# 知られざる Oracle Database 12*c* の新機能

## 有償オプションを購入しなくても使える便利な新機能

篠田典良 /日本ヒューレット・パッカード株式会社 / 2014年9月7日

# 自己紹介

篠田 典良(しのだ のりよし)

- 所属
  - 日本ヒューレット・パッカード株式会社 コンサルティング事業統括
    noriyoshi.shinoda@hp.com
- 経歴
  - 1990年 日本ディジタルイクイップメント株式会社
  - 2000年 日本ヒューレット・パッカード株式会社
- 現在の業務
  - Oracle Database, Microsoft SQL Server, PostgreSQL, Vertica等 RDBMS全般に関するシステムのデリバリー、コンサルティング等
  - オープンソース製品に関する調査、検証
  - Oracle Database関連書籍の執筆

# Agenda

## 知られざるOracle Database 12*c*の新機能

- 列データ型の制限拡張
- IDENTITY列
- FETCH N ROWS ONLY
- In-Database Archiving
- SEQUENCEの拡張
- 非表示列
- オンライン操作
- マルチスレッド・インスタンス
- PGA制限
- Unified Auditing
- Patch Set 12.1.0.2 情報

# 列データ型の制限拡張

# 列データ型の制限拡張

## データ型の最大バイト数の上限が変更

列データ型と PL/SQL データ型の制限（単位: バイト）

| データ型 | Oracle Database 11*g* 列最大長 | Oracle Database 11*g* PL/SQL最大長 | Oracle Database 12*c* 列最大長 |
|---|---|---|---|
| CHAR | 2,000 | 32,767 | 2,000 |
| NCHAR | 2,000 | 32,767 | 2,000 |
| VARCHAR2 | 4,000 | 32,767 | 32,767 |
| NVARCHAR2 | 4,000 | 32,767 | 32,767 |
| RAW | 2,000 | 32,767 | 32,767 |

# 文字列型の最大サイズ拡張

変更手順

**注意！一度変更すると元に戻せません**

標準では無効になっているため、変更が必要。

以下の手順で設定を変更する。

```
$ sqlplus / AS SYSDBA
SQL> STARTUP UPGRADE
SQL> ALTER SYSTEM SET max_string_size = EXTENDED ;
SQL> @?/rdbms/admin/utl32k.sql
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

# 文字列型の最大サイズ拡張

実行例

```
SQL> CREATE TABLE large1 (col1 VARCHAR2(8192), col2 RAW(32767)) ;
表が作成されました。

SQL> DESCRIBE large1
名前                            NULL?    型
------------------------------ -------- ----------------------------
COL1                                     VARCHAR2(8192)
COL2                                     RAW(32767)
```

# 文字列型の最大サイズ拡張

実体は？

制限が拡張された列の実体は、インライン格納方式のBLOB型（拡張データ型と呼ばれる）。物理フォーマットは格納表領域のタイプと初期化パラメータdb_securefileに依存する。

```
SQL> CREATE TABLE large2(col1 VARCHAR2(32767)) ;
表が作成されました。

SQL> SELECT OBJECT_NAME, OBJECT_TYPE, GENERATED FROM USER_OBJECTS ;
OBJECT_NAME                    OBJECT_TYPE     GENERATED
------------------------------ --------------- ---------
SYS_IL0000080866C00001$$       INDEX           Y
SYS_LOB0000080866C00001$$      LOB             Y
LARGE2                         TABLE           N
```

＊インライン格納方式 = 4,000バイトまでのデータはテーブルと同一セグメントに格納する。4,000バイトを超えるデータはLOBセグメントに格納される。

# 文字列型の最大サイズ拡張

## インデックスを作成する

インデックスは作成できるが以下の制限あり

・列サイズが「ブロック・サイズ × 75% − オーバーヘッドまで」の制限は変更なし

・内部的にはファンクション索引が作成されるようだが、ALL_IND_EXPRESSIONSビューには格納されない。

```
| Id  | Operation                           | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |         |     1 |    21 |     2 (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID BATCHED| LARGE1  |     1 |    21 |     2 (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | IX1_L1  |     1 |       |     1 (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(INTERNAL_FUNCTION("LARGE1"."COL1"))
   2 - access("COL1"='Data1')
       filter(INTERNAL_FUNCTION("COL1"))
```

# 文字列型の最大サイズ拡張

LOBを使わない方法は無いのか？

```
SQL> CREATE TABLE large2 (col1 VARCHAR2(1)) ;
表が作成されました。

SQL> ALTER TABLE large2 MODIFY (col1 VARCHAR2(32767));
表が変更されました。

SQL> SELECT OBJECT_NAME, OBJECT_TYPE FROM USER_OBJECTS ;
OBJECT_NAME                    OBJECT_TYPE
------------------------------ ------------------------------
LARGE2                         TABLE
```

または初期化パラメータ _scalar_type_lob_storage_threshold を変更する。

# IDENTITY列



# IDENTITY列

## テーブル列に一意な値を自動設定する機能

- SQL Server (IDENTITY属性), PostgreSQL (serialデータ型), MySQL (AUTO_INCREMENT属性) 等, 他のRDBMSでは一般的な機能。Oracle Database 12*c*でやっと採用。
- 自動採番された値をユーザーが更新(UPDATE / INSERT)できるかどうかを選択することができる。
  - SQL Serverは IDENTITY指定列のUPDATE不可(INSERTは属性により可)
  - PostgreSQLはserial列のUPDATE/INSERT可
- CREATE TABLE文の列指定にSEQUENCEオブジェクトと同じ属性を指定可能。
- 対象列にはNOT NULL制約が自動的に付与される。
- NULL値を格納しようとした場合に、エラーになるか採番された値を格納するかを指定可能。

# IDENTITY列

## 使用例#1 更新不可能列として作成

```
SQL> CREATE TABLE idtbl1 (id NUMBER GENERATED ALWAYS AS IDENTITY,
                  val VARCHAR2(10)) ;

SQL> INSERT INTO idtbl1 VALUES ('Value1') ;
ORA-00947: 値の個数が不足しています。

SQL> INSERT INTO idtbl1 VALUES (DEFAULT, 'Value1') ;
1行が作成されました。

SQL> UPDATE idtable1 SET id = 2 ;
ORA-32796: GENERATED ALWAYSで作成されたアイデンティティ列は更新できません
```

# IDENTITY列

使用例#2 更新可能列として作成（開始番号を 10 に指定）

```
SQL> CREATE TABLE idtbl2 (id NUMBER
        GENERATED BY DEFAULT AS IDENTITY (START WITH 10), val VARCHAR2(10)) ;
表が作成されました。

SQL> INSERT INTO idtbl2 (val) VALUES ('Value2') ;
1行が作成されました。

SQL> UPDATE idtbl2 SET  id = 200 ;
1行が更新されました。
```

# IDENTITY列

実体は自動生成されるSEQUENCEオブジェクトとDEFAULTの組み合わせ

- 利用するためにはCREATE SEQUENCEシステム権限が必要
- 自動生成されたSEQUENCEオブジェクトの属性変更はALTER TABLE文で実行
- INSERT文がエラーになっても自動生成される番号は更新される。

```
SQL> CREATE TABLE idtable1 (col1 NUMBER GENERATED ALWAYS AS IDENTITY) ;
表が作成されました。

SQL> SELECT OBJECT_NAME, OBJECT_TYPE, GENERATED FROM USER_OBJECTS ;
OBJECT_NAME                    OBJECT_TYPE                    GENERATED
------------------------------ ------------------------------ ----------
IDTABLE1                       TABLE                          N
ISEQ$$_92462                   SEQUENCE                       Y
```

# IDENTITY列

## 情報の参照と操作

- ALL_TAB_IDENTITY_COLSビュー追加。ALL_TAB_COLUMNSビュー更新。
- 自動生成されたシーケンス・オブジェクトは削除できないが、NEXTVAL操作は実行可能

```
SQL> SELECT COLUMN_NAME, GENERATION_TYPE, IDENTITY_OPTIONS FROM
          USER_TAB_IDENTITY_COLS WHERE TABLE_NAME='IDTABLE1' ;
COLUMN_NAME   GENERATION_   IDENTITY_OPTIONS
------------- ------------- ----------------------------------------
COL1          ALWAYS        START WITH: 1, INCREMENT BY: 1, MAX_VALUE:
                            9999999999999999999999999999, MIN_VALUE: 1, …
SQL> SELECT ISEQ$$_92462.NEXTVAL FROM DUAL ;
NEXTVAL
----------
         2
```

# IDENTITY列

## DEFAULTとSEQUENCEの組み合わせ

- 列値のDEFAULT指定にシーケンスを指定できるようになった。
- 依存関係はチェックされていないので、シーケンスを削除することができる。

```
SQL> CREATE SEQUENCE seqdef1 ;
順序が作成されました。
SQL> CREATE TABLE tabdef1 (col1 NUMBER DEFAULT seqdef1.NEXTVAL,  col2 CHAR(8)) ;
表が作成されました。

SQL>  DROP SEQUENCE seqdef1 ;
順序が削除されました。
SQL> INSERT INTO tabdef1 (col2) VALUES ('data') ;
ORA-02289: 順序が存在しません。
```

# FETCH N ROWS ONLY



# FETCH N ROWS ONLY

順序付けられた途中のレコードを抜き出し

- 「先頭から10レコード」、「6番目から5レコード」の検索
  - Oracle Database 11*g*まではROWNUM疑似列またはROW_NUMBER関数を使う必要があった。

Oracle Database 11*g*までの記述例

```
SELECT first_name, last_name, salary FROM (
    SELECT first_name, last_name, salary,
     ROW_NUMBER() OVER (ORDER BY salary DESC)
           ranking FROM employees)
    WHERE ranking BETWEEN 6 AND 11
```

# FETCH N ROWS ONLY

## Oracle Database 12*c*の新構文

Oracle Database 12*c*の記述例

```
SELECT first_name, last_name, salary FROM employees ORDER BY salary DESC
   OFFSET 5 ROWS FETCH NEXT 6 ROWS ONLY
```

- OFFSET句を省略すると、先頭の指定数レコードを抽出可能。
- FETCH句を省略すると、OFFSET指定した先頭レコード以外の全レコードを出力。
- FIRSTとNEXTは同じ意味、ROWSとROWも同じ意味。
- レコード数以外に、PERCENT指定、同値の値を出力する WITH TIES指定も可能。
- 記述例
  - FETCH FIRST 5 ROWS ONLY
  - OFFSET 5 ROWS FETCH NEXT 5 PERCENT ROWS ONLY
  - OFFSET 5 ROWS FETCH NEXT 5 ROWS WITH TIES

# FETCH N ROWS ONLY

## 実体はROW_NUMBER関数への書き換え

実行計画を確認

```
SQL> EXPLAIN PLAN FOR SELECT * FROM employees ORDER BY salary DESC
        OFFSET 5 ROWS FETCH NEXT 6 ROWS ONLY ;
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY) ;
```

・内部的には、サブクエリー、ROW_NUMBER関数とCASE句を実行している。

```
1 - filter("from$_subquery$_002"."rowlimit_$$_rownumber"<=CASE  WHEN
  5>=0) THEN 5 ELSE 0 END +6 AND rom$_subquery$_002"."rowlimit_$$_rownumber"> 5)
2 - filter(ROW_NUMBER() OVER ( ORDER BY INTERNAL_FUNCTION("SALARY") DESC )
             <=CASE  WHEN (5>=0) THEN 5 ELSE 0 END +6)
```

# FETCH N ROWS ONLY

## 制限事項

- FOR UPDATE句と一緒に使えない。
- シーケンスと一緒に使えない。

# In-Database Archiving



# In-Database Archiving

## 不要データを論理的に不可視にする機能

- テーブル内にデータは格納したままでレコードを不可視 (=Archiving) にできる。
  - SQL文のWHERE句の変更不要
- 機能を利用するテーブルにはCREATE | ALTER TABLE ROW ARCHIVAL文を実行。
  - ORA_ARCHIVE_STATE非表示列が利用できるようになる。ALL_TAB_COLUMNSビューには表示されないが、ALL_TAB_COLSビューでは確認できる。
  - ORA_ARCHIVE_STATE列値を '0' (0x30) 以外に設定した列は、WHERE句の条件から自動的に外れる。
- アーカイブされたレコードを参照方法も提供される。
  - ALTER SESSION SET ROW ARCHIVAL VISIBILITY文で可視条件を変更可能。

# In-Database Archiving

使い方: 対象レコードをアーカイブ化

```
SQL> ALTER TABLE employees ROW ARCHIVAL ;
表が変更されました。
SQL> SELECT COUNT(*) FROM employees ;
COUNT(*)
----------
       107
SQL> UPDATE employees SET ORA_ARCHIVE_STATE = '1' WHERE employee_id = 202 ;
1行が更新されました。
SQL> SELECT COUNT(*) FROM employees ;
COUNT(*)
----------
       106
```

# In-Database Archiving

使い方: アーカイブ化されたレコードを参照

```
SQL> ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ALL ;
セッションが変更されました。
SQL> SELECT COUNT(*) FROM employees ;
 COUNT(*)
----------
       107
SQL> ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ACTIVE ;
セッションが変更されました。
SQL> SELECT COUNT(*) FROM employees ;
 COUNT(*)
----------
       106
```

# In-Database Archiving

## 実行計画

```
SQL> EXPLAIN PLAN FOR SELECT * FROM employees ;
解析されました。

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            |     1 |    69 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMPLOYEES  |     1 |    69 |     3   (0)| 00:00:01 |
------------------------------------------------------------------------------------

  1 - filter("EMPLOYEES"."ORA_ARCHIVE_STATE"='0')
```

# In-Database Archiving

## 注意点

- 仮想的にレコードを不可視にしているだけであり、制約は有効。
  - SELECT文を実行して条件に一致したレコードが存在しないのに、INSERT文で一意制約違反という事態がありうる。
- ROW ARCHIVAL指定されたテーブルには1バイト列が複数追加。
  - ORA_ARCHIVE_STATE列 (VARCHAR2(1)型) = 指定された値を保存
  - SYS_NC{99999}$列 (RAW(1)型) = 用途不明(CREATE TABLE時は作成されない)。
  - ROW ARCHIVALを指定を指定したテーブルは列数の制限が小さくなる。
- 使用を解除する場合はALTER TABLE NO ROW ARCHIVAL文を実行。
  - 該当テーブルから列を削除する処理を行う(ALTER TABLE DROP COLUMNと同じ)。
  - 大規模テーブルで実行する場合には I/Oに注意。

# SEQUENCEの拡張

# SEQUENCEの拡張

## シーケンス・オブジェクトの拡張（一部ドキュメント上に存在しない）

- CREATE SEQUENCE sequence_name SESSION
  - セッション単位で初期化されるシーケンス。
  - 永続化されない。
  - SESSIONの代わりにGLOBALを指定するか省略すると従来のシーケンスとなる。
  - SQL*Loader の制御ファイルに使用可能、使用方法は従来と同じ。
  - SQL Referenceマニュアルは何を言っているのかわからない。
- CREATE SEQUENCE sequence_name KEEP | PARTITION number
  - 用途不明。
  - ALL_SEQUENCESビューのPARTITION_COUNT列、KEEP_VALUE列に反映。
- ALL_SEQUENCESビューの拡張
  - PARTITION_COUNT, SESSION_FLAG, KEEP_VALUE列が追加。

# 非表示列

# 非表示列

## 存在しないように見える列

- DESCRIBLEコマンドや、INSERT文/SELECT * 文から列を隠す機能。
  - 列にINVISIBLE属性（またはVISIBLE属性）を指定する。
- 明示的に列名を指定した場合は使用可能。
  - INSERT文で列リストを指定 – INSERT INTO table (column1, column2, …)
  - SELECT文で列リストを指定 – SELECT column1, column2, …
  - UPDATE文で列名を指定 – UPDATE table SET column1 = value …
  - WHERE条件で列名を指定 – WHERE column = value …
- セキュリティ向上目的には使えない

# 非表示列

実行例

```
SQL> CREATE TABLE tblinv1 (col1 CHAR, col2 CHAR INVISIBLE, col3 CHAR);
表が作成されました。
SQL> DESCRIBE tblinv1
名前                                      NULL?    型
----------------------------------------- -------- ----------------------------
C1                                                 CHAR(1)
C3                                                 CHAR(1)

SQL> INSERT INTO tblinv1 VALUES ('A', 'B', 'C') ;
ORA-00913: 値の個数が多すぎます。
SQL> INSERT INTO tblinv1(col1, col2, col3) VALUES ('A', 'B', 'C') ;
1行が作成されました。
```

# 非表示列

実行例

```
SQL> ALTER TABLE tblinv1 MODIFY (col2 VISIBLE) ;
表が変更されました
SQL> DESCRIBE tblinv1
名前                                      NULL?    型
----------------------------------------- -------- ----------------------------
COL1                                                CHAR(1)
COL3                                                CHAR(1)
COL2                                                CHAR(1)
```

- VISIBLEを指定しても列は元の定義位置に戻らず、列定義の最後に追加される。
- 複数のINVISIBLE 列が存在する場合は定義順にソートされる。
- SQL*PlusでINVISIBLE列を指定する場合は「SET COLINVISIBLE ON」を実行。

# アーキテクチャーの変更とシステム管理

# オンライン操作

## ロックを行わないDDL実行

- ONLINE句が指定できるDDLが増加
  - DROP INDEX ONLINE
  - ALTER INDEX UNUSABLE ONLINE
  - ALTER TABLE SET UNUSED ONLINE
  - ALTER TABLE DROP ONLINE
  - ALTER TABLE MOVE PARTITION ONLINE
  - ALTER TABLE MOVE SUBPARTITION ONLINE

# オンライン操作

## ロックを行わずにデータ移動

- データファイルの移動
  - ALTER DATABASE MOVE DATAFILE '旧パス' TO '新パス'
  - オフライン化不要、かつ物理ファイル実体の移動まで1文で実行。
  - 内部的にはファイル・コピー＋旧ファイル削除であるため、一時的に2ファイル作成される。

- セグメントの移動
  - ALTER TABLE table_name MOVE PARTITION partition_name ONLINE …
  - DMLのロックを行わない。
  - この機能ができたのでADOが利用できる。
  - テーブル／パーティション属性ROW MOVEMENTの設定に依存せずに移動が可能。
  - パーティション内のレコードのROWIDは変更される。

# マルチスレッド

## バックグラウンド・プロセスとサーバー・プロセスのスレッド化と集約

- バックグラウンド・プロセスのスレッド化
  - 初期化パラメータthreaded_executionをtrueに設定（デフォルト値 false）（要再起動）
  - インスタンスを以下のプロセスに集約
    - ora_pmon_{SID}, ora_psp0_{SID}, ora_vktm_{SID}, ora_dbw0_{SID}, ora_u004_{SID}, ora_u005_{SID}, …
- サーバー・プロセスのスレッド化
  - listener.oraファイルのパラメータDEDICATED_THROUGH_BROKER_{リスナー名} をonに設定（デフォルト off）
  - バックグラウンド・プロセスのスレッド化が前提
- 接続時にユーザー名／パスワードが必須になる（connect / as sysdba不可）。

# マルチスレッド

実行例

```
$ ps -ef | grep ora_ | grep -v grep
oracle   15316     1  0 18:05 ?        00:00:00 ora_pmon_ORCL1
oracle   15318     1  0 18:05 ?        00:00:00 ora_psp0_ORCL1
oracle   15326     1  1 18:05 ?        00:00:00 ora_vktm_ORCL1
oracle   15330     1  1 18:05 ?        00:00:00 ora_u004_ORCL1
oracle   15336     1  9 18:05 ?        00:00:05 ora_u005_ORCL1
oracle   15342     1  0 18:05 ?        00:00:00 ora_dbw0_ORCL1
```

共有メモリーの構造は変更なし

# PGA制限

PGAの使用量を一定量に制限する

- 初期化パラメータpga_aggregate_limitに制限値を指定する。
  - 制限を超えると最もメモリー使用量が多い(highest untunable PGA memory)コネクションに対してORA-4036を発生。
  - 値の制限は 2GB～(物理メモリー – SGA) × 120 %

- 旧バージョンまでは、初期化パラメータpga_aggregate_targetを指定。
  - メモリー使用量は事実上制限できなかった。

# Unified Auditing

## 監査機能の統合

- 従来の標準監査、SYS監査、ファイングレイン監査を統合。
- 標準状態では両方有効(Mixed Mode)になっている。
  - Unified Auditingのみに設定するにはOracle Databaseプロダクトの再リンクが必要
- 新しい機能
  - CREATE AUDIT POLICY文で監査設定を行う。
  - 監査システム全体の設定はDBMS_AUDIT_MGMTパッケージを使用。
  - 監査データの書き込み方法(同期／非同期)を選択可能
    - 書込みはGEN0バックグラウンド・プロセスが実行
    - 非同期を選択した場合は3秒ごとにフラッシュ
  - 統一されたビューUNIFIED_AUDIT_TRAILから監査データを参照
    - 元データとなるAUDSYS.CLI_SWP$*テーブルは読み取り専用
  - Data Pump / SQL*Loaderの実行も監査可能

# Unified Auditing

## 監査ポリシーの作成

監査ポリシーを作成するにはCREATE AUDIT POLICY文で行う。複数の監査対象を単一のAUDIT POLICYに指定可能。

・権限の監査

CREATE AUDIT POLICY ポリシー名 PRIVILEGES システム権限, …

・ロールの監査

CREATE AUDIT POLICY ポリシー名 ROLES ロール, …

・アクションの監査

CREATE AUDIT POLICY ポリシー名 ACTIONS オブジェクト権限 ON …

CREATE AUDIT POLICY COMPONENNT=DATAPUMP, DV, DIRECT_LOAD, …

・条件を指定するWHEN句等を指定することができる

# Unified Auditing

## 監査ポリシーの有効化と確認

監査ポリシーを有効化するにはAUDIT POLICY文で行う。

・ポリシーの有効化
　AUDIT POLICY ポリシー名
・ポリシーの確認ビュー
　AUDIT_UNIFIED_POLICIES　　ポリシー情報
　AUDIT_UNIFIED_ENABLED_POLICIES　有効化されているポリシー情報

# Unified Auditing

## 標準で有効になっている監査ポリシー

| ポリシー名 | 設定 | 対象 | 成功 | 失敗 |
|---|---|---|---|---|
| ORA_SECURECONFIG | ユーザー、<br>ロール、<br>プロファイル、<br>PDB、<br>データベース・リンク、<br>パブリック・シノニム、<br>監査、<br>ディレクトリ等の作成／削除／変更等 | 全ユーザー | YES | YES |
| ORA_LOGON_FAILURES | クライアント接続 | 全ユーザー | NO | YES |

# Patch Set 12.1.0.2 情報

# Patch Set 12.1.0.2 で追加された機能一覧

| 新機能 （参考 https://blogs.oracle.com/otnjp/entry/database_12c_new_feature） | |
|---|---|
| Advanced Index Compression | PDB File Placement in OMF |
| Approximate Count Distinct | PDB Logging Clause |
| Attribute Clustering | PDB Metadata Clone |
| Automatic Big Table Caching | PDB Remote Clone |
| FDA Support for CDBs | PDB Snapshot Cloning Additional Platform Support |
| Full Database Caching | PDB STANDBYS Clause |
| In-Memory Aggregation | PDB State Management Across CDB Restart |
| In-Memory Column Store | PDB Subset Cloning |
| JSON Support | Rapid Home Provisioning |
| New FIPS 140 Parameter for Encryption | Zone Maps |
| PDB CONTAINERS Clause | |

# Patch Set 12.1.0.2で追加された初期化パラメータ

| 初期化パラメータ | |
|---|---|
| common_user_prefix | inmemory_max_populate_servers |
| db_performance_profile | inmemory_trickle_repopulate_servers_percent |
| DBFIPS_140 | instant_restore |
| enable_goldengate_replication | java_restrict |
| inmemory_size | optimizer_inmemory_aware |
| inmemory_clause_default | pdb_os_credential |
| inmemory_force | pdb_lockdown |
| inmemory_query | |

# Patch Set 12.1.0.2で変更された初期化パラメータ

| 初期化パラメータ | 変更点 | 12.1.0.1 | 12.1.0.2 |
|---|---|---|---|
| resource_limit | デフォルト変更 | false | true |
| compatible | デフォルト変更 | 12.1.0.1 | 12.1.0.2 |
| optimizer_features_enable | デフォルト変更 | 12.1.0.1 | 12.1.0.2 |
| sec_protocol_error_further_action | デフォルト変更 | CONTINUE | (DROP,3)<br>（マニュアルではCONTINUE） |
| sec_max_failed_login_attempts | デフォルト変更 | 10 | 3 （マニュアルでは10） |
| audit_sys_operations | デフォルト変更 | false | true（マニュアルではfalse） |
| log_archive_local_first | 削除 | | |
| parallel_fault_tolerance_enabled | 削除 | | |

マニュアルと差異がある初期化パラメータはLinux x86-64版で確認。

# Thank you