn は、ARCHIVELOG モードで自動アーカイブが有効な時にしか存在しません。NOARCHIVELOG の Free で V\$PROCESS を見ると、表のプロセスのうち動いているのは DBW0・LGWR・CKPT・SMON・PMON (と補助の CLMN) ・LREG・MMON・MMNL で、ARC0 はありません。このほかにも GEN0・DBRM・VKTM など多数のバックグラウンド・プロセスが動いています。

プロセスが異常終了した時の動きも問われます。サーバー・プロセスが異常終了すると、PMON グループがプロセス・リカバリを行い、そのセッションのロックなどを後始末します。インスタンスが異常終了した場合は、次の起動の OPEN の時に、SMON がインスタンス・リカバリを自動で行います。利用者の操作は要りません。

書込みの契機：コミット、チェックポイント、高速コミット [1.2]

メモリー上の変更がいつディスクに出るかは、LGWR と DBWn で契機が違います。この区別が、この節でいちばん問われます。

LGWR が REDO ログ・バッファを書く契機は 5 つあります。

- トランザクションがコミットした時
- オンライン REDO ログでログ・スイッチが起きた時
- 前回の書込みから 3 秒たった時
- REDO ログ・バッファの 3 分の 1 が埋まった時 (または 1MB の REDO がたまった時)
- DBWn がバッファを書く前に、未書込みの REDO を見つけた時

最後の契機は、ログ先行書込みの決まりによります。DBWn がダーティ・バッファを書く前に、そのバッファの変更を表す REDO が、先にディスクへ出ていなければなりません。

DBWn が書く契機は、コミットではありません。きれいで再利用できるバッファが見つからず、サーバー・プロセスが DBWn に合図した時があります。チェックポイントを進めるために DBWn が定期的を書く時や、表領域が読取り専用やオフラインになる時もあります。

COMMIT では、LGWR が COMMIT レコードを REDO ログ・バッファに置いて、すぐにディスクへ書きます。この書込みが終わった時点で、トランザクションはコミット済みになります。変更されたデータ・ブロックの書込みは待ちません。REDO がディスクにあれば完了とし、データ・ブロックはあとで DBWn が書く。この仕組みが高速コミットです。COMMIT の時点で、データ・ファイルへの書込みは保証されていません。

高速コミットの意味を、障害の場面で考えます。COMMIT が成功した直後にインスタンスが異常終了し、データ・ブロックの変更が、まだデータ・ファイルに書かれていなかったとします。それでもコミットは失われません。再起動して OPEN する時に、SMON がオンライン REDO ログを読んで、データ・ファイルに反映されていない変更をやり直します（ロールフォワード）。続いて、コミットされていなかった変更を UNDO で取り消します（ロールバック）。この 2 段階がインスタンス・リカバリです。必要なのはオンライン REDO ログと現在のデータ・ファイルだけで、バックアップは要りません。

CKPT は、チェックポイントの位置を記録するプロセスです。チェックポイント位置は、バッファ・キャッシュにある最も古いダーティ・バッファで決まり、インスタンス・リカバリはそこから始まります。CKPT は制御ファイルとデータ・ファイル・ヘッダーにこの位置を書き、DBWn に書込みを指示します。データ・ブロッ

クも REDO も、CKPT 自身は書きません。「CKPT がダーティ・バッファをデータ・ファイルへ書く」という選択肢は、典型的な誤りです。

```
ALTER SYSTEM CHECKPOINT;  
ALTER SYSTEM SWITCH LOGFILE;
```

ALTER SYSTEM CHECKPOINT は手動でチェックポイントを起こします。ALTER SYSTEM SWITCH LOGFILE は手動でログ・スイッチを起こし、ログ・スイッチでもチェックポイントが起きます。

CDB とプロセス・メモリーの共有 [1.2]

1 つの CDB のすべての PDB は、同じ SGA と同じバックグラウンド・プロセスを共有します。PDB ごとに LGWR や DBWn が起動することはありません。PDB ごとには持たない物は、インスタンス (SGA・PGA の枠組みとバックグラウンド・プロセス)、制御ファイル、オンライン REDO ログです。

場面で確かめます。PDB1 と PDB2 のある CDB で、PDB1 の利用者が大量の更新をコミットしたとします。この REDO を書くのは、CDB に 1 つだけの LGWR です。PDB2 の利用者のコミットも、同じ REDO ログ・バッファと同じオンライン REDO ログに入り、別々には書き分けられません。PDB ごとの REDO ログは存在しません。

物理ファイル [1.3]

データベースは、次のファイルでできています。[^c1-5]

ファイル	内容	要点
制御ファイル	データベース名、データ・ファイルと REDO ログの名前と場所、ログ順序番号、チェックポイント情報など	CONTROL_FILES で場所を指定。複数の多重化する
データ・ファイル	表領域のデータの入れ物	表領域は 1 つ以上のファイルで構成される
一時ファイル	一時表領域のデータ	REDO を生まず、リカバリの対象でもない
オンライン REDO ログ・ファイル	変更の記録	2 グループ以上を循環で使う
アーカイブ REDO ログ・ファイル	満杯になったオンライン REDO ログの写し	ARCHIVELOG モードの時だけ作られる
パラメータ・ファイル	初期化パラメータ	SPFILE (バイナリ) と PFILE (テキスト)
パスワード・ファイル	SYSDBA などの管理権限を持つユーザーのパスワード	データベースが開いていなくても認証に使える

制御ファイルは、起動の MOUNT の段階で読まれます。データベースを開いて復旧するのに必要な情報(チェックポイント情報を含む)を持つので、別々のディスクに多重化して持ちます。全部のコピーを失うと、メディア・リカバリが必要になります。

一時ファイルは、ソートやハッシュ処理であふれた作業データを入れる、一時表領域専用のファイルです。永続的なオブジェクトは入りません。REDO を生成せず (NOLOGGING)、読取り専用にもできず、メディア・リカバリの対象にもなりません。一時ファイルは、V\$DATAFILE と DBA_DATA_FILES には出ず、V\$TEMPFILE と DBA_TEMP_FILES に出ます。

アーカイブ REDO ログは、データベースの構成要素ではありません。ARCHIVELOG モードの時に、満杯になったオンライン REDO ログのメンバーを、利用者の指定した場所へ複製したものです。バックアップからのメディア・リカバリ、スタンバイの更新、LogMiner による履歴の分析に使います。ARCHIVELOG モードは CDB 全体で 1 つの設定で、PDB ごとには切り替えられません。

パラメータ・ファイルには、サーバー・パラメータ・ファイル (SPFILE) とテキストの PFILE があります。SPFILE はバイナリで、Oracle だけが読み書きします。実行中に ALTER SYSTEM で変えた値を、再起動後も残せるので、推奨されます。PFILE は key=value 形式のテキストで、Oracle は読めますが書けません。Free の SPFILE は ORACLE_HOME の dbs ディレクトリにある spfileFREE.ora です。

パスワード・ファイルは、データベースが開いていない時でも管理者を認証するために、データベースの外に置かれます。データ・ディクショナリはデータ・ファイルの中にあり、開く前には読めないからです。

論理構造の階層 [1.3]

論理構造は、小さい順に、データ・ブロック、エクステント、セグメント、表領域です。[[^]c1-6]

単位	説明
データ・ブロック	論理的な記憶の最小単位で、入出力の最小単位。Oracle ブロックとも呼ぶ
エクステント	論理的に連続するデータ・ブロックの集まり。1 つのデータ・ファイルの中に収まる

単位	説明
セグメント	1つのオブジェクト(表や索引)に割り当てたエクステントの集まり。1つの表領域にだけ属する
表領域	セグメントを入れる論理的な入れ物。実体は1つ以上のデータ・ファイルまたは一時ファイル

包含関係は「表領域 ⊃ セグメント ⊃ エクステント ⊃ データ・ブロック」です。「1つのセグメントが複数の表領域にまたがる」は誤りで、セグメントは1つの表領域に属します。一方、1つの表領域は複数のデータ・ファイルを持てます。同じセグメントのエクステントが、同じ表領域の別々のデータ・ファイルに置かれることはあります。ただし、1つのエクステントが複数のデータ・ファイルにまたがることはありません。

データ・ブロックの大きさを決めるのは、初期化パラメータ DB_BLOCK_SIZE です。指定しなかった時の既定は文書によって書き方が違い(参照ガイドは 8192、概要ガイドは OS によって決まり標準は 4KB か 8KB)、Free の値は 8192 バイト (8KB) です。データベースの作成時にだけ設定でき、作成後は変更できません。[^c1-7]

```
SELECT name, value FROM v$parameter WHERE name = 'db_block_size';
```

NAME	VALUE
-----	-----
db_block_size	8192

この値は SYSTEM 表領域と SYSAUX 表領域に適用され、ほかの表領域の既定のブロック・サイズにもなります。表領域ごとに別のブロック・サイズ (非標準ブロック・サイズ) を指定する方法もあります。

ブロックの内部は、ブロック・ヘッダー (トランザクション情報など)、表ディレクトリ、行ディレクトリ、空き領域、行データでできています。行データは、ブロックの下から上へ詰められます。PCTFREE は、将来の UPDATE のために空けておく割合の設定。値が小さいと、更新で行が長くなった時にブロックへ収まらず、行が別のブロックへ移る行移行が増えます。LOB 型の列には PCTFREE は適用されません。

表領域の管理方式には、ローカル管理 (既定) とディクショナリ管理があります。ローカル管理は、データ・ファイル・ヘッダーのビットマップで空き領域を管理するので、データ・ディクショナリへの再帰的な操作が起きません。ディクショナリ管理は、データ・ディクショナリの表でエクステントを管理するため、性能に響くことがあります。ローカル管理のエクステントの大きさは、一律 (既定 1MB) か自動決定で、一時表領域は一律だけです。

セグメント内の空き領域の管理方式は、ASSM (自動セグメント領域管理) が既定です。ASSM は永続のローカル管理表領域で使い、PCTFREE だけを指定します。旧来の手動管理 (MSSM) は PCTUSED や FREELISTS も指定する方式で、推奨されません。

Free の表領域は、すべてローカル管理です。永続表領域のうち SYSAUX と USERS が、ASSM (AUTO) になっています。

```
SELECT tablespace_name, block_size, contents, extent_management,  
       segment_space_management  
FROM dba_tablespaces ORDER BY 1;
```

TABLESPACE_NAME	BLOCK_SIZE	CONTENTS	EXTENT_MAN	SEGMENT
-----	-----	-----	-----	-----
SYSAUX	8192	PERMANENT	LOCAL	AUTO
SYSTEM	8192	PERMANENT	LOCAL	MANUAL
TEMP	8192	TEMPORARY	LOCAL	MANUAL
UNDOTBS1	8192	UNDO	LOCAL	MANUAL
USERS	8192	PERMANENT	LOCAL	AUTO

セグメントには種類があります。データ・セグメントは表や表パーティションごと、索引セグメントは索引ごとに作られ、既定では最初の行を入れる時に作られます(遅延セグメント作成)。実際に確かめます。表を作った時点ではセグメントは0個で、10行をINSERTすると、USERS表領域に1つできる。

SEGMENT_NAME	SEGMENT_TYPE	TABLESPACE	EXTENTS	BLOCKS	BYTES
-----	-----	-----	-----	-----	-----
T	TABLE	USERS	1	8	65536
EXTENT_ID	FILE_ID	BLOCKS	BYTES		
-----	-----	-----	-----		
0	24	8	65536		

セグメントはエクステント1つ、そのエクステントはブロック8つ(8KB × 8 = 64KB)で、10行はすべて同じブロックに入っています。一時セグメントは、ソートなどの作業場所で、処理が終わると消えます。UNDOセグメントは、ロールバックと読取り一貫性のためのUNDOデータを持ちます。UNDOデータが表や索引のセグメントに入ることはありません。

制御ファイルと REDO ログの多重化 [1.3]

オンライン REDO ログは、REDO ログ・グループという単位で管理します。グループは、同じ内容を持つ複数のメンバー（ファイル）からなります。メンバーが 1 つのグループも許されますが、グループが 1 つだけの構成は認められません。最低 2 つ必要なのは、1 つが書き込み中の間に、もう 1 つをアーカイブまたは消去できるようにするためです。[^c1-5]

LGWR は、現在のグループのすべてのメンバーに、同時に同じ内容を書きます。グループが満杯になると次のグループへ移ります（ログ・スイッチ）。最後のグループの次は最初のグループに戻り、円を描いて使い回します。同じグループのメンバーは同じ大きさでなければなりません。LGWR が別々のグループのメンバーへ同時に書くことはありません。メンバーを別々のディスクに置いて多重化すると、ディスク障害で REDO を失う事態を避けられます。

Free は、グループが 2 つで、各グループのメンバーが 1 つ、各 10MB です。

```
SELECT group#, members, bytes/1024/1024 mb, status FROM v$log;
```

GROUP#	MEMBERS	MB	STATUS
-----	-----	-----	-----
1	1	10	CURRENT
2	1	10	ACTIVE

場面で確かめます。グループが 3 つで、各グループにメンバーが 2 つ（合計 6 ファイル）の構成で、グループ 2 のメンバーの 1 つのディスクが壊れたとします。同じグループのもう 1 つのメンバーが同じ内容を持つので、データベースは止まりません。壊れたメンバーを作り直せば、元の状態に戻ります。ちなみに

に「グループ 1 つ、メンバー 2 つ」の構成は、グループが最低 2 つという条件を満たしません。

グループの再利用の条件は、アーカイブ・モードで変わります。NOARCHIVELOG では、満杯のグループは、その変更がデータ・ファイルに書かれたあとで再利用できます。ARCHIVELOG では、それに加えてアーカイブ済みでなければなりません。インスタンス・リカバリにまだ必要なグループ (ACTIVE) は、上書きされません。

制御ファイルも多重化します。同じ内容の複数のコピーを同時に開いて書き、場所は CONTROL_FILES に書きます。コピーの 1 つが使えなくなっても、ほかのコピーが最新なら、メディア・リカバリなしで再び開けます。Free は制御ファイルを 2 つ (control01.ctl と control02.ctl) 持ちます。

制御ファイルは、同じ内容のコピーを複数持つ 1 段の構造で、グループという単位はありません。REDO ログは、グループが 2 つ以上あり、各グループがメンバーを複数持つ 2 段の構造になっています。

CDB と PDB の物理ファイル [1.3]

ファイルは、CDB 全体で共有する物と、PDB ごとに持つ物に分かれます。[^c1-4]

区分	ファイル	備考
CDB で共有	制御ファイル	PDB ごとの制御ファイルはない
CDB で共有	オンライン REDO ログ	PDB ごとの REDO ログはない
CDB で共有	UNDO (共有 UNDO の時)	CDB で 1 つの UNDO 表領域

区分	ファイル	備考
PDB ごと	SYSTEM と SYSAUX のデータ・ファイル	ルート・シード・各 PDB がそれぞれ持つ
PDB ごと	ユーザーの表領域のデータ・ファイル	
PDB ごと	UNDO 表領域 (ローカル UNDO の時)	
PDB ごと	一時表領域 (任意)	

Free はローカル UNDO が有効 (LOCAL_UNDO_ENABLED が TRUE) で、ルート・シード・PDB のそれぞれに UNDOTBS1 があります。データ・ファイルの数は次のとおりで、制御ファイルは CDB 全体で 2 つ (多重化したコピー) だけです。

```
SELECT con_id, count(*) files FROM v$datafile GROUP BY con_id ORDER BY 1;
SELECT count(*) controlfiles FROM v$controlfile;
```

CON_ID	FILES
1	4
2	3
3	4

CONTROLFILES
2

FREEPDB1 (CON_ID 3) の 4 つ は system01.dbf・sysaux01.dbf・undotbs01.dbf・users01.dbf です。temp01.dbf は一時ファイルなので、V\$DATAFILE には出ません。

場面で確かめます。CDB に PDB が 3 つある時のファイルの数を数える問題では、制御ファイルとオンライン REDO ログは CDB に 1 組だけで、PDB を足しても増えません。増えるのは、各 PDB の SYSTEM・SYSAUX・ユーザーのデータ・ファイルで、ローカル UNDO の時は UNDO のデータ・ファイルも増えます。

ARCHIVELOG モードは、REDO ログが CDB で共有なので、CDB 全体で 1 つです。PDB1 だけを ARCHIVELOG にする操作は存在しません。

サンプルはここまでです

続き(第2章～巻末)は、ORACLE MASTER Silver 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。

SAMPLE

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

版

2026年10月版(Oracle Japan「Oracle AI Database 26ai Administration Associate 1Z0-182-JPN」試験内容チェックリスト(13分野・34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。