
ORACLE®

Oracle Database 12*c* Release 2 CoreTech Seminar

12.2.0.1
Recovery Manager & Online Operation

日本オラクル株式会社
クラウド・テクノロジー事業統括
Cloud/Big Data/DISプロダクト本部
南野　英梨子
Database & Exadataプロダクトマネジメント本部
佐々木亨
2016/10

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

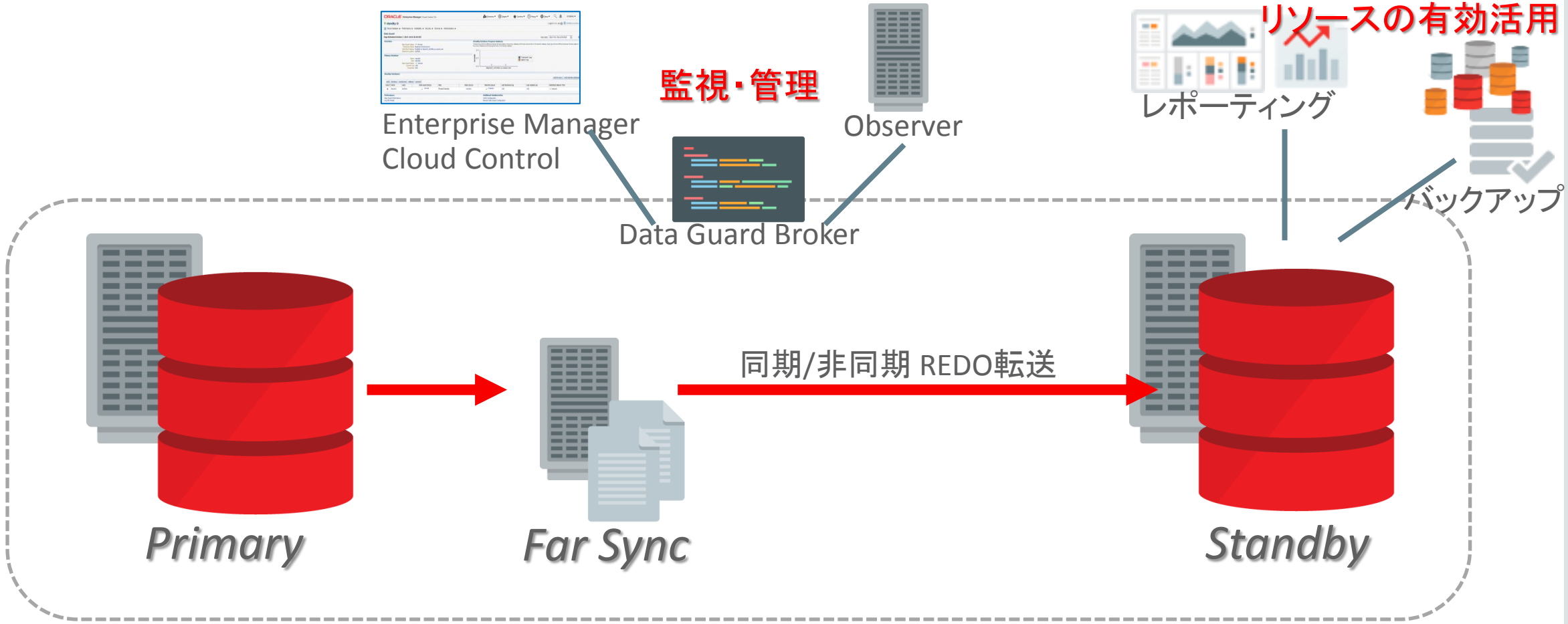
Agenda

1. Active Data Guard/Data Guard 12.2 新機能
2. Recovery Manager (RMAN) 12.2新機能
3. Online Operation 12.2新機能

Active Data Guard/Data Guard 12.2新機能

Active Data Guard/Data Guard

データ保護・可用性の提供 + 適切なROIを実現



Active Data Guard/Data Guard 12.2 まとめ

管理性

- ✓ パスワードファイルの自動同期
- ✓ PDB サブセット・スタンバイ
- ✓ スタンバイ・データベース/Far Sync作成
- ✓ NOLOGGING 処理に対する検証やリカバリ
- ✓ Data Guard Broker の操作性の向上
- ✓ REST API 対応

リソース有効活用

- ✓ ADG のDatabase In-Memory 対応
- ✓ 複数インスタンスでのパラレルREDO適用
- ✓ AWR リモート・スナップショット
- ✓ SQL Tuning Advisor サポート

データ保護

- ✓ 複数スタンバイ環境での同期転送の向上
- ✓ プライマリとスタンバイの比較によるプロアクティブな検証
- ✓ REDO転送先の自動切り替えの強化
- ✓ ファスト・スタート・フェイルオーバー複数ターゲット・スタンバイ
- ✓ 最大保護モードでのファスト・スタート・フェイルオーバー
- ✓ ファスト・スタート・フェイルオーバーの複数オブザーバー

計画/計画外停止の切り替え

- ✓ ローリング・アップグレードとの連携強化
- ✓ Data Guard Broker のMTA
- ✓ 実行中の処理を捌いた後のスイッチオーバー
- ✓ Active Data Guard の接続済コネクションの継続

管理性

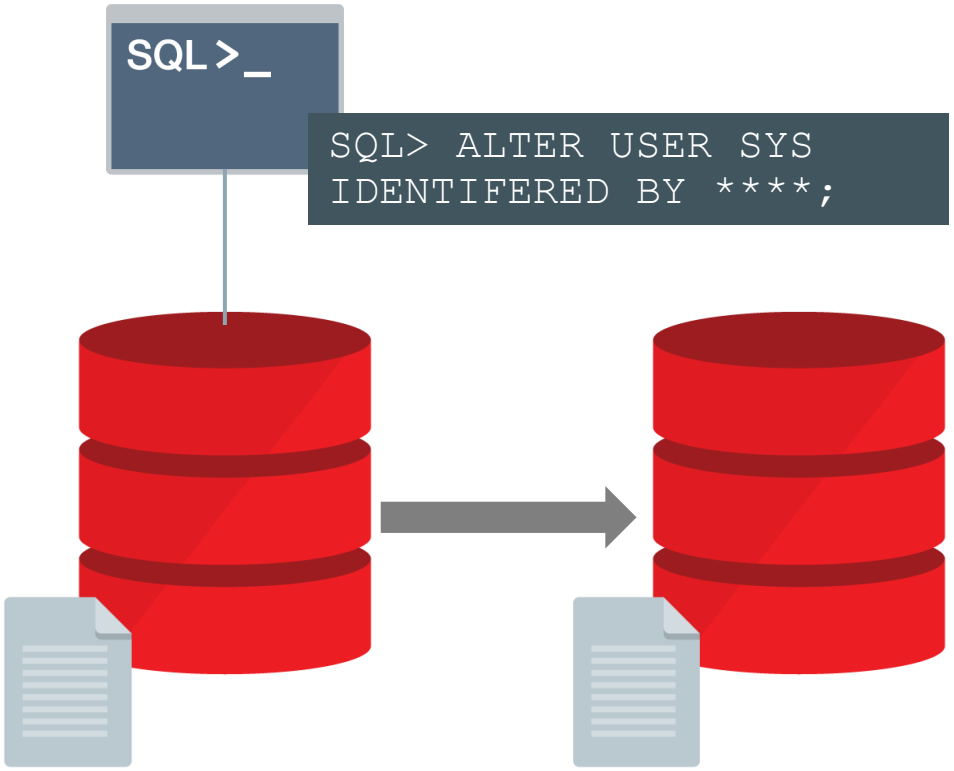
- ✓ パスワードファイルの自動同期
- ✓ PDB サブセット・スタンバイ
- ✓ スタンバイ・データベース/Far Sync作成
- ✓ NOLOGGING 処理に対する検証やリカバリ
- ✓ Data Guard Broker の操作性の向上

NEW IN 12.2

パスワードファイルの自動同期

プライマリでのパスワード変更を自動的に反映

- プライマリでパスワードファイルで認識されているユーザーのパスワード変更時、パスワードファイルが更新される
- 12.1まで:スタンバイに手動コピーが必要
- 12.2から:
  - REDO 適用による自動同期
    - メンテナンス時の作業漏れ防止
    - Far SyncはREDO適用をしないため、手動コピーが必要
  - SSL証明書のための新しいREDO転送認証プロトコル‘redo_transport_user’

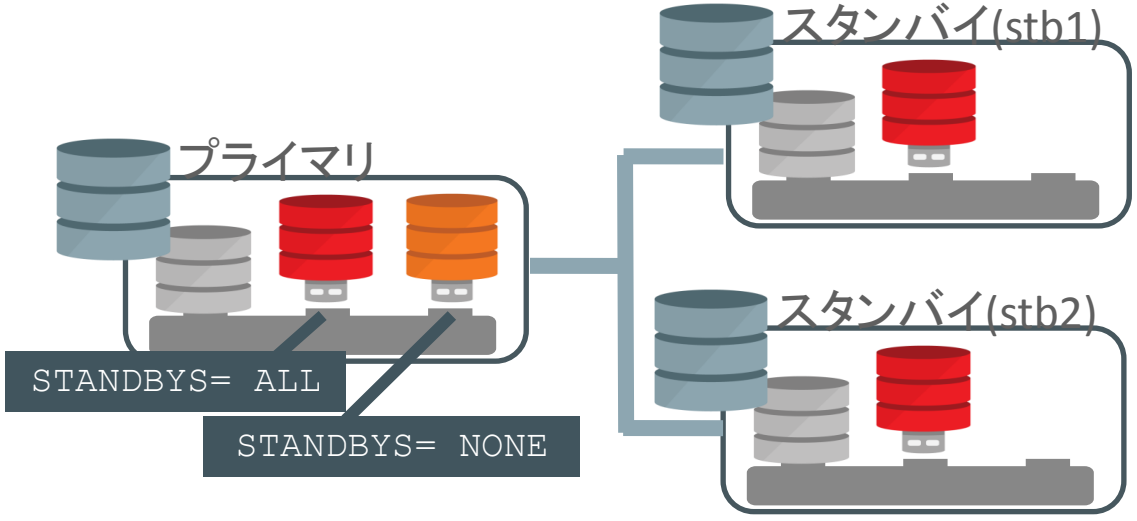


PDB サブセット・スタンバイ

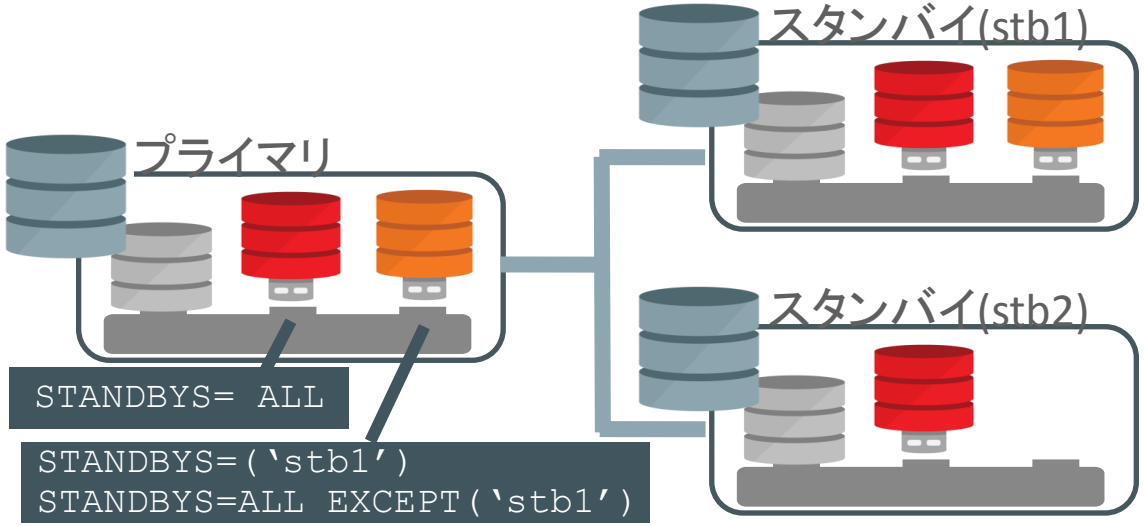
NEW IN 12.2

PDBのスタンバイへの同期に関する設定の強化

- 12.1 : 全スタンバイに同期する(ALL)か同期しない(NONE)の指定が可能



- **12.2～** : さらに**同期させるスタンバイの指定**が可能 ＋スタンバイ側のパラメータで、**同期させるPDB名の指定**が可能



✓ PDB単位でのDR要件定義が可能になり、全体でのリソース使用量の削減が可能

✓ スイッチオーバー/フェイルオーバー後は、切り替え前後で構成が異なることに注意

NEW IN 12.2

PDB サブセット・スタンバイ

設定方法

- 指定したスタンバイに同期

```
SQL> CREATE PLUGGABLE DATABASE … STANDBYS=('<スタンバイ名>');
```

- 指定したスタンバイ以外に同期

```
SQL> CREATE PLUGGABLE DATABASE … STANDBYS=ALL EXCEPT('<スタンバイ名>');
```

- スタンバイ側でREDO適用を有効化するPDBを制御することも可能
  - 適用するPDB名を指定(デフォルト=＊)

```
SQL> ALTER SYSTEM SET ENABLED_PDBS_ON_STANDBY="<PDB名>" SCOPE=BOTH;
```

参考)MTA 環境での Data Guard の注意事項

Doc ID 2049127.1: Data Guard Impact on Oracle Multitenant Environments

ORACLE

PDB サブセット・スタンバイ

確認方法

- V$PDBS : REDO適用が有効になっているPDB名を確認
  - 作成時に対象外にしたスタンバイでは、下記のコマンド結果で対象PDB名が表示されない

```
SQL> select name, open_mode,recovery_status from v$pdbs;

NAME                 OPEN_MODE                      RECOVERY_STATUS
-------------------- ------------------------------ -----------------------
PDB$SEED             READ ONLY                      ENABLED
PDB01                MOUNTED                        ENABLED
PDB02                MOUNTED                        ENABLED
```

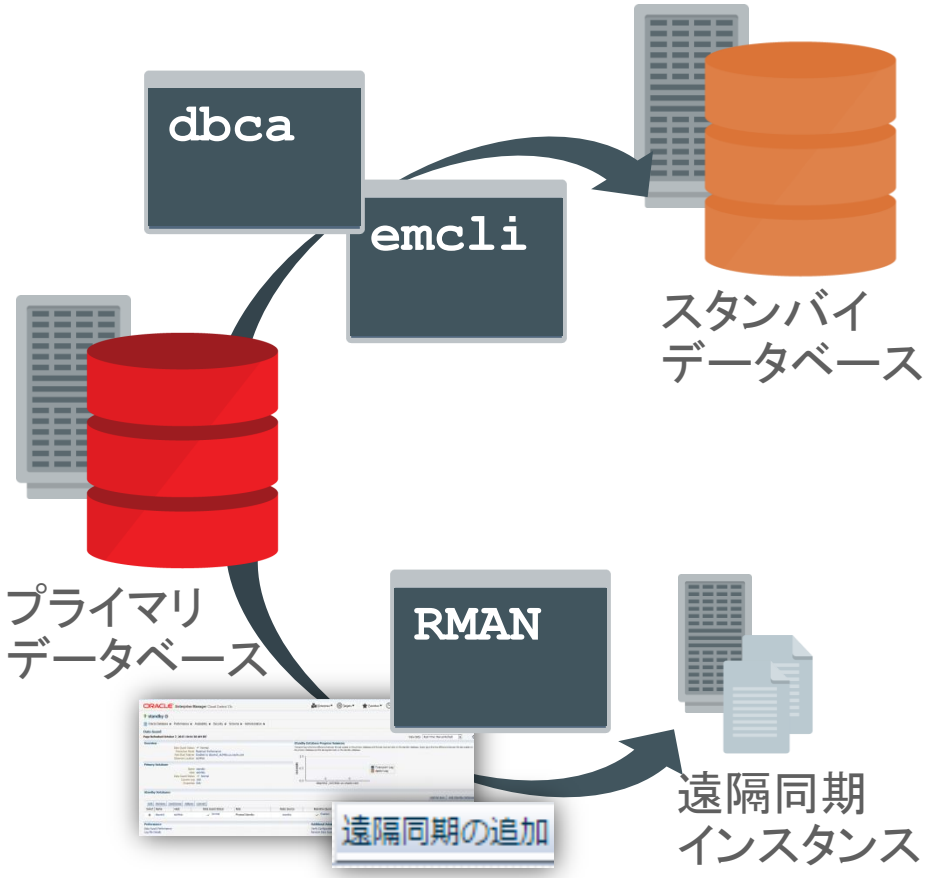
スタンバイ・データベース/Far Syncインスタンス作成

NEW IN 12.2

作成方法の強化

- スタンバイ・データベースの作成
  - DBCA サイレントモードによる作成をサポート
  - EMCLI による作成をサポート

- Far Syncインスタンスの作成
  - RMAN DUPLICATE コマンドでの作成をサポート
  - EMCC からのUIによる作成をサポート



DBCA によるスタンバイ・データベース作成

シンプルでスクリプト化が可能

NEW IN 12.2

- スタンバイ側からdbca –silent コマンドを実行

```
dbca -silent -createDuplicateDB -gdbName BOS.domain.com -sid BOS1
-sysPassword oracle
-primaryDBConnectionString MyHost.domain.com:1521/bos.domain.com
-createAsStandby -dbUniquename BOS1
```

- 制限
  - プライマリがシングル・データベース/Non-CDB の環境のみ可能

EMCLI によるスタンバイ・データベース作成

シンプルでスクリプト化が可能

NEW IN 12.2

- EMCLIでもコマンドベースでのスタンバイの作成が可能

```
emcli create_standby -source_db_target_name="BOS"
-source_db_target_type="rac_database"
-dest_oracle_sid="BOS1_1"
-spname="BOS1"
-use_broker
```

  - デフォルトは、RMAN duplicate from active database で行われる
  - 作成時にBroker も有効化した状態で完了させることが可能
    - EMCC のData Guard 管理ページで監視/管理するためには、Broker構成は有効が必要

RMAN DUPLICATE でのFar Syncの作成

スタンバイ・データベース作成時と同様の手順でFar Syncも作成可能

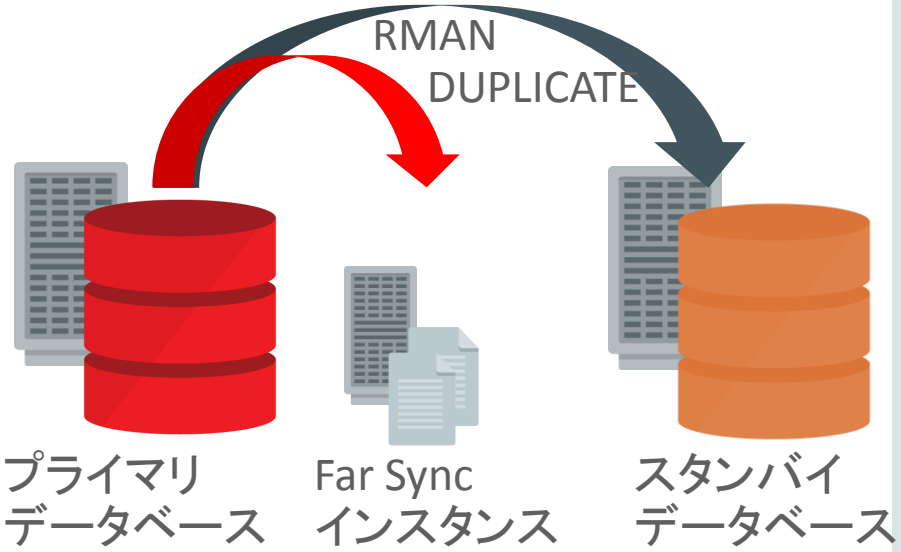
ADG NEW IN 12.2

- 12.1から実装されたFar Sync
- RMAN DUPLICATE コマンドを使ったFar Sync作成をサポート
  - プライマリもしくはバックアップから

```
DUPLICATE TARGET DATABASE FOR FARSYNC FROM ACTIVE DATABASE;
DUPLICATE TARGET DATABASE FOR FARSYNC BACKUP LOCATION …
```

- スタンバイ作成時と同様の手順で可能

```
connect target sys@SFO;
connect auxiliary sys@SFOFS;
run {allocate channel prmy1 type disk;
   allocate auxiliary channel stby1 type disk;
   duplicate target database for farsync
     from active database
     spfile parameter_value_convert ('SFO','SFOFS')
        set 'db_unique_name'='SFOFS';}
```

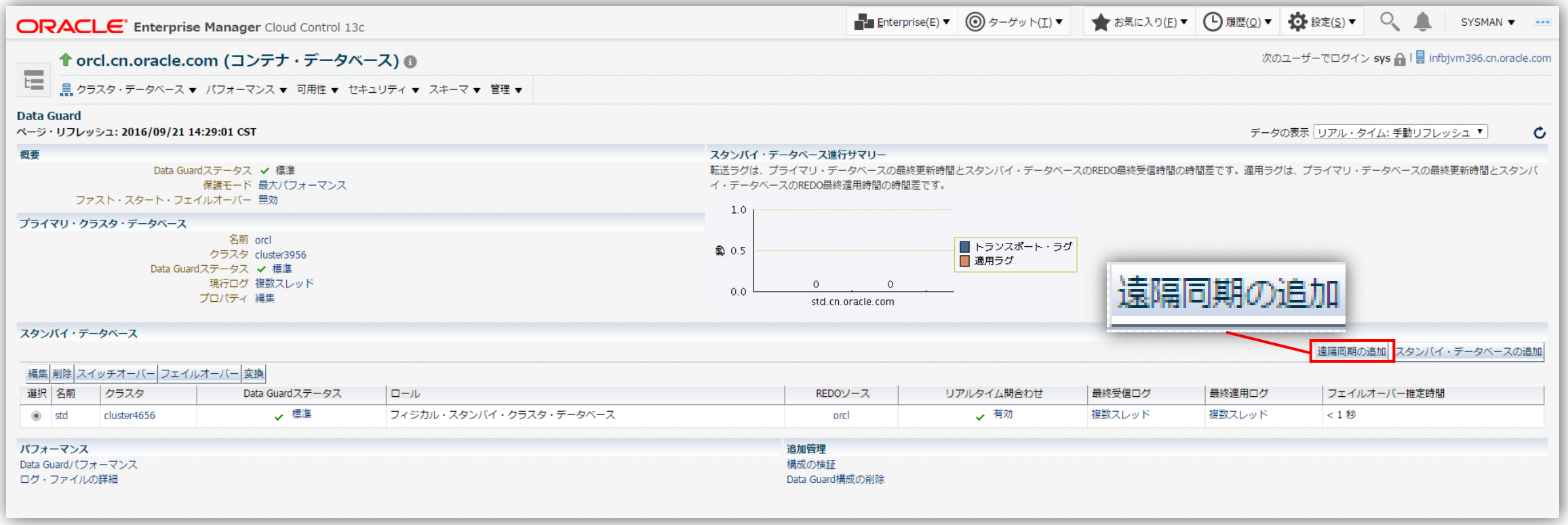


Enterprise Manager Cloud Controlでの Far Sync作成

作成と管理/監視に対応

ADG NEW IN 12.2

- UI ベースで可能になり、より作成/管理/監視が簡単に



スタンバイ・データベース/Far Sync作成
方法一覧 (12.2)

	スタンバイ・データベース	Far Syncインスタンス
手動コピーによる作成 (マニュアル記載基本作成方法)	○	○
RMAN DUPLICATE FROM ACTIVE DATABASE	○ Push-Base / Pull-Base(12.1～)	○ **NEW in 12.2**
RMAN DUPLICATE FROM BACKUP	○	○ **NEW in 12.2**
DBCA サイレントモード	○ **NEW in 12.2** ※ プライマリがシングル+ Non-CDB の場合のみ	-
EMCC によるUIベース	○	○ **NEW in 12.2**
EMCLI	○ **NEW in 12.2**	-

✓ *StandbyとFar Syncを同様の手順で作成可能*
✓ *スクリプト化が容易*

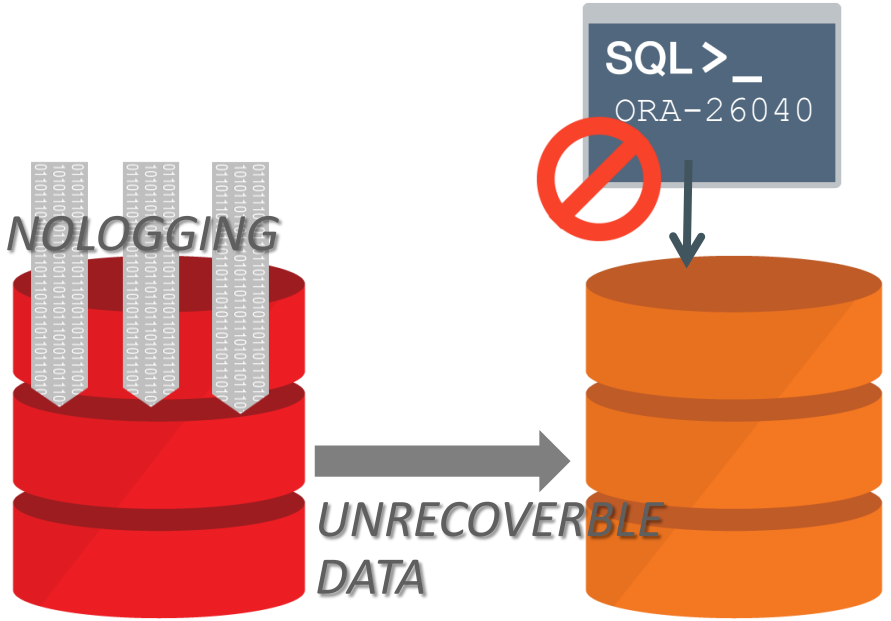
✓ *UIベースで簡単に*

✓ *Broker構成がデフォルトで有効*

NOLOGGING 処理に対する検証やリカバリ

DWH向けの機能強化

- データロードなどREDO生成を抑えて実行時間を短縮させる際にNOLOGGINGを使うケース
  - ダイレクト・パス・インサートなど
  - NOLOGGING の処理は、リカバリ不可なブロックとして情報のみが転送
  - スタンバイに反映させるためにリカバリ作業が必要
    - スタンバイ側で該当データにアクセスした際のエラー




```
SQL> select count(*) from tab1;
select count(*) from tab1
                     *
行1でエラーが発生しました。:
ORA-01578:Oracleデータ・ブロックに障害が発生しました(ファイル番号15、ブロック番号131)
ORA-01110: データファイル15:'+DATA/xxx/xxx/DATAFILE/ts1.347.922632353'
ORA-26040: データ・ブロックがNOLOGGINGオプションを使用してロードされました。
```

NOLOGGING 処理に対する検証やリカバリ

NEW IN 12.2

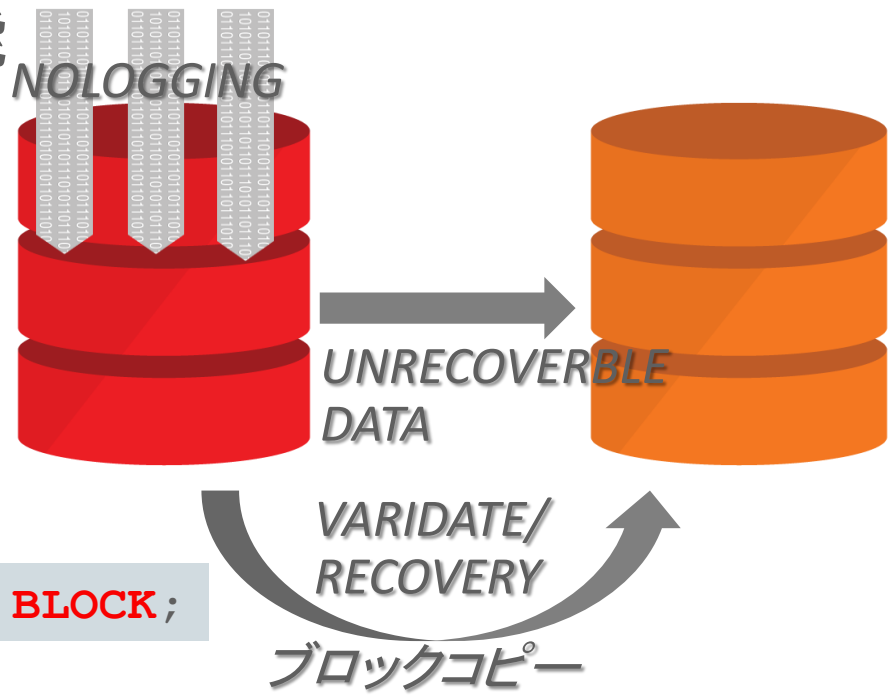
DWH向けの機能強化

- 12.2から:スタンバイでNOLOGGINGで更新されたブロックに対するVARIDATE/RECOVERが可能
  - RMAN による検証とリカバリで、修復が簡単に
    - アクセスしないと気付けなかったNONLOGGED BLOCK に気付け、1コマンドでリカバリが可能に
  - データベース全体 or データファイルごとに実行
  - 実行時は、管理リカバリモード(MRP)の停止が必要
    - 検証

```
RMAN> VALIDATE DATABASE/DATAFILE ...NONLOGGED BLOCK;
```

    - リカバリ

```
RMAN> RECOVER DATABASE/DATAFILE ... NONLOGGED BLOCK;
```



NOLOGGING 処理に対する検証やリカバリ

NEW IN 12.2

参考) 1. スタンバイ側でNOLOGGING 処理のブロックを確認

- 管理リカバリモード(MRP)の停止

```
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
```

- 下記のいずれかの方法で検証
  - 特定のデータファイルを検証

```
RMAN> VALIDATE DATAFILE <データファイルID> NONLOGGED BLOCK;
```

  - データベース全体を検証

```
RMAN> VALIDATE DATABASE NONLOGGED BLOCK;
```

- V$NONLOGGED_BLOCK を確認

```
SQL> select  FILE#,BLOCK#,BLOCKS,NONLOGGED_START_CHANGE#,NONLOGGED_END_CHANGE#
     from V$NONLOGGED_BLOCK;

     FILE#     BLOCK#     BLOCKS NONLOGGED_START_CHANGE# NONLOGGED_END_CHANGE#
---------- ---------- ---------- ------------------------ ----------------------
        15        131         10                  2701076                2701105
```

NOLOGGING 処理に対する検証やリカバリ

NEW IN 12.2

参考) 2. スタンバイ側でNONLOGGED のブロックをリカバリ

- 下記のいずれかの方法でリカバリ
  - 特定のデータファイルをリカバリ

```
RMAN> RECOVER DATAFILE <データファイルID> NONLOGGED BLOCK;
```

  - データベース全体をリカバリ

```
RMAN> RECOVER DATABASE NONLOGGED BLOCK;
```

- V$NONLOGGED_BLOCK で該当ブロックを確認

```
SQL> select  FILE#,BLOCK#,BLOCKS,NONLOGGED_START_CHANGE#,NONLOGGED_END_CHANGE#
     from V$NONLOGGED_BLOCK;

レコードが選択されませんでした。
```

- 管理リカバリモードを開始

```
SQL> RECOVER MANAGED STANDBY DATABASE DISCONNECT;
```

ORACLE

Data Guard Broker の操作性の向上

NEW IN 12.2

DGMGRL コマンドラインの操作拡張

- スクリプトやOSコマンドの実行が可能に
  - SQL*Plus に類似した操作が可能
    - @(スクリプト実行)、/(再実行)、HOST or ! (OSコマンド)
      SPOOL(ログ出力)、SET TIME ON/OFF(時刻表示)

- オブザーバをバックグラウンドで起動が可能に

```
DGMGLR> START OBSERVER BOSOBSERVER LOGFILE IS '/tmp/bosobserver'
        IN BACKGROUND CONNECT IDENTIFIER IS BOS;
```

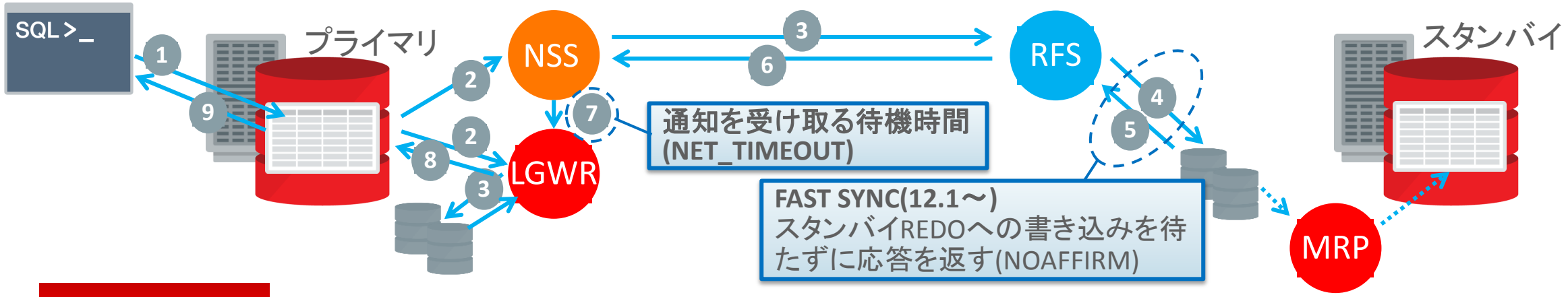
データ保護

- ✓ 複数スタンバイ環境での同期転送の向上
- ✓ プライマリとスタンバイの比較によるプロアクティブな検証
- ✓ REDO転送先の自動切り替えの強化
- ✓ ファスト・スタート・フェイルオーバー複数ターゲット・スタンバイ
- ✓ 最大保護モードでのファスト・スタート・フェイルオーバー
- ✓ ファスト・スタート・フェイルオーバーの複数オブザーバー

複数スタンバイ環境での同期転送の向上

Data Guard 構成全体で複数の同期モードを制御して、プライマリへの影響を軽減

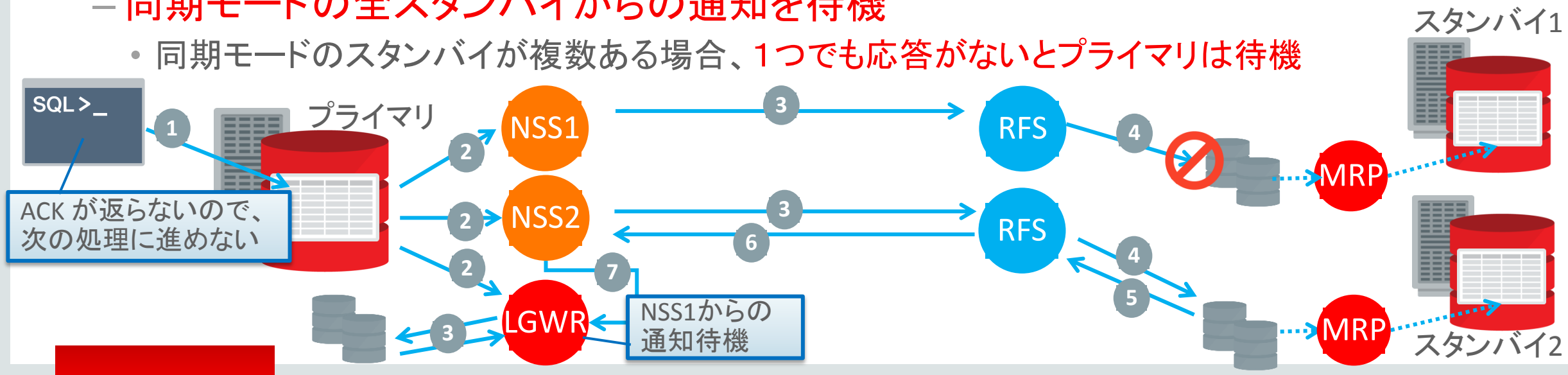
- 同期モードは、スタンバイ側にREDO転送されていることを保証
  - プライマリ側のLGWR プロセスが、通知を受け取る待機時間を、各スタンバイ毎の宛先を設定するLOG_ARCHIVED_DEST_n パラメータの NET_TIMEOUT 値で設定
    - 12.1 から、RFS がREDOを受け取った時点で通知を返すFAST SYNC が実装
  - 同期モードの全スタンバイからの通知を待機
    - 同期モードのスタンバイが複数ある場合、1つでも応答がないとプライマリは待機



複数スタンバイ環境での同期転送の向上

Data Guard 構成全体で複数の同期モードを制御して、プライマリへの影響を軽減

- 同期モードは、スタンバイ側にREDO転送されていることを保証
  - プライマリ側のLGWR プロセスが、通知を受け取る待機時間を、各スタンバイ毎の宛先を設定するLOG_ARCHIVED_DEST_n パラメータの NET_TIMEOUT 値で設定
    - 12.1 から、RFS がREDOを受け取った時点で通知を返すFAST SYNC が実装
  - 同期モードの全スタンバイからの通知を待機
    - 同期モードのスタンバイが複数ある場合、1つでも応答がないとプライマリは待機

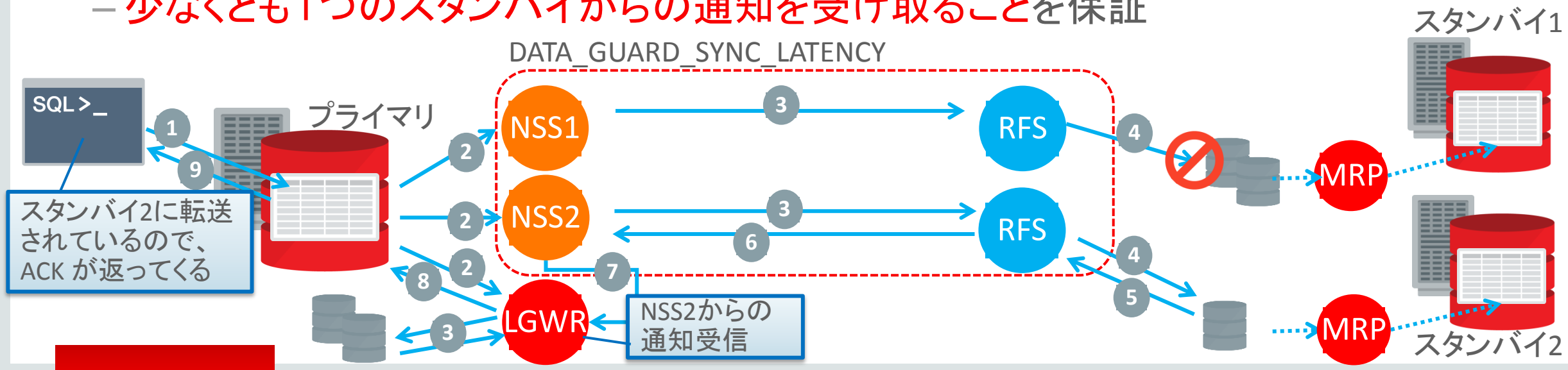


複数スタンバイ環境での同期転送の向上

NEW IN 12.2

Data Guard 構成全体で複数の同期モードを制御して、プライマリへの影響を軽減

- 12.2から：プライマリが通知を受け取るまでの全体としての待機時間
  - DATA_GUARD_SYNC_LATENCY パラメータ
    - 同期モードの全スタンバイの共通設定
    - 設定可能な数値: 0～LOG_ARCHIVE_DEST_*n* のNET_TIMEOUT 値
  - 少なくとも1つのスタンバイからの通知を受け取ることを保証



プライマリとスタンバイの比較

NEW IN 12.2

潜在的なブロック破損に対するプロアクティブな検知が可能に

- プライマリとスタンバイを比較して、Lost Write 等のブロック破損を検知
  - プライマリと1つ以上のスタンバイを比較可能
    - Logical standby、Far Sync 、Cascade Standby は比較対象外
  - データベース全体 or データファイル単位で検証可能
  - DB_LOST_WRITE_PROTECT パラメータの設定がない環境でも検知が可能
    - 検証用REDO生成が不要
    - パラメータ設定時にもダイレクト・パス・リードは検証REDO生成されない
  - 空ブロックはスキップされるため、素早く比較可能
  - dbms_dbcomp.dbcomp プロシージャ or DGMGRL の VALIDATE DATABASE/DATAFILE
    - 実行状況はV$SESSION_LONGOPS で確認可能

プライマリとスタンバイの比較

NEW IN 12.2

DBMS_DBCOMP.DBCOMP プロシージャ

```
SQL> exec sys.dbms_dbcomp.dbcomp
     ('[datafile-name|datafile-number|ALL]', -- 比較対象データファイル
      '<output-file-name>', -- 出力ファイル名
      [true|false] -- ダンプファイルの出力(デフォルト false)
      );
```

- 実行例

```
SQL> declare
       Datafile_Name_or_Number varchar2(1000);
       Output_File varchar2(1000);
     begin
       dbms_output.enable(1000000);
       Datafile_Name_or_Number:= 'all' ;
       Output_File:='BlockCompareAll';
       sys.dbms_dbcomp.DBCOMP(Datafile_Name_or_Number,Output_File,TRUE);
     end;
   /
PL/SQLプロシージャが正常に完了しました。
```

プライマリとスタンバイの比較

NEW IN 12.2

DGMGRL

```
DGMGRL> VALIDATE DATABASE [database-name|ALL] -- スタンバイ・データベース
        DATAFILE [datafile-name|datafile-number|ALL] -- 比較対象データファイル
        OUTPUT="<output-file-name>" -- 出力ファイル名
```

- 実行例

```
DGMGRL> validate database orcl datafile 15 output=check15;
操作にはデータベース"orcl"への接続が必要です
接続中...
orclに接続しました
出力ファイルが/u01/app/oracle/diag/rdbms/orcl/orcl1/trace内に作成されました(ホスト"xxx")
```

プライマリとスタンバイの比較

NEW IN 12.2

実行状況の確認

```
SQL> SELECT target_desc, sofar, totalwork
     FROM V$SESSION_LONGOPS WHERE opname ='Block Compare';
    TARGET_DESC              SO FAR     TOTALWORK
    -------------------- ---------- ----------
    Compared Blocks             23914     340639
    Lost Writes                     0          0
```

出力ファイル内容

```
Summary:
Different data block pairs: 324
*******************************************************************************
                        TOTAL: total no. of blocks found
...
ID TOTAL   CORR SKIPPED DIFFV   SAMEV   SAMEV&C ENCERR  LWLOC  LWRMT DIFFPAIR
00 0104183 0000 0104183 0000000 0000000 0000000 0000000 000000 000000 0000000
02 0021538 0000 0008658 0000027 0012853 0012529 0000000 000000 000000 0000324
...
```

プライマリとスタンバイの比較
実行データベースと状態による比較パターン

NEW IN 12.2

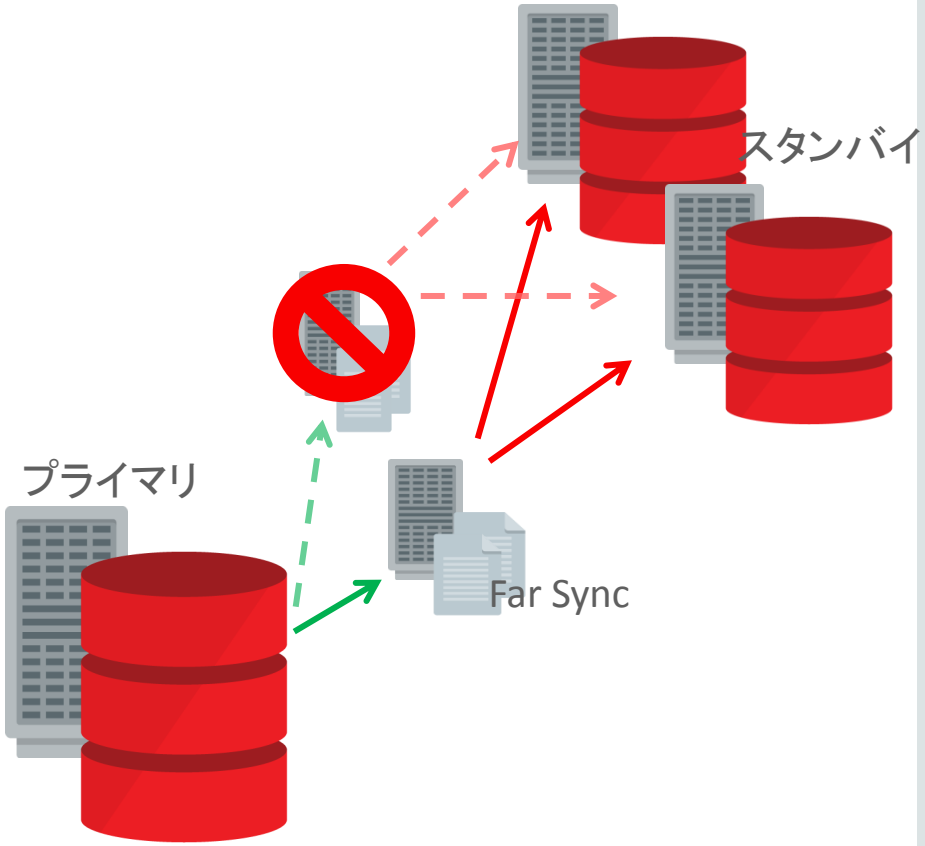
#	コマンド実行	プライマリの状態	スタンバイの状態	比較対象
Case1	プライマリ	MOUNT or OPEN	全スタンバイがMOUNT or R/O	プライマリと全スタンバイを比較
Case2	1スタンバイ	MOUNT or OPEN	全スタンバイがMOUNT or R/O	プライマリと実行スタンバイを比較
Case3	プライマリ	MOUNT or OPEN	一部のスタンバイがMOUNT or R/O	プライマリとMOUNT以上の一部のスタンバイを比較
Case4	1スタンバイ	MOUNT or OPEN	一部のスタンバイ(実行DB含む)がMOUNT or R/O	プライマリと実行スタンバイを比較
Case5	1スタンバイ	NOMOUNT or CLOSE	MOUNT or R/O	エラーが返り、比較不可 (両方MOUNT以上である必要がある)
Case6	プライマリ	MOUNT or OPEN	NOMOUNT or CLOSE	メッセージが返り、比較不可 (両方MOUNT以上である必要がある

REDO転送先の自動切り替えの強化

グループとプライオリティの設定

NEW IN 12.2

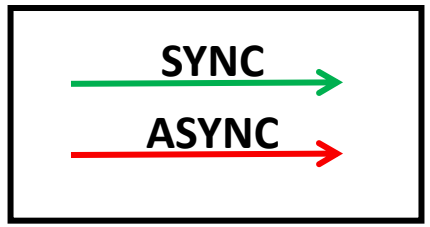
- 12.1から：REDO転送先に転送不可になった際に、宛先を自動的に切り替えが可能
  - LOG_ARCHIVE_DEST_n パラメータのALTERNATE で設定
  - フェイル・バックは自動的に行われる
  - 2つ以上ALTERNATE が設定されている場合は、フェイル・バックは不可
- 12.2から：グループ/プライオリティを設定可能
  - 設定に基づき、どこに切り替えるか、フェイル・バックするかを自動判断　→　意図した切り替えが可能



REDO転送先の自動切り替えの強化

NEW IN 12.2

プライマリに設定するLOG_ARCHIVE_DEST_n パラメータによる宛先の切り替わり例




```
LOG_ARCHIVE_DEST_2=
'SERVICE=fs1 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,_roles);
```

```
LOG_ARCHIVE_DEST_4=
'SERVICE=STB1 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```

右のパラメータは
全てプライマリの設定

FS1(Far Sync)

FS2(Far Sync)

STB1 (スタンバイ)

STB2 (スタンバイ)

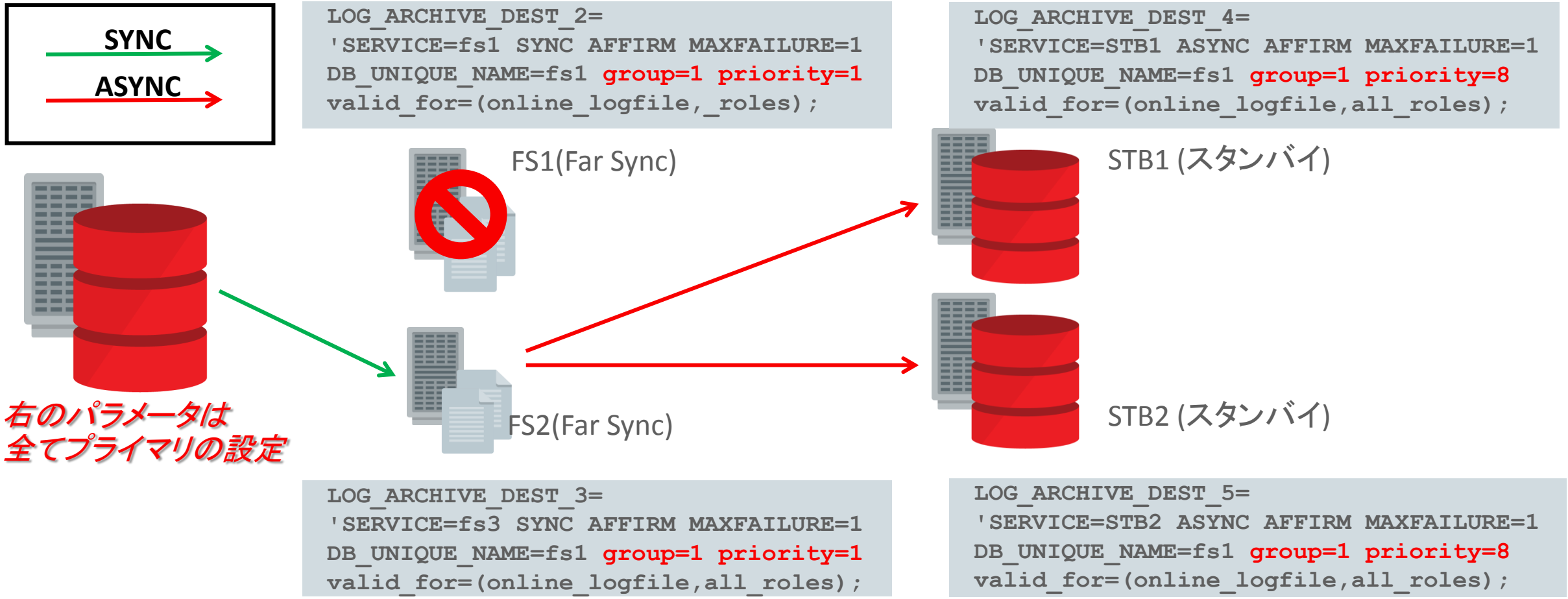
```
LOG_ARCHIVE_DEST_3=
'SERVICE=fs3 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,all_roles);
```

```
LOG_ARCHIVE_DEST_5=
'SERVICE=STB2 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```

REDO転送先の自動切り替えの強化

NEW IN 12.2

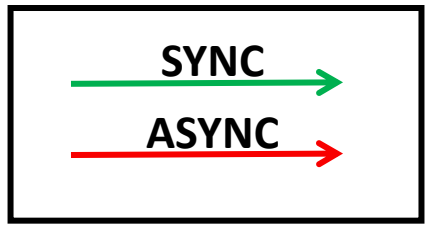
1)FS1停止。同プライオリティのFS2に切り替え、FS2から低プライオリティのSTB1/STB2へ



REDO転送先の自動切り替えの強化

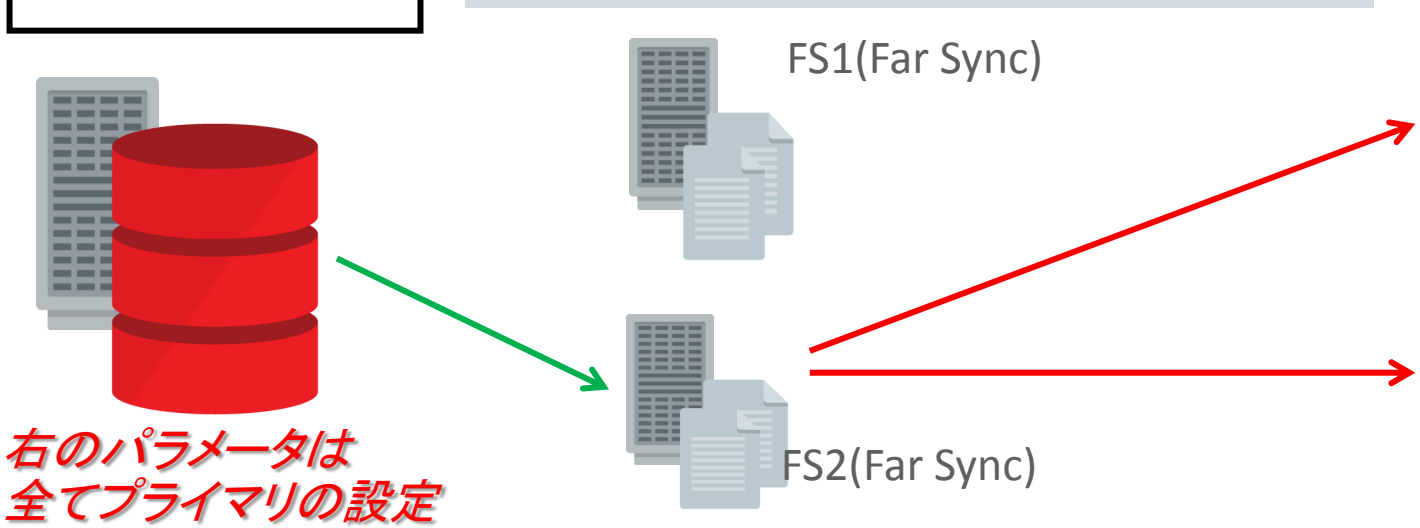
NEW IN 12.2

2)FS1復旧。同プライオリティのため、フェイルバックはしない




```
LOG_ARCHIVE_DEST_2=
'SERVICE=fs1 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,_roles);
```

```
LOG_ARCHIVE_DEST_4=
'SERVICE=STB1 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```



右のパラメータは
全てプライマリの設定

```
LOG_ARCHIVE_DEST_3=
'SERVICE=fs3 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,all_roles);
```

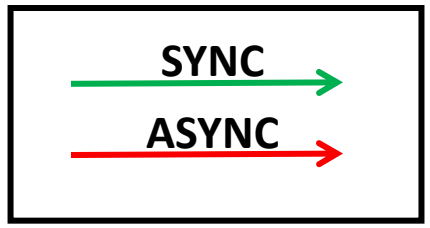
```
LOG_ARCHIVE_DEST_5=
'SERVICE=STB2 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```

ORACLE

REDO転送先の自動切り替えの強化

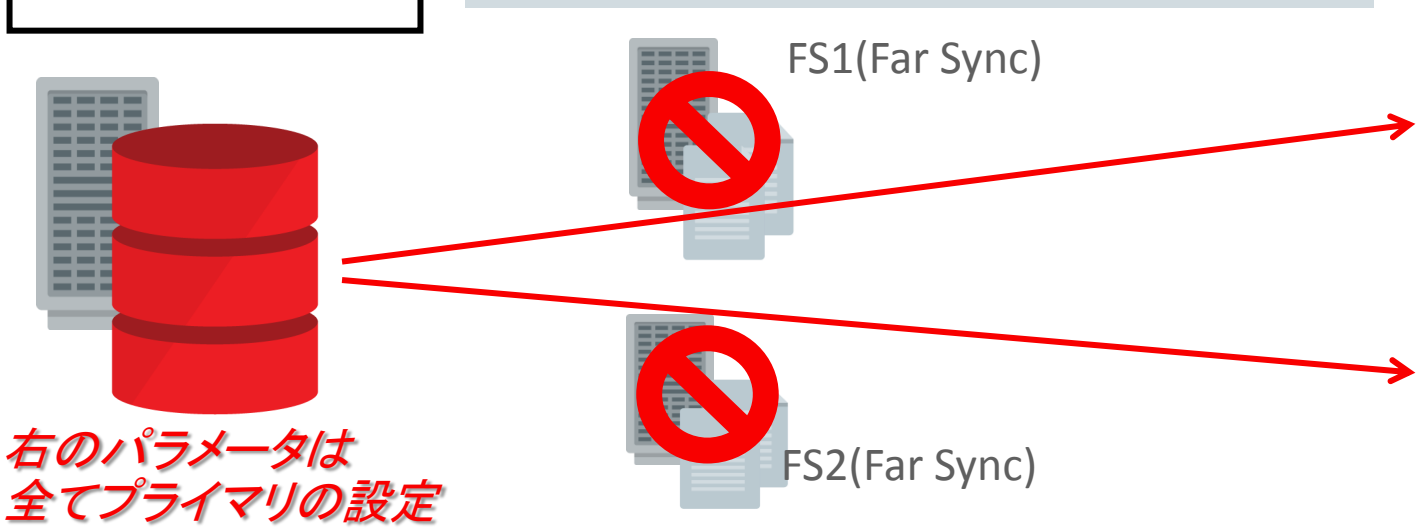
NEW IN 12.2

3)FS1とFS2停止。同プライオリティのSTB1/STB2両方に切り替え




```
LOG_ARCHIVE_DEST_2=
'SERVICE=fs1 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,_roles);
```

```
LOG_ARCHIVE_DEST_4=
'SERVICE=STB1 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```



右のパラメータは
全てプライマリの設定

```
LOG_ARCHIVE_DEST_3=
'SERVICE=fs3 SYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=1
valid_for=(online_logfile,all_roles);
```

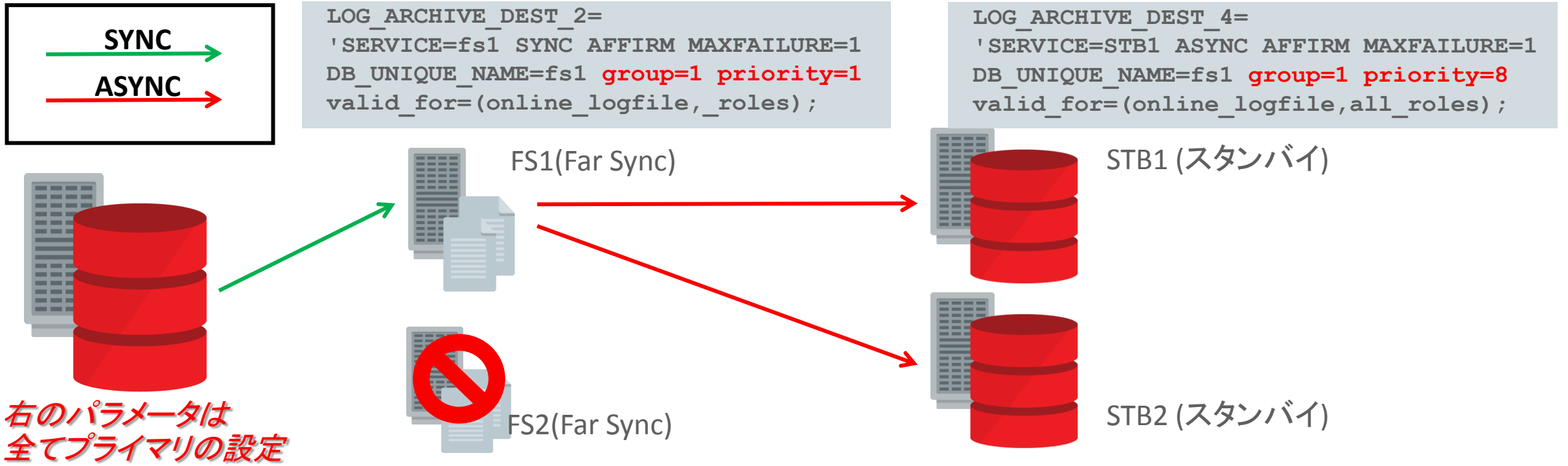
```
LOG_ARCHIVE_DEST_5=
'SERVICE=STB2 ASYNC AFFIRM MAXFAILURE=1
DB_UNIQUE_NAME=fs1 group=1 priority=8
valid_for=(online_logfile,all_roles);
```

ORACLE

REDO転送先の自動切り替えの強化

NEW IN 12.2

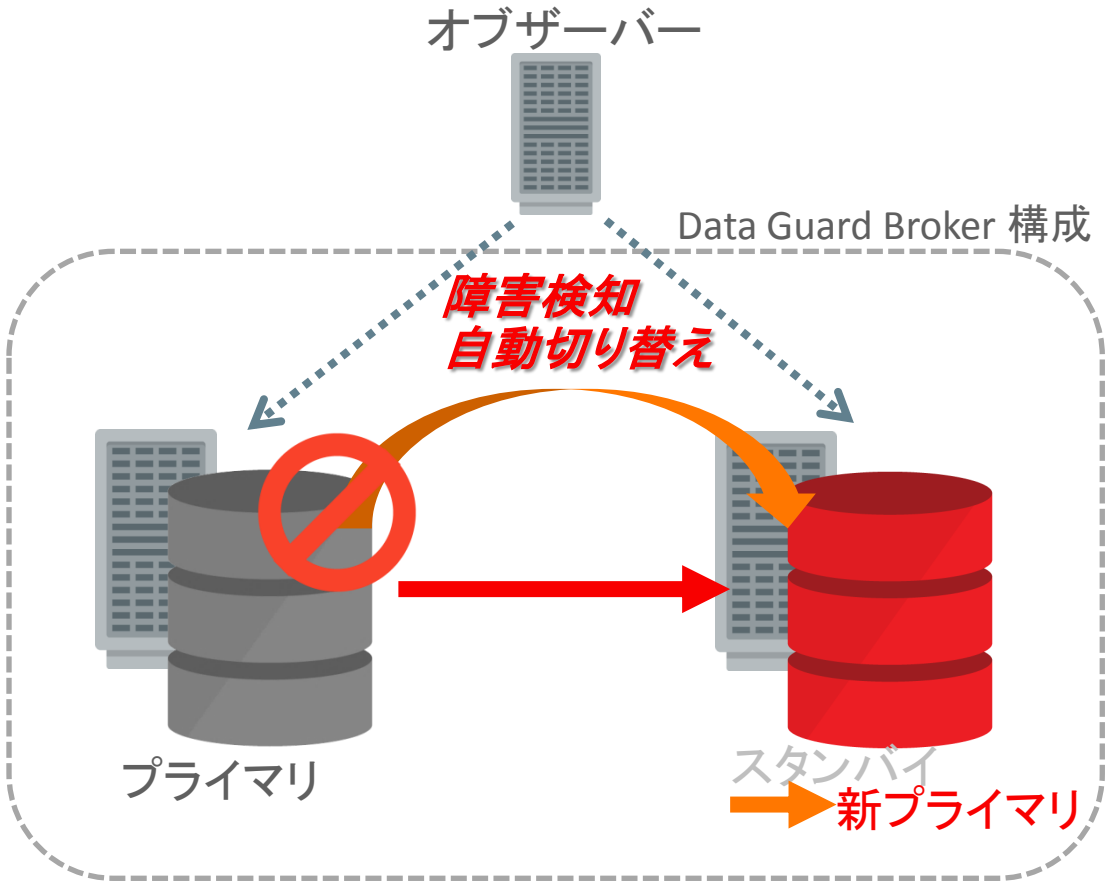
4)FS1復旧。STB1/STB2より高プライオリティのため、FS1へフェイルバック



Data Guard ファスト・スタート・フェイルオーバー

障害時に自動的にData Guard の切り替えを実行し、可用性を高め業務を継続

- Data Guard Broker 構成が有効になっている環境で、障害時に監視プロセス(オブザーバー)による自動切り替え機能
  - Data Guard Broker 構成 + オブザーバー
  - アプリケーションからのコールによる実行も可能
  - 旧プライマリの自動復旧
    - フェイルオーバー後に旧プライマリを自動で復旧(フラッシュ・バック)し、スタンバイとしてData Guard 構成に組み込む
    - フラッシュバック・データベースの有効必須



ファスト・スタート・フェイルオーバーの複数ターゲット・スタンバイ

NEW IN 12.2

複数スタンバイ構成全体で、ファスト・スタート・フェイルオーバーが設定

- 複数スタンバイが設定可能になり、構成全体でフェイル・オーバー可能なスタンバイに自動的に切り替えることが可能
  - 複数スタンバイ指定 or ANY を設定可能

```
DGMGRL> EDIT DATABASE <プライマリ> SET PROPERTY
        FastStartFailoverTarget='<STB>,<STB>…'|ANY;
```

  - 構成確認

```
DGMGRL> show database verbose orcl;
データベース - orcl
  ロール:       PRIMARY
  意図した状態   TRANSPORT-ON
  インスタンス:
    orcl
  プロパティ:
...
    FastStartFailoverTarget = 'std1,std2'
```

最大保護モードのファスト・スタート・フェイルオーバー

NEW IN 12.2

全保護モードが設定可能に

- 12.1まで

  ファスト・スタート・フェイルオーバーを設定する場合には、保護モードは最大可用性モードもしくは最大パフォーマンスモードが選択可能(最大保護モードは不可)

- 12.2から

  2つ以上の同期モードのスタンバイ・データベースがある場合には、最大保護モードも指定可能となり、全ての保護モードが選択可能

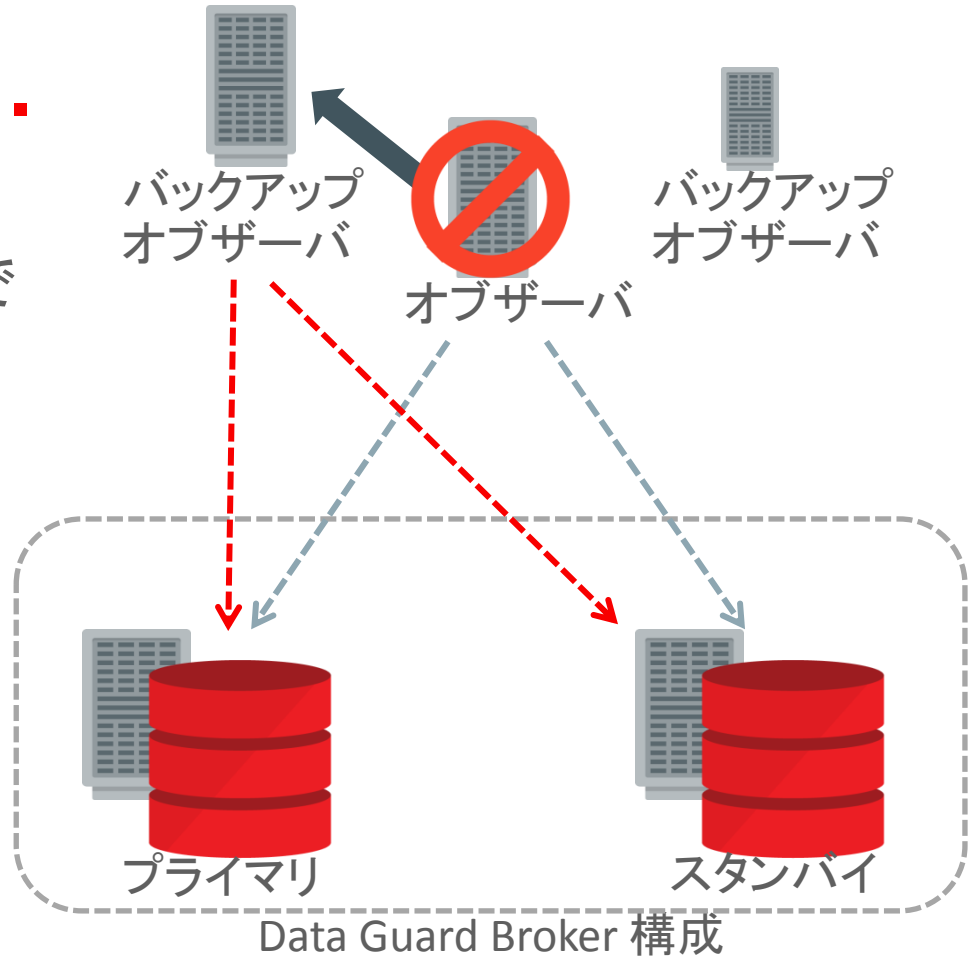
```
DGMGRL>  show configuration;
構成 - broker393394
  保護モード: MaxProtection
  メンバー:
  orcl - プライマリ・データベース
    std1 - フィジカル・スタンバイ・データベース
    std2 - フィジカル・スタンバイ・データベース
ファスト・スタート・フェイルオーバー: ENABLE
構成ステータス:
SUCCESS   (ステータスは12秒前に更新されました)
DGMGRL> show database verbose orcl;
データベース - orcl
  ロール:            PRIMARY
  意図した状態   TRANSPORT-ON
  インスタンス:
    orcl
  プロパティ:
...
    FastStartFailoverTarget = 'std1,std2'
...
```

ファスト・スタート・フェイルオーバー複数オブザーバー

NEW IN 12.2

オブザーバーの可用性向上

- オブザーバーを複数持つことにより、ファスト・スタート・フェイルオーバーを常に有効に
  - これまでも EMCC を使用することにより、HA構成での別サーバー上へ切り替えることは可能
  - 最大3つまで起動可能
    - 1つがマスター、残りがバックアップ
    - マスター停止/通信不可時に、バックアップが昇格
    - 各オブザーバーごとにログ出力
    - 同一サーバー上で複数起動は不可



ファスト・スタート・フェイルオーバー複数オブザーバー

NEW IN 12.2

オブザーバーの可用性向上

- 確認

```
DGMGRL>  show observer;
構成 - broker393394
  プライマリ:     orcl
  ターゲット:     std1

オブザーバ"xxxx101.oracle.com" - マスター
  ホスト名:                    xxxx101.oracle.com
  プライマリに対する最後のping: 2秒前
  ターゲットに対する最後のping: 3秒前

オブザーバ"xxxx102.oracle.com" - バックアップ
  ホスト名:                    xxxx102.oracle.com
  プライマリに対する最後のping: 0秒前
  ターゲットに対する最後のping: 3秒前

オブザーバ"xxxx103.oracle.com" - バックアップ
  ホスト名:                    xxxx103.oracle.com
  プライマリに対する最後のping: 0秒前
  ターゲットに対する最後のping: 3秒前
```

- オブザーバーの起動

```
DGMGRL> start observer in background connect
identifier is orcl1;
コマンド"START OBSERVER"が接続識別子"orcl1"を使用して発行
されました
```

- マスターの変更

```
DGMGRL> set masterobserver to xxxx102.oracle.com;
```

提案されたマスター・オブザーバがData Guard Broker構成に送信されました。マスター・オブザーバの切替えが実際に行われるかどうかを確認するには、SHOW OBSERVERを実行してください。

ORACLE

ファスト・スタート・フェイルオーバー複数オブザーバー

NEW IN 12.2

管理性の向上

- 1コマンドで複数オブザーバーの起動/停止が可能
- 構成ファイルを事前に用意
  - 認証のために Wallet を使用

```
BROKER_CONFIGS =(
  (CONFIG =(NAME=<構成名>)
             (CONNECT_ID=<接続詞>)(FILES=<構成ファイル>))
  (CONFIG =(NAME=<構成名>)
             (CONNECT_ID=<接続詞>)(FILES=<構成ファイル>))
CONFIG_GROUPS = (
   (GROUP =(NAME=<グループ名> )
            (CONFIG_LIST = (NAME=<構成名>)(NAME=<構成名>)))
```

ファスト・スタート・フェイルオーバー複数オブザーバー

NEW IN 12.2

1) 構成ファイルを用意

- CONNECT_ID は、プライマリ、スタンバイからも接続可能なものを指定

```
BROKER_CONFIGS =(
   (CONFIG =(NAME=SALES)(CONNECT_ID=SALES_P)(FILES=/home))
   (CONFIG =(NAME=HR)(CONNECT_ID=HR_P)(FILES=/home))
   (CONFIG =(NAME=CUSTOMER)(CONNECT_ID=CUSTOMER_P)(FILES=/home))
   (CONFIG =(NAME=ORDERS)(CONNECT_ID=ORDERS_P)(FILES=/home)))
CONFIG_GROUPS = (
   (GROUP =(NAME=GRP_A )(CONFIG_LIST = (NAME=SALES)(NAME=ORDERS)))
   (GROUP =(NAME=GRP_B)(CONFIG_LIST = (NAME=HR)(NAME=CUSTOMER))))
```

ORACLE

ファスト・スタート・フェイルオーバー複数オブザーバー

NEW IN 12.2

2) 構成ファイルを設定して、1コマンドで複数オブザーバーを起動

- 構成ファイルをセット

```
DGMGRL> set observerconfigfile='<構成ファイルのパス>';
```

```
DGMGRL> set observerconfigfile='/home/myobservers.ora';
DGMGRL> show observerconfigfile;
ObserverConfigFile=/home/myobservers.ora
```

- 構成ファイル内に設定したグループのオブザーバを起動

```
DGMGRL> start observing <構成ファイル内のグループ名> ;
```

```
DGMGRL> start observing grp_a ;
ObserverConfigFile=/home/myobservers.ora
Observer configuration file parsing succeeded
Submitted command "START OBSERVER" using connect identifier "SALES_P"
Submitted command "START OBSERVER" using connect identifier "HR_P"

Check superobserver.log and individual observer logs
```

リソースの有効活用

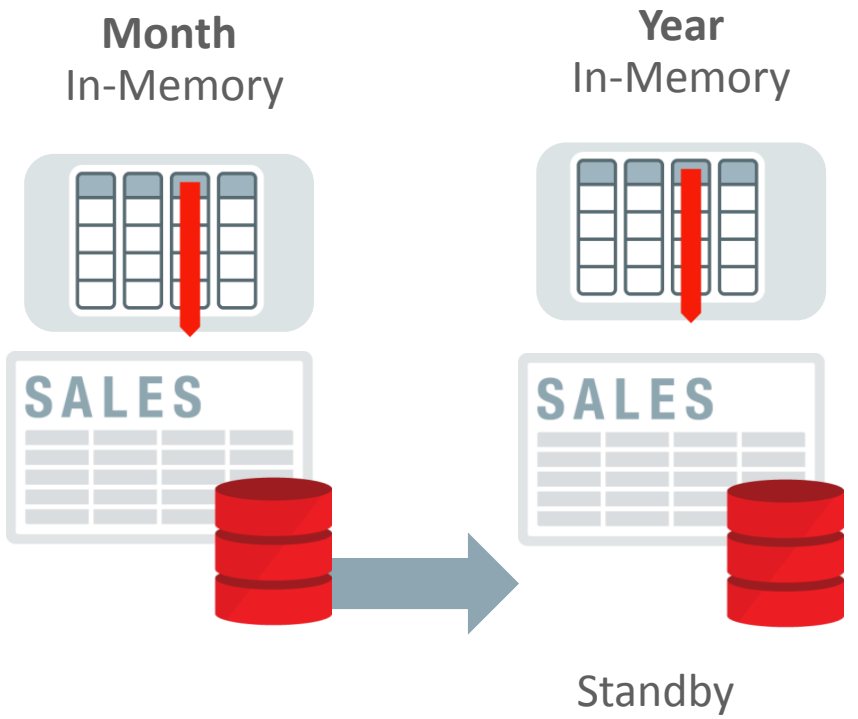
- ✓ Active Data GuardのDatabase In-Memory 対応
- ✓ 複数インスタンスでのREDO適用
- ✓ Active Data GuardのDiagnostic Packサポート

Database In-Memory 対応

ADG NEW IN 12.2

Active Data GuardでDatabase In-Memoryし、オフロードや用途ごとのリソース活用

- Database In-Memory をActive Data Guard 上で利用可能
  - スタンバイでIMC 表・カラムの作成
  - プライマリでポピュレート対象のデータと異なるデータをポピュレート可能



複数インスタンスでのREDO適用

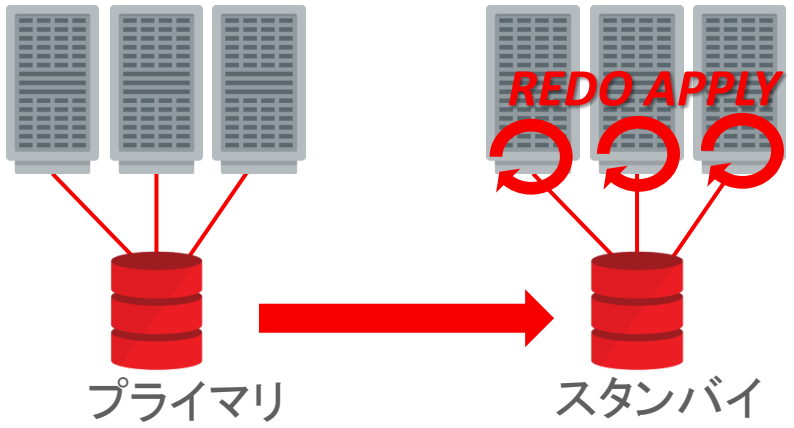
NEW IN 12.2

REDO適用性能の向上

- RACの複数インスタンスでREDO適用が可能
  - 大規模/REDO生成量が多い環境で、REDO適用性能の向上による適用ラグの短縮
    - 検証結果: 8ノードRACで、3,500MB/sec 適用性能

```
SQL> RECOVER MANAGED STANDBY DATABASE DISCONNECT USING INSTANCES [ALL|<N>];
```

    - n : 1～インスタンス数、ノードごとに1スレッド
      - ALL の場合は、全インスタンスで同じモード(mount/read-only)
      - Broker プロパティ値 : **ApplyInstances**
  - 下記を使用しているスタンバイ環境では不可
    - 高速増分バックアップ機能
    - Database In-Memory 機能
      - パラレルREDO 適用を使用する場合、ADG_IMC_ENABLED 値が全インスタンスで同じある必要あり



複数インスタンスでのREDO適用

設定例

NEW IN 12.2

- SQL*Plus

```
RECOVER MANAGED STANDBY DATABASE DISCONNECT USING INSTANCES ALL;
```

```
RECOVER MANAGED STANDBY DATABASE DISCONNECT USING INSTANCES 2;
```

- DGMGRL (Broker)

```
DGMGRL> edit database 'stb' set property applyinstances=ALL;
Property "applyinstances" updated
```

```
DGMGRL> show database 'stb' applyinstances;
ApplyInstances = 'all'
```

複数インスタンスでのREDO適用

確認方法/注意

NEW IN 12.2

- パラレル適用が有効になっているかの確認
  - PRnnプロセスを確認
  - Brokerの**ApplyInstances**プロパティ値
- MRP は1ノードでのみ起動
  - V$DATAGUARD_PROCESSの結果で、MRP起動インスタンスでのみ、MRPのエントリが出力
  - DGMGRL で確認した際に、apply instanceは1つ(MRP起動ノード)のみ

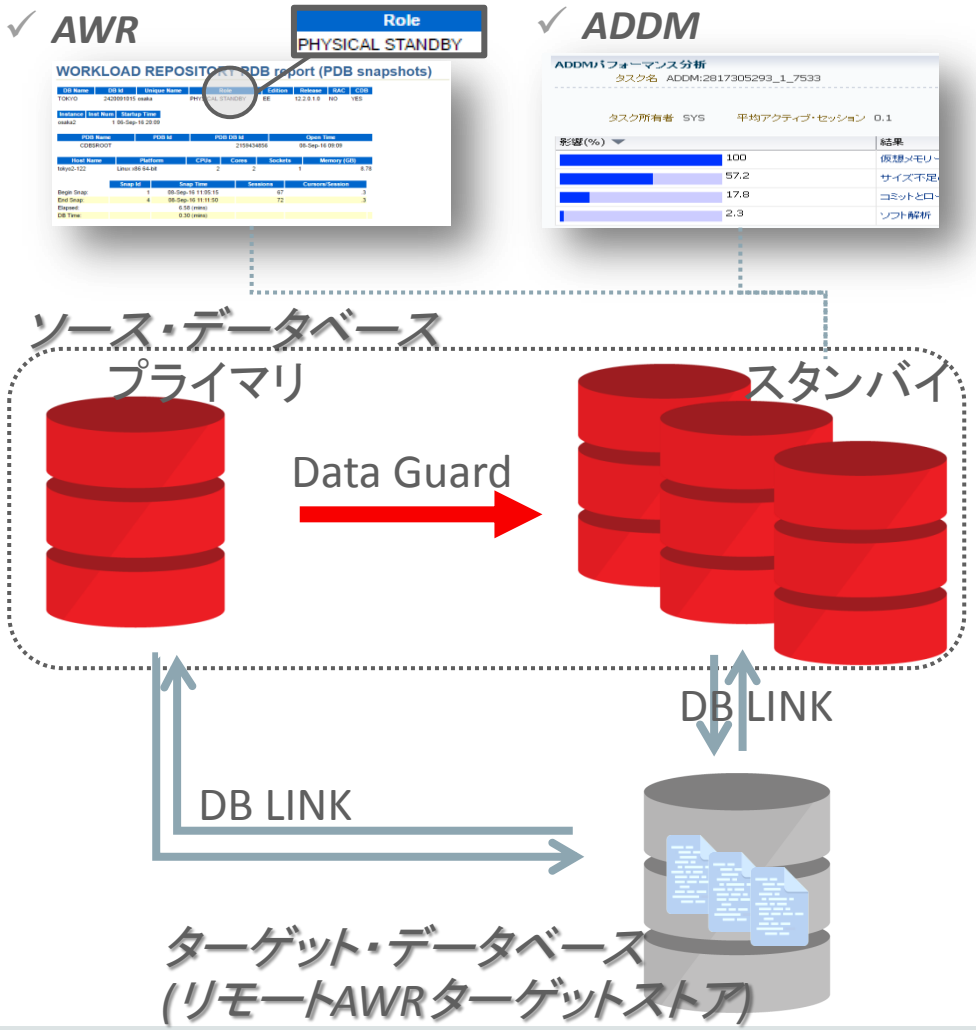
```
DGMGRL> show database 'stb'

Database - stb

  Role:                PHYSICAL STANDBY
  Intended State:      APPLY-ON
  Transport Lag:       0 seconds (computed 1 second ago)
  Apply Lag:           0 seconds (computed 1 second ago)
  Average Apply Rate:  5.00 KByte/s
  Real Time Query:     ON
  Instance(s):
    stb1
    stb2 (apply instance)

Database Status:
SUCCESS
```

Active Data Guard のDiagnostic Pack のサポート

ADG NEW IN 12.2

AWR リモート・スナップショット

- スタンバイのワークロードは、プライマリと異なる
  - 12.1 まで：スタンバイstatspackなどで分析
- スタンバイのスナップショットの取得・レポート作成が可能に
  - AWRレポートの作成、Import/Exportが可能
  - AWR データを使用した、自動データベース診断モニター(ADDM) が可能
  - SQL Tuning Advisor も利用可能

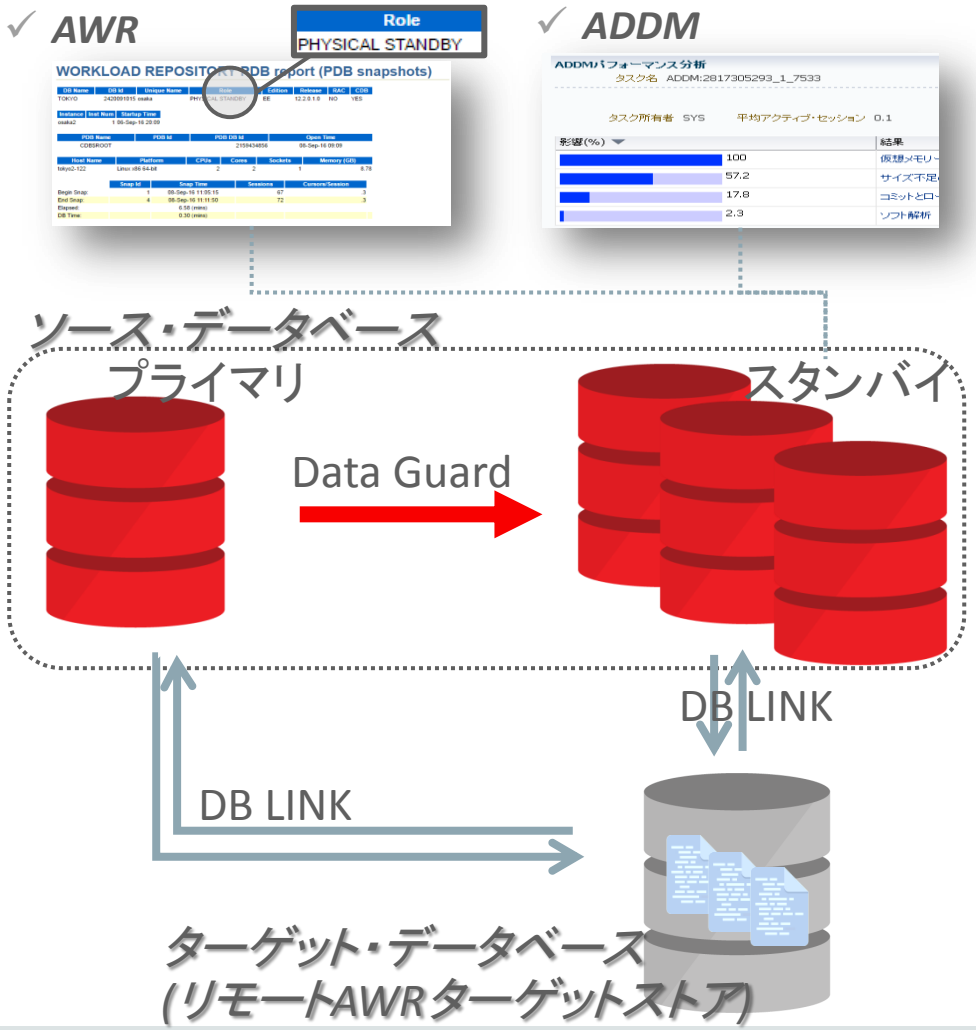


Active Data Guard のDiagnostic Pack のサポート

ADG NEW IN 12.2

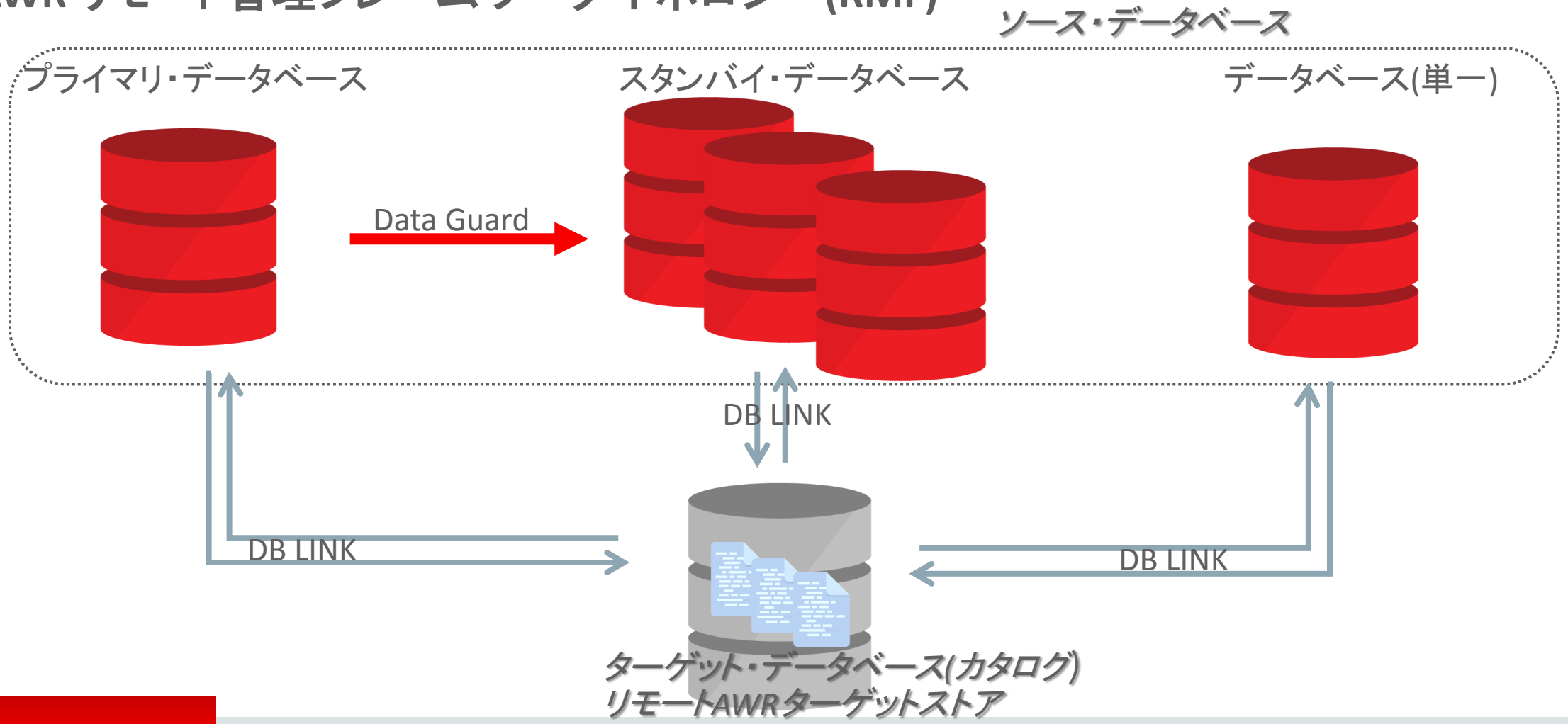
AWR リモート・スナップショット

- リモート・スナップショットを書き込み可能なターゲット・データベースに格納
  - 情報取得元:ソース、格納先:ターゲット(R/W)
    - プライマリはソースorターゲットどちらも可
    - スタンバイはソースのみ
    - ソース/ターゲットともに非Data Guard 環境可
  - スナップショット取得対象のソースから、格納先のターゲットにDB LINK 経由で情報を送信
    - スナップショット取得はターゲットから実行
    - 取得は、定期実行(スケジューリング)もしくは手動実行



AWR リモート・スナップショット

AWR リモート管理フレームワーク・トポロジー(RMF)



AWR リモート・スナップショット

導入することによるメリット

ADG NEW IN 12.2

- スタンバイをソースにした場合
  - スタンバイのAWRが作成可能になり、分析が容易に
    - これまでスタンバイの分析は、スタンバイ Statspackを使用
- プライマリや単一データーベースをソースにした場合
  - 本番データベース内に蓄積していたAWRデータを外出し可能
    - スナップショットの蓄積によるデータファイル(SYSAUX)の肥大化の防止
    - 分析作業を本番環境外で行える
  - 複数データベースの情報を一ヶ所に集約化
- プライマリをターゲットにした場合
  - スタンバイにも情報が反映されるため、スタンバイからAWRレポートが作成可能

AWR リモート・スナップショット

設定方法

ADG NEW IN 12.2

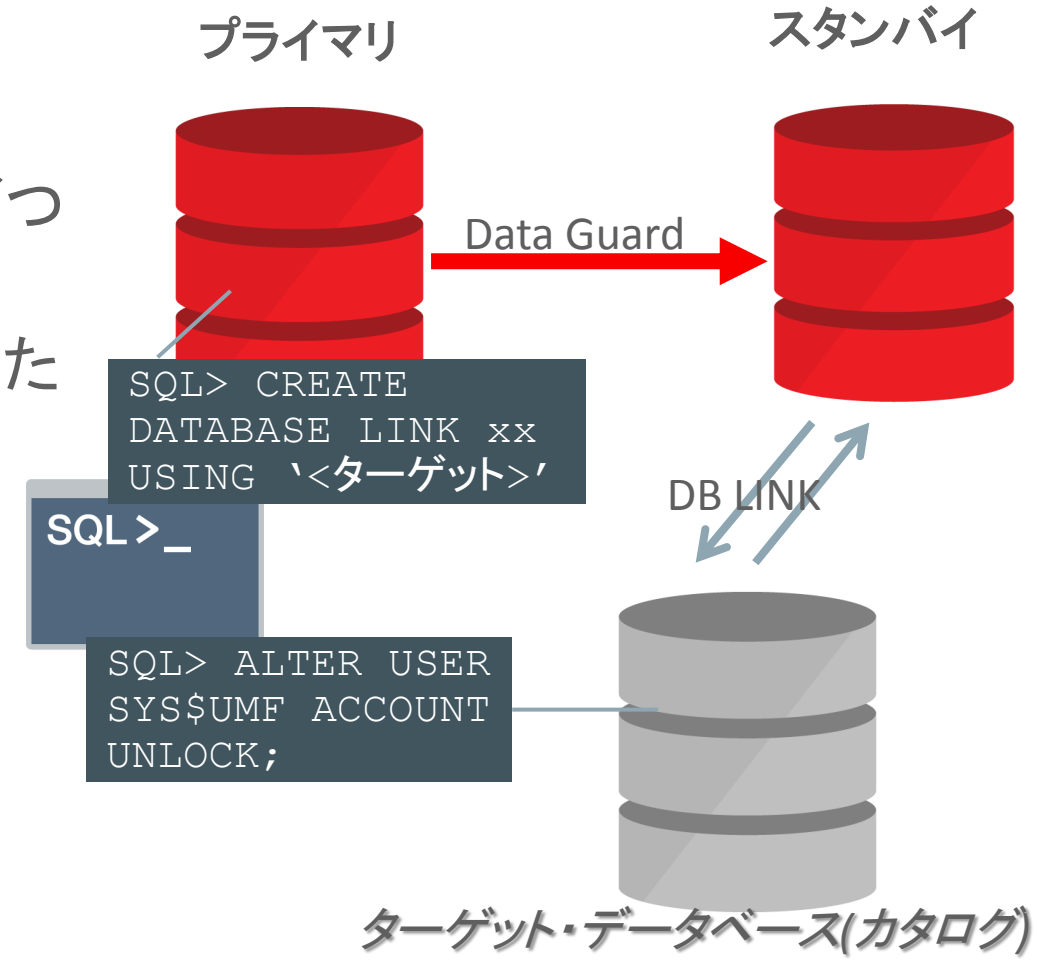
#	実行データベース	実行内容
1	ソース/ターゲット	事前準備
2	ソース/ターゲット	ソースとターゲットを定義
3	ターゲット	RMF トポロジーの作成
4	ターゲット	RFM トポロジーにソースを登録
5	ターゲット	(オプション)ソースとターゲット候補間のDBLINK作成
6	ターゲット	ソースから情報を提供するためのAWRサービスを登録

AWR リモート・スナップショット

1) 事前準備

ADG NEW IN 12.2

- ソースとターゲット間のDBLINK作成
  - 双方向に必要なため、各ターゲットごとに2つずつ必要になる
  - スタンバイからのDBLINKは、DDL実行は不可のため、プライマリで実行してREDO適用により反映
- SYS$UMF ユーザーのアンロック
  - デフォルトユーザー
  - RMF関連ビューや表のアクセス権限あり



AWR リモート・スナップショット

2)ソースとターゲットを定義

ADG NEW IN 12.2

- ソースとターゲットでDBMS_UMF.configure_node() プロシージャーを実行
  - DBMS_UMF.configure_node('<ソースorターゲット名>')
  - 値を指定しない場合は、DB_UNIQUE_NAME値が使用される

```
SQL> show user
USER is "SYS$UMF"

SQL> select DB_UNIQUE_NAME,DATABASE_ROLE from v$database;
DB_UNIQUE_NAME                 DATABASE_ROLE
------------------------------ ----------------
target                         PRIMARY

SQL> exec DBMS_UMF.configure_node('target');

PL/SQL procedure successfully completed.
```

ORACLE

AWR リモート・スナップショット

3) RMFトポロジーの作成

ADG NEW IN 12.2

- ターゲットでDBMS_UMF.create_topology() プロシージャーを実行
  - DBMS_UMF.create_topology (‘<トポロジー名>’)
  - 大文字と小文字は区別される

```
SQL>  exec DBMS_UMF.create_topology ('Topology_to');

PL/SQL procedure successfully completed.

SQL> select * from DBA_UMF_TOPOLOGY;

TOPOLOGY_NAME          TARGET_ID TOPOLOGY_VERSION TOPOLOGY
-------------------- ---------- ---------------- --------
Topology_to          2620454915                1 ACTIVE
```

AWR リモート・スナップショット

4) RMF トポロジーにソースを登録

ADG NEW IN 12.2

- ターゲットでDBMS_UMF.register_node () プロシージャーを実行
  - DBMS_UMF.register_node('<トポロジー名>','<ソース名>','<ソースへのDBLINK>', '<ソースからのDBLINK>', '<ソースかどうか>', '<ターゲット候補かどうか>')

```
SQL> exec DBMS_UMF.register_node(TOPOLOGY_NAME=>'Topology_to', NODE_NAME=>'tokyo',
DBLINK_TO_NODE=>'DBLINK_T_TO_SP', DBLINK_FROM_NODE=>'DBLINK_SP_TO_T', AS_SOURCE=>'TRUE',
AS_CANDIDATE_TARGET=>'TRUE');

PL/SQL procedure successfully completed.
SQL> select * from DBA_UMF_REGISTRATION;

TOPOLOGY_NAME        NODE_NAME              NODE_ID  NODE_TYPE AS_SO AS_CA STATE
-------------------- -------------------- ---------- ---------- ----- ----- ------------
Topology_to          tokyo                1496711999          0 TRUE  TRUE  OK
Topology_to          target               2620454915          0 FALSE FALSE OK
Topology_to          osaka                2420091015          0 TRUE  FALSE OK
```

AWR リモート・スナップショット

5) (オプション) ソースとターゲット候補間のDBLINK作成

ADG NEW IN 12.2

- ロール変換や障害時に、ソースやターゲットが切り替わることを想定して、可能性があるソース/ターゲット候補間でのDBLINKを作成
- ターゲットでDBMS_UMF.create_link () プロシージャーを実行
  - DBMS_UMF.create_link (‘<トポロジー名>’,’<候補1>’, ’<候補2>’, ’<候補1から候補2へのDBLINK名>’ ,’<候補2から候補1へのDBLINK名>’);

```
SQL> exec DBMS_UMF.create_link ('Topology_to', 'tokyo', 'osaka','DBLINK_SP_TO_SS',
'DBLINK_SS_TO_SP');

PL/SQL procedure successfully completed.
```

AWR リモート・スナップショット

6)ソースから情報を提供するためのAWRサービスを登録

ADG NEW IN 12.2

- ターゲットでDBMS_WORKLOAD_REPOSITORY.register_remote_databaseプロシージャーを実行
  - DBMS_WORKLOAD_REPOSITORY.register_remote_database('<ソース名>')

```
SQL> exec DBMS_WORKLOAD_REPOSITORY.register_remote_database(node_name=>'tokyo');

PL/SQL procedure successfully completed.

SQL> select * from DBA_UMF_SERVICE;

TOPOLOGY_NAME           NODE_ID SERVICE_ID
-------------------- ---------- --------------------
Topology_to          2420091015 AWR
Topology_to          1496711999 AWR
```

計画/計画停止時の切り替え

- ✓ ローリング・アップグレードとの連携強化
- ✓ 実行中の処理を捌いた後のスイッチオーバー
- ✓ Active Data Guard の接続済コネクションの継続

ローリング・アップグレードとの連携強化

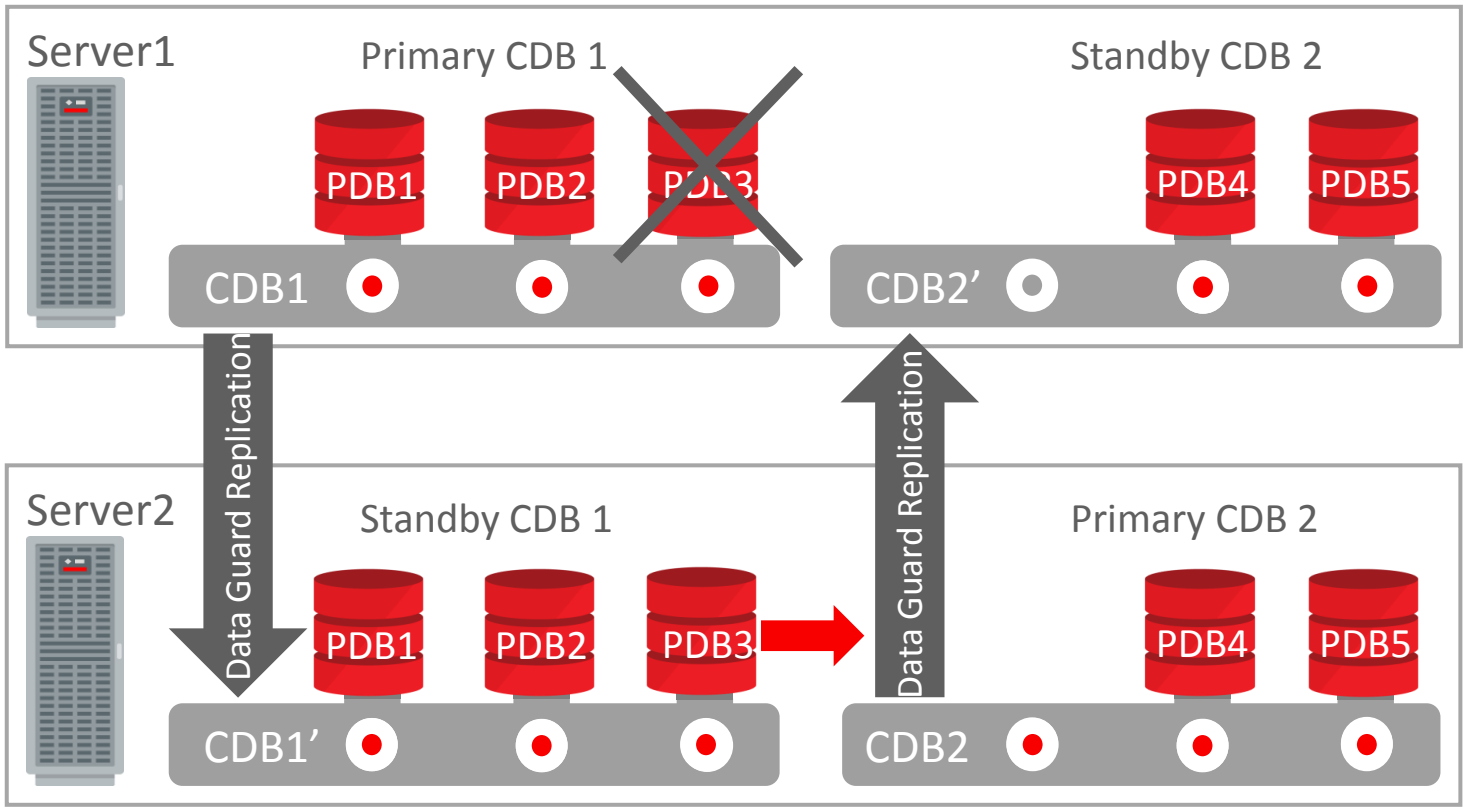
NEW IN 12.2

- ローリング・アップグレードの Enterprise Manager サポート
- DBMS_ROLLING アップグレード・プロセス内の新しいサポート
- Active Data Guard のローリング・アップグレードに対する Data Guard Broker サポート

Data Guard Broker によるPDBの移行/フェイルオーバー

Data Guard 環境でのPDB移動が簡単に

- Data Guard Broker で管理している Data Guard 環境間で、同一サーバー上のPDB移動が可能
  - フェイルオーバー(右図)
    - スタンバイCDBのPDBを別の構成のプライマリCDBにフェイルオーバー
  - 移行
    - プライマリCDBのPDBを別の構成のプライマリCDBに移行



Data Guard Broker によるPDBの移行/フェイルオーバー

実行例

- ソース側で実行

```
DGMGRL> MIGRATE PLUGGABLE DATABASE <PDB名> TO
        CONTAINER <ターゲットCDB名> USING <xxxx.xml>
        CONNECT AS <ターゲットCDBユーザー>/<パスワード>@<ターゲット接続記述子>;
```

- 例

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB3 TO
        CONTAINER CDB2 USING PDB3.xml
        CONNECT AS sys/mypassword@CDB2;
```

Data Guard Broker によるPDBの移行/フェイルオーバー

前提・制限

- ターゲットのCDBは、ソースのCDBと同一サーバーである必要がある
  - リモート(別サーバー)PDBリロケーションはData Guard Brokerではなく、relocateコマンドで実行可能
- ソースとターゲットのCDBで、同一の共有ストレージを使用
  - データファイルのコピーは、Data Guar Broker での操作のプロセス内ではない
- ターゲットCDBにplugする際、STANDBYS=NONE で追加される
  - ターゲット側の構成のスタンバイには、手動で設定する必要がある
- 移行対象のPDBは、ソースCDBから削除される

実行中の処理を捌いた後のスイッチオーバー
スイッチオーバー時の手順の簡略化

NEW IN 12.2

- スイッチ・オーバー時、実行中の処理を完了させた後に切り替える
  - 11.2～ srvctl コマンドでサービス停止/開始をすることで対応可能
  - 12.2 Broker の1コマンドで可能に

	12.1 までの手順	12.2 での手順
プライマリでサービス停止	`srvctl stop service`	`DGMGRL> switchover to <standby> wait`
スイッチ・オーバー実行	`DGMGRL> switchover to <standby>`	
新プライマリでサービス開始	`srvctl start service`	

  - CRSリソースのサービスの設定で、どれだけ待機するか設定可能
    - drain_timeout

Active Data Guard の接続済コネクションの継続

ロール変換時にコネクションを残すことが可能に

ADG NEW IN 12.2

- スイッチ・オーバーやフェイル・オーバー時のスタンバイがプライマリに変換される際に、スタンバイに接続済のコネクションを残すことが可能
  - Active Data Guard のスタンバイは、Read-Onlyの処理が可能
    - これまでは、ロール変換時に接続済コネクションは切断され、再接続が必要
  - 接続済のコネクションを残すことにより再接続不要
    - ロール変換後の再接続に要する手順/時間の削減
  - STANDBY_DB_PRESERVE_STATES パラメータ
    - スタンバイ・ロールの時にのみ有効
    - 設定値: NONE(デフォルト) | ALL | SESSION

ORACLE

その他

- ✓ 12.2 で導入されたパラメータ
- ✓ V$DATAGUARD_PROCESS ビュー

12.2 で導入されたパラメータ

NEW IN 12.2

Active Data Guard 関連のパラメータ

パラーメータ名	値(*デフォルト値*)	詳細
ADG_IMC_ENABLED	true | false (*true*)	Active Data Guard のスタンバイ側でDBIM の有効化
DATA_GUARD_SYNC_LATENCY	<数値>(*0*)	同期(SYNC)モードでREDO 転送を設定している場合に、LGWR の最初の書き込みをどれくらい待つかを設定 設定可能な数値は、0～LOG_ARCHIVE_DEST_n のNET_TIMEOUT 値
DATA_TRANSFER_CACHE_SIZE	<数値> K|M|G (※デフォルト値は、詳細列を参照)	NOLOGGING ブロックのリカバリ(RMAN のRECOVER ….NOLOGGIED BLOCK コマンド) でのREDO転送のキャッシュサイズ。 SGA_TARGET が設定されていない場合は無効。SGA_TARGET が設定されている場合は、デフォルト0
ENABLED_PDBS_ON_STANDBY	<PDB名>(*)	Data Guard スタンバイ側に同期する対象のPDB名リスト 例) “*”,”PDB*A”,”-PDB1*A” …
STANDBY_DB_PRESERVE_STATES	NONE | SESSION | ALL (*NONE*)	Active Data Guard のスタンバイをプライマリにロール変換する際に、接続済コネクションを残すかを設定 preserving user sessions when a standby is converted to primaryと関係

V$DATAGUARD_PROCESS ビュー

スタンバイ関連のプロセスを確認するビューの変更

NEW IN 12.2

- V$MANAGED_STANDBY は12.2では非推奨になり、V$DATAGUARD_PROCESSを使用

```
SQL> SELECT CLIENT_PID, NAME, THREAD#, SEQUENCE#, ROLE,INSTANCE, ACTION
     FROM V$DATAGUARD_PROCESS;
CLIENT_PID NAME     THREAD#  SEQUENCE# ROLE                      INSTANCE ACTION
---------- ----- ---------- ---------- ---------------------- ---------- ------------
     19153 rfs             1          6 RFS async                       0 IDLE
     12640 rfs             0          0 RFS archive                     0 IDLE
     12630 rfs             1          6 RFS ping                        0 IDLE
         0 ARC1            0          0 archive redo                    0 IDLE
         0 ARC0            0          0 archive local                   0 IDLE
         0 TT01            0          0 redo transport timer            0 IDLE
         0 TT00            0          0 gap manager                     0 IDLE
         0 MRP0            0          0 managed recovery                0 IDLE
         0 ARC3            0          0 archive redo                    0 IDLE
         0 ARC2            0          0 archive redo                    0 IDLE
         0 TMON            0          0 redo transport monitor          0 IDLE
         0 LGWR            0          0 log writer                      0 IDLE
```

Active Data Guard/Data Guard 12.2 まとめ

管理性

- ✓ パスワードファイルの自動同期
- ✓ PDB サブセット・スタンバイ
- ✓ スタンバイ・データベース/Far Sync作成
- ✓ NOLOGGING 処理に対する検証やリカバリ
- ✓ Data Guard Broker の操作性の向上
- ✓ REST API 対応

リソース有効活用

- ✓ ADG のDatabase In-Memory 対応
- ✓ 複数インスタンスでのパラレルREDO適用
- ✓ AWR リモート・スナップショット
- ✓ SQL Tuning Advisor サポート

データ保護

- ✓ 複数スタンバイ環境での同期転送の向上
- ✓ プライマリとスタンバイの比較によるプロアクティブな検証
- ✓ REDO転送先の自動切り替えの強化
- ✓ ファスト・スタート・フェイルオーバー複数ターゲット・スタンバイ
- ✓ 最大保護モードでのファスト・スタート・フェイルオーバー
- ✓ ファスト・スタート・フェイルオーバーの複数オブザーバー

計画/計画外停止の切り替え

- ✓ ローリング・アップグレードとの連携強化
- ✓ 実行中の処理を捌いた後のスイッチオーバー
- ✓ Active Data Guard の接続済コネクションの継続

Active Data Guard/Data Guard 12.2 まとめ

管理性

Data Guard 環境の
管理や操作がより簡単

リソース有効活用

スタンバイで可能な機能を強化し、
リソース有効活用でROIを高める

データ保護

データ保護や自動化を強化し、
DR用途としての業務継続性を強化

計画/計画外停止の切り替え

計画/計画外停止時の
手順の簡易化で
切り替え時間の短縮

リファレンス
マニュアル・ドキュメント

- Data Guard 概要および管理 12*c* リリース 2 (12.2)
  - 全般

    http://docs.oracle.com/cd/E82638_01/SBYDB/toc.htm
- Data Guard Broker, 12*c* Release 2 (12.2)
  - 全般(英語)

    http://docs.oracle.com/database/122/DGBKR/toc.htm
- Database 高可用性概要, 12c リリース 2 (12.2)
  - 全般

    http://docs.oracle.com/cd/E82638_01/HAOVW/toc.htm

Recovery Manager 12.2新機能

NEW IN 12.2

RMAN 12.2新機能一覧

1. RMANコマンド機能拡張
2. RMAN表単位リカバリ機能拡張
   - 表単位リカバリ中のディスクスペースチェック
   - REMAP SCHEMAを用いたスキーマを跨る表単位リカバリ
3. XTTSを用いたクロス・プラットフォームでのPDB移行
4. ネットワークを介したクロス・プラットフォーム・リカバリ

ORACLE

NEW IN 12.2

1) RMANコマンド機能拡張

REDOを失ったときのバックアップからのデータベース・リカバリ

- 12.1まで
  - recover databaseコマンドは最新のログが無いことによりエラーを返す
  - 最後の有効なアーカイブREDOを特定して、UNTIL CANCEL句を付けてリカバリをする必要がある

```
SQL> recover database until cancel;
CANCEL
SQL> alter database open resetlogs;
```

- 12.2から
  - UNTIL AVAILABLE REDO句を付けることで最後の利用可能なREDOを自動的に見つけてリカバリ可能
  - CDB Rootでのみ実施可能

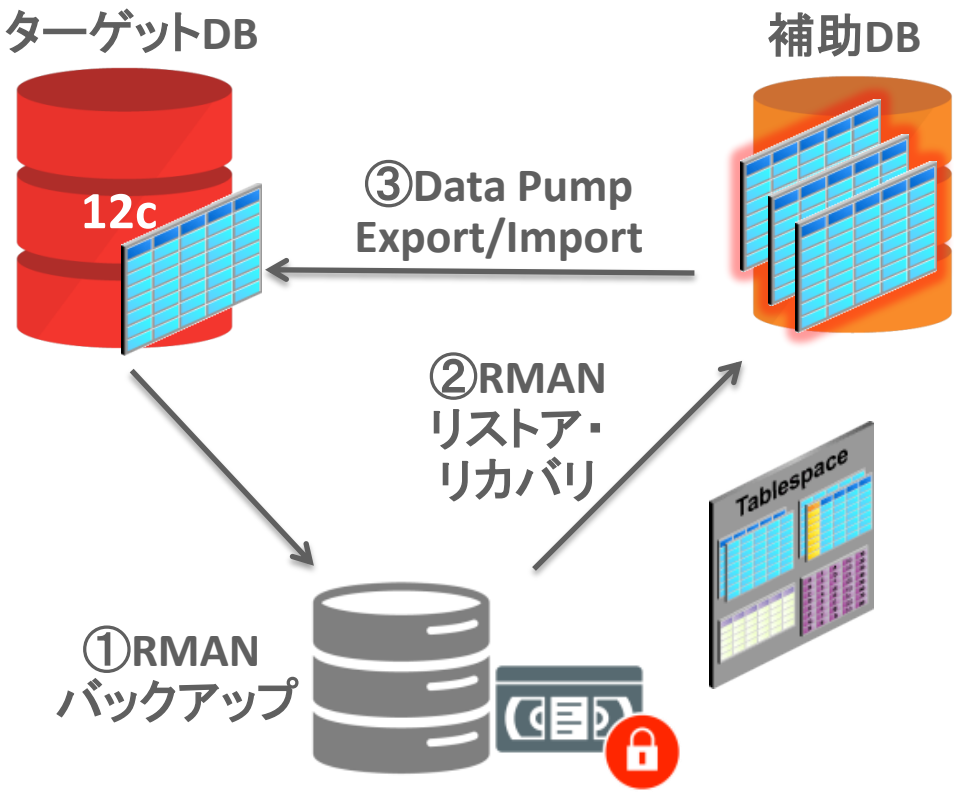
```
RMAN> recover database until available redo;
RMAN> alter database open resetlogs;
```

2) RMAN表単位リカバリ機能

表単位リカバリとは(12.1からの機能)

- 表を指定した時間までリカバリ可能
- 表のDROP/TRUNCATE/定義変更時も使用可能
- RMANバックアップを補助DBにリストアし、そこから必要なデータを抽出してリカバリ
- データだけをData Pumpのダンプファイル形式で出力することも可能

- 表名・表領域名を変更してリカバリ可能(REMAP)
  - 表Aを1日前の状態にリカバリし、表Bとして作成
  - スキーマを跨ったREMAPは不可(12.1まで)



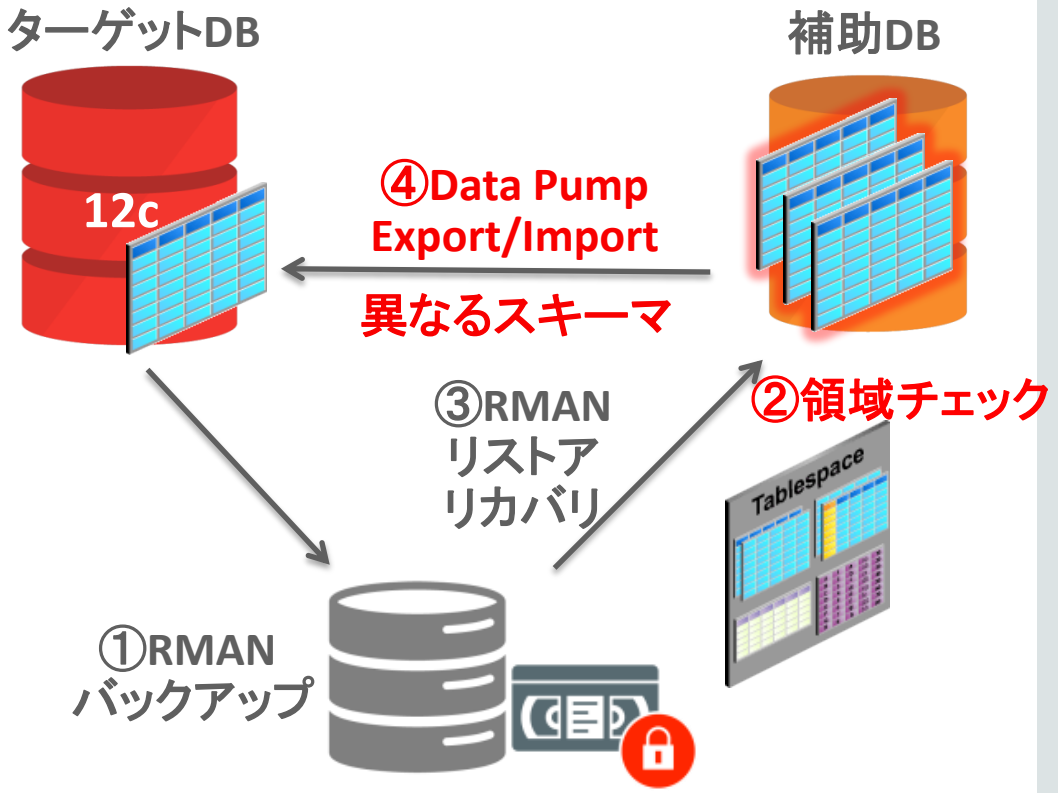
NEW IN 12.2

2) RMAN表単位リカバリ機能

12.2新機能

- 補助インスタンスのディスクスペースチェック
  - 表リカバリはSYSTEM, SYSAUX, UNDOとユーザ表領域をリストアするための領域を必要とする
  - リカバリ途中で失敗することを防ぐため補助インスタンス上のディスクスペースのチェックを**「事前に」**行う
- スキーマを跨ったリカバリ
  - 表レベルリカバリを異なるスキーマ下で実施可能
  - REMAP TABLEの後に<OLD>:<NEW>スキーマ名を指定

```
RECOVER TABLE hr.dept, sales.prod
UNTIL SCN 1234 AUXILIARY DESTINATION
'/tmp/' REMAP TABLE hr.dept:dev.testdept,
sales.prod:mkt.newprod;
```



NEW IN 12.2

2) RMAN表単位リカバリ機能

補助インスタンスのディスクスペースチェック

- RMANの表単位リカバリの機能は暗黙的に補助インスタンスを作り、バックアップをリストアする。十分なスペースがないと、OSレベルでエラーが返る。

実行例(12.1)

```
ORA-19502: write error on file
"/tmp/oracle/recover/ORCL/datafile/o1_mf_system_cvzr4on7_.dbf", block number
11264 (block size=8192)
ORA-27072: File I/O error
Linux-x86_64 Error: 28: No space left on device
```

- 12.2では機能拡張によって、補助インスタンスの作成前に利用可能なディスクスペースのチェックを行うようになった

実行例(12.2)

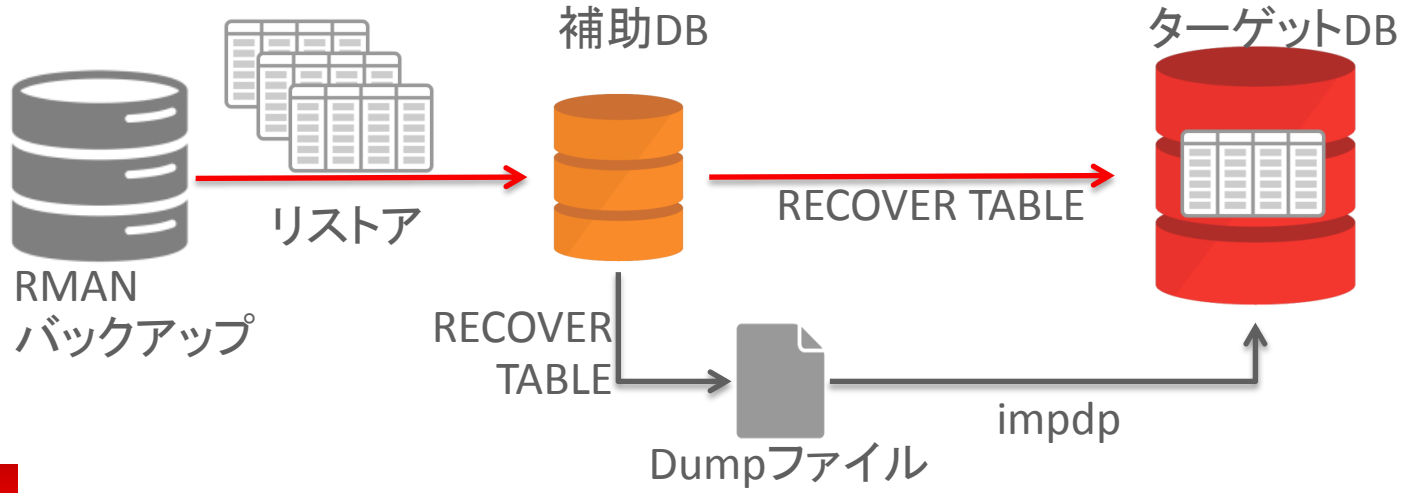
```
RMAN-05122: there is insufficient disk space to perform this table recovery
```

NEW IN 12.2

2) RMAN表単位リカバリ機能

スキーマを跨ったリカバリ

- 表単位リカバリでスキーマを跨った REMAP TABLE/TABLESPACE指定が可能になった
- これまではスキーマを跨ったREMAP TABLE/TABLESPACEを行うには、表単位リカバリで特定表のexpdpのDumpファイルを生成して、そのDumpファイルを用いて別スキーマにImpdpを手動実行する必要があった
- 12.2ではRMANコマンド1つで、スキーマを跨った表単位リカバリが可能



NEW IN 12.2

2) RMAN表単位リカバリ機能

スキーマを跨ったリカバリ

	12.1以前	12.2以降
1	一旦ダンプファイルにデータを抽出	RMANのみでスキーマ跨ったリカバリ可能

```
RMAN> recover table hr.dept of
pluggable database pdb1
  2   until scn 1560409
  3   auxiliary destination '/tmp/'
  4   datapump destination '/tmp/'
  5   dump file 'dept.dat'
  6   notableimport;
```

```
RMAN> recover table hr.dept of
pluggable database pdb1
  2   until scn 1560409
  3   remap table hr.dept:dev.testdept
  4   auxiliary destination '/tmp/';
```

2 スキーマを跨ってインポート

```
$ impdp system/Oracle12c@pdb1
DIRECTORY=pdb1_dumpdir
DUMPFILE=dept.dat TABLES=hr.dept
REMAP_TABLE=hr.dept:dev.testdept
```

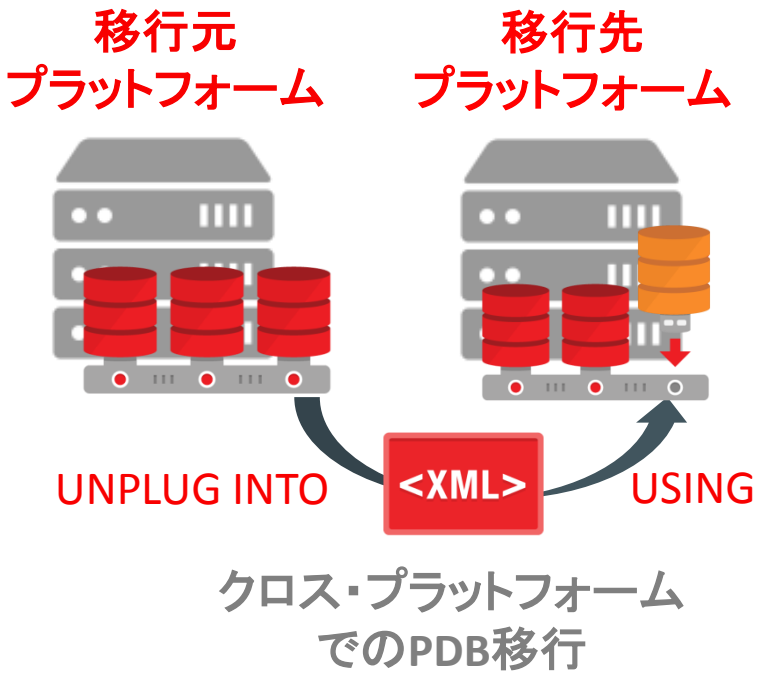
12.2 からは1つのステップで完了する

NEW IN 12.2

3) XTTSを用いたクロス・プラットフォームでのPDB移行

リモートの異なるプラットフォームのCDBにPDBをPlugin

- 同一エンディアンのクロス・プラットフォームでPDBの移行をサポート
- UNPLUG INTO <XML>句を付けてソースPDBをRMANバックアップの取得とともにUnplug
  - UNPLUGプロセスの中でXMLファイルが生成される
  - PDBメタデータ、エンディアン、SCNの情報が含まれる
- USING <XML>句を付けてPDBをクロス・プラットフォームのCDBにリストア・リカバリプロセスの中でPluginできる
- 全体の手順は通常のクロス・プラットフォーム移行と同様



NEW IN 12.2

3) XTTSを用いたクロス・プラットフォームでのPDB移行

RMANバックアップを用いたクロス・プラットフォームでの移行方法一覧

	データベース単位	PDB単位 (New in 12.2)	表領域単位(XTTS)
バックアップ取得単位	データベースの一括バックアップ	PDB毎のバックアップ	表領域毎のバックアップ
エンディアン変換	**不可**	**不可**	可
非一貫性での取得	不可	**可**	**可**
READ ONLY の時間	データベースのバックアップ取得中	**最後の差分増分バックアップ取得中**	**最後の差分増分バックアップ取得中**
ステップ数	少ない	多い	多い
圧縮	圧縮バックアップ可 未使用ブロックの圧縮可	圧縮バックアップ可 未使用ブロックの圧縮可	圧縮バックアップ可 未使用ブロックの圧縮可

ORACLE

NEW IN 12.2

3) XTTSを用いたクロス・プラットフォームでのPDB移行

実行例(一貫性バックアップを使用したLinux → Windowsの移行)

#	移行元CDB	移行先CDB
1	ソースPDBの停止 `ALTER PLUGGABLE DATABASE <PDB名> CLOSE;`	
2	Backup取得 `BACKUP FOR TRANSPORT UNPLUG INTO '/tmp/pdb.xml' PLUGGABLE DATABASE <PDB名> FORMAT '/tmp/%u';`	
3	移行先環境へのファイル転送	
4		Backupのリストア + Plug-in `RESTORE USING 'D:¥backup¥pdb.xml' FILE_NAME_CONVERT=('D:¥oradata','/oradata') FOREIGN PLUGGABLE DATABASE <PDB名> FORMAT 'D:¥oradata¥%U' FROM BACKUPSET 'D:¥backup¥<backupset名>';`

NEW IN 12.2

3) XTTSを用いたクロス・プラットフォームでのPDB移行

実行例(非一貫性バックアップを使用したLinux → Windowsの移行)

#	移行元CDB	移行先CDB
1	Level 0 Backupの取得(Online) `BACKUP INCREMENTAL LEVEL 0 FOR TRANSPORT ALLOW INCONSISTENT PLUGGABLE DATABASE <PDB名> FORMAT '/tmp/%u';`	2 Level 0 Backupのリストア `RESTORE FOREIGN PLUGGABLE DATABASE <PDB名> FORMAT 'D:¥oradata¥%U' FROM BACKUPSET 'D:¥backup¥<backupset名>';`
3	増分Backup取得 (停止タイミングまで繰返す) `BACKUP INCREMENTAL FROM SCN xxx FOR TRANSPORT ALLOW INCONSISTENT PLUGGABLE DATABASE <PDB名> FORMAT '/tmp/%u';`	5 増分Backupによるリカバリ + Plug-in `RECOVER USING 'D:¥backup¥pdb.xml' FOREIGN DATAFILECOPY … FROM BACKUPSET 'D:¥backup¥<backupset名>';`
4	PDB停止後、最後の増分Backup取得 + Unplug `BACKUP INCREMENTAL FROM SCN xxx FOR TRANSPORT UNPLUG INTO '/tmp/pdb.xml' PLUGGABLE DATABASE <PDB名> FORMAT '/tmp/%u';`	6 最後の増分Backupを使ったリカバリ + Plug-in `RECOVER USING 'D:¥backup¥pdb.xml' FOREIGN DATAFILECOPY … FROM BACKUPSET 'D:¥backup¥<backupset名>';`

NEW IN 12.2

3) XTTSを用いたクロス・プラットフォームでのPDB移行

注意点

- FOR TRANSPORT句 :
  バックアップ・セットを使用してクロス・プラットフォーム・バックアップを作成します。このバックアップ・セットは、制御ファイルでカタログ化されません

  → 差分増分バックアップを取るには FROM SCN指定が必要

- 現時点では、増分バックアップを使ったリカバリ時には、ソース側のUnplug時(最後の一貫性バックアップ時)に生成されたXMLファイルを指定する必要があります

  → 増分バックアップを使ったリカバリは、ソースPDBのClose、バックアップ取得、ファイル転送後にしか実施できない。停止時間を極小化することはできない

3) XTTSを用いたクロス・プラットフォームでのPDB移行

RMANを使わないクロス・プラットフォーム(同一エンディアン)でのPDB移行方法

1. Unplugとデータファイルコピーによる移行
   - PDBをCloseし、Unplug
   - データファイル、XMLファイルをターゲットに転送
   - PDBをPlugし、Open

2. PDB12.2の新機能のホット・クローン
   - ソースPDBの停止無く移行可能
   - ターゲットからソースへDB Linkを張る前提

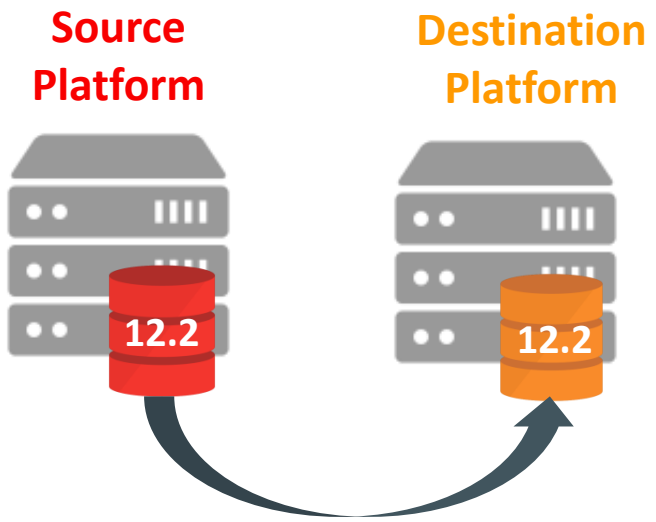
4)ネットワークを介したクロス・プラットフォーム・リカバリ

NEW IN 12.2

- RMAN 12cR1から FROM SERVICE句が利用可能
- RMAN 12cR2からはネットワーク経由でデータファイルのクロス・プラットフォームでのリストア＆リカバリが可能

```
RESTORE FOREIGN DATAFILE 5 FORMAT
'/oradata/orcl/users.dbf' FROM SERVICE 'inst1';
```

```
RECOVER FOREIGN DATAFILECOPY
'/oradata/orcl/users.dbf' FROM SERVICE 'inst1';
```



NEW IN 12.2

RMAN 12.2新機能一覧

1. RMANコマンド機能拡張
2. RMAN表単位リカバリ機能拡張
   - 表単位リカバリ中のディスクスペースチェック
   - REMAP SCHEMAを用いたスキーマを跨る表単位リカバリ
3. XTTSを用いたクロス・プラットフォームでのPDB移行
4. ネットワークを介したクロス・プラットフォーム・リカバリ

Online Operation

表のオンライン再定義とは

- 下記のような理由で表の構造を論理的・物理的に変更する場合に、表の可用性に大きな影響を与えずに構造を変更することができるメカニズム
  - 問合せまたはDMLのパフォーマンスを改善するため
  - アプリケーションの変更に対応するため
  - 記憶域を管理するため
- 再定義プロセスの大部分で、問合せ及びDMLを使用して表にアクセスできる
- 表が排他モードでロックされるのは、表のサイズに関係なくごくわずかな間のみでユーザに対しては完全に透過的

表のオンライン再定義フェーズ

1. 表のオンライン再定義操作の妥当性の検証

```
SQL> DBMS_REDEFINITION.CAN_REDEF_TABLE()
```

2. 仮表の作成(再定義後の属性で作成)

```
SQL> CREATE TABLE new_table ...;
```

3. 表のオンライン再定義の開始 → データの初期同期が行われる

```
SQL> DBMS_REDEFINITION.START_REDEF_TABLE()
```

4. 仮表に対してトリガーや索引及び制約の定義

```
SQL> DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS();
```

5. オンライン表再定義の終了 → データの最終同期が行われる

```
SQL> DBMS_REDEFINITION.FINISH_REDEF_TABLE()
```

表のオンライン再定義フェーズ

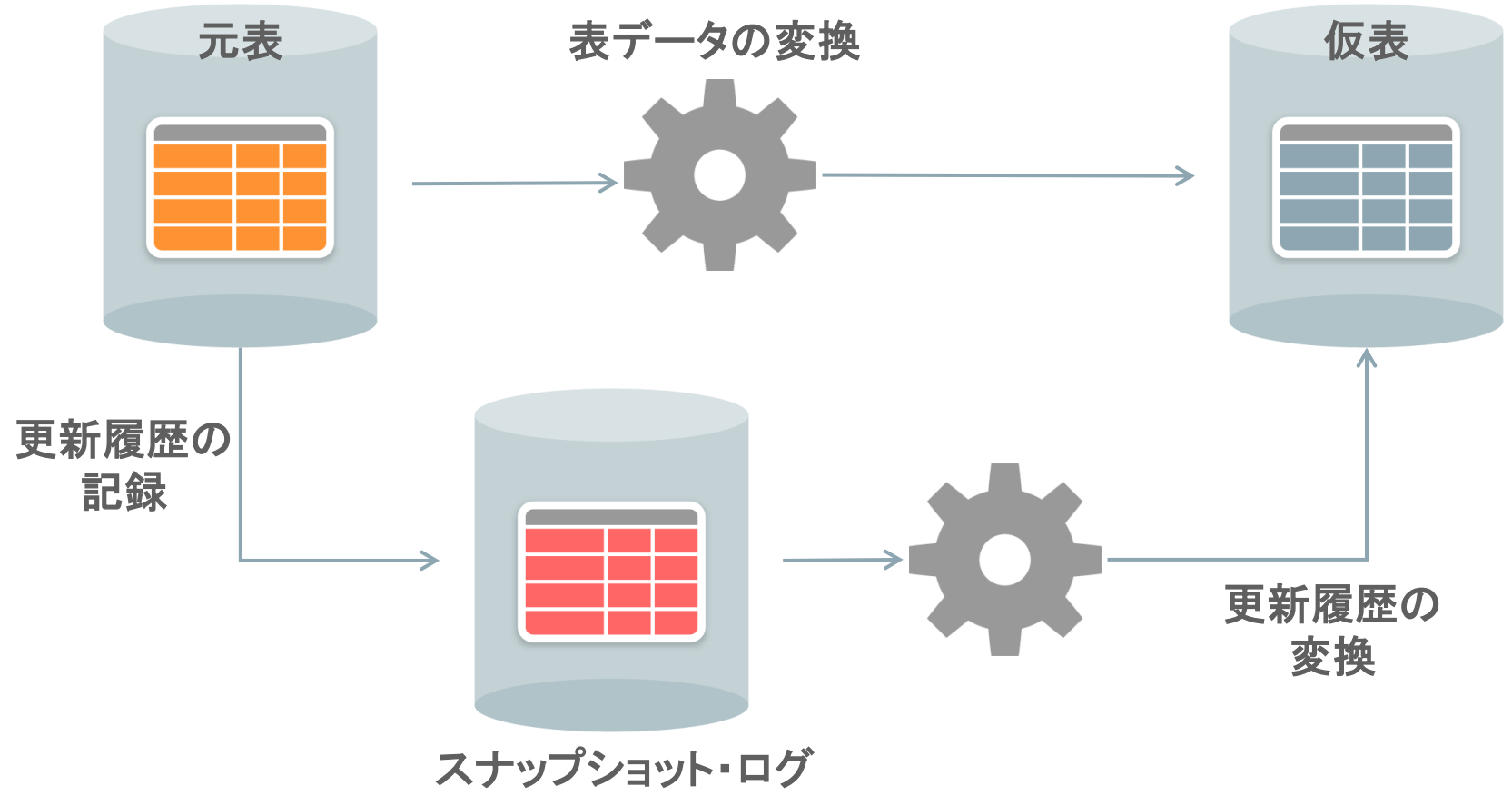
- 元表と仮表の間で同期を取る
  - 再定義の開始、終了を行う間に元表に対する変更が多い場合は、同期を行うことが望ましい

```
SQL> DBMS_REDEFINITION.SYNC_INTERIM_TABLE()
```

- 再定義の中断

```
SQL> DBMS_REDEFINITION.ABORT_REDEF_TABLE()
```

表のオンライン再定義の動作



NEW IN 12.2

オンライン・オペレーション 12.2 新機能一覧

1. オンライン再定義のエラーからの再開
2. 再定義のロールバック
3. 再定義の進捗確認
4. 再定義中のバッチ更新の最適化
5. 依存関係のあるマテリアライズド・ビューをリフレッシュ
6. オンラインテーブル移動
7. 非パーティション表をオンラインでパーティション化
8. パーティション/サブパーティションのオンライン分割
9. パーティション交換のためのテーブル作成
10. Filteredパーティションメンテナンス

上記7-10の機能については、
パーティション機能のセッションで紹介

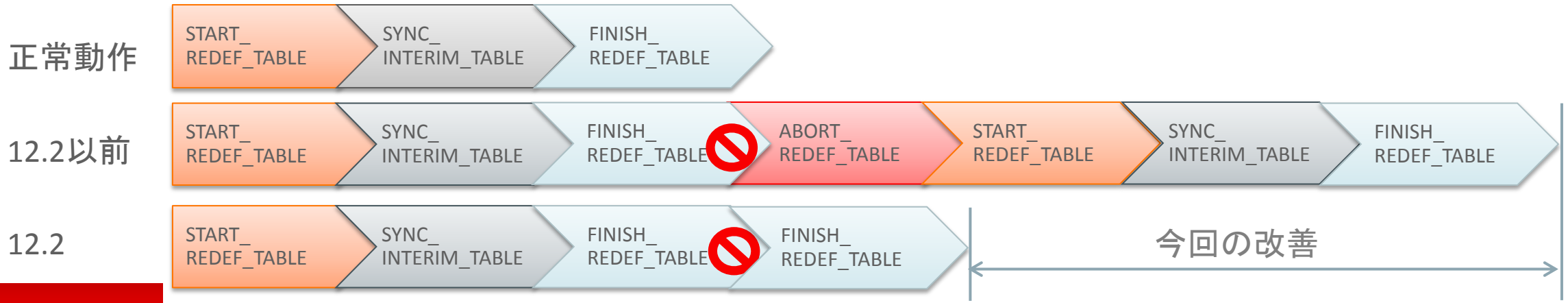
ORACLE

NEW IN 12.2

1) オンライン再定義のエラーからの再開

機能概要

- *SYNC_INTERIM_TABLE/ FINISH_REDEF_TABLE*中に失敗した場合、失敗の原因対処を実施後に再実行が可能となった
  - *START_REDEF_TABLE, ALTER TABLE MOVE ONLINE, ALTER TABLE MODIFY PARITION ONLINE*は対象外
  - *DBA_REDEFINITION_STATUS.ACTION*列 を見て対処実施
  - 以前は再定義プロセスを強制終了(*ABORT_REDEF_TABLE*)し、内部的に作成される一時表、ログを削除し、再定義プロセスの最初から再実行する必要があった



NEW IN 12.2

1) オンライン再定義のエラーからの再開

実行例(ORA-12081からの復旧)

実行例

```
-- オンライン再定義開始
SQL> BEGIN
  2  DBMS_REDEFINITION.START_REDEF_TABLE(
  3  uname        => 'ユーザ名',
  4  orig_table   => '元表名',
  5  int_table    => '仮表名',
  6  options_flag => DBMS_REDEFINITION.CONS_USE_PK);
  7  END;
  8  /

SQL> insert into ユーザ名.元表名  values (1);
SQL> commit;

-- エラーを発生させるために READ ONLY化(次のデータ同期処理に失敗)
SQL> alter table ユーザ名.仮表名  read only;
```

NEW IN 12.2

1) オンライン再定義のエラーからの再開

実行例(ORA-12081からの復旧)

実行例

```
-- 更新されたデータを一旦同期させるが、READ ONLY化されているので失敗する
SQL> BEGIN
  2  DBMS_REDEFINITION.FINISH_REDEF_TABLE('ユーザ名', '元表名', '仮表名');
  3  END;
  4  /
*
ERROR at line 1:
ORA-42009: error occurred while synchronizing the redefinition
ORA-12008: error in materialized view or zonemap refresh path
ORA-12081: update operation not allowed on table "ユーザ名"."仮表名"
ORA-06512: at "SYS.DBMS_REDEFINITION", line 219
ORA-06512: at "SYS.DBMS_REDEFINITION", line 5311
ORA-06512: at line 2
```

NEW IN 12.2

1) オンライン再定義のエラーからの再開

実行例(ORA-12081からの復旧)

実行例

```
-- 再定義の状態を確認
SQL> SELECT BASE_TABLE_NAME, INTERIM_OBJECT_NAME, OPERATION, STATUS,
RESTARTABLE, ACTION FROM DBA_REDEFINITION_STATUS;

BASE_TABLE_NAME  INTERIM_OBJECT_NAME  OPERATION           STATUS  R ACTION
---------------- -------------------- ------------------- ------- - --------------
ユーザ名          仮表名               FINISH_REDEF_TABLE Failure Y Fix 12081 error

-- エラーの原因対処
SQL> alter table ユーザ名.仮表名 read write;

-- エラー解消後、12.2からは再実行をするだけで良い
SQL> BEGIN
  2  DBMS_REDEFINITION.SYNC_INTERIM_TABLE('ユーザ名', '元表名', '仮表名');
  3  END;
  4  /
```

NEW IN 12.2

2) オンライン再定義のロールバック

機能概要

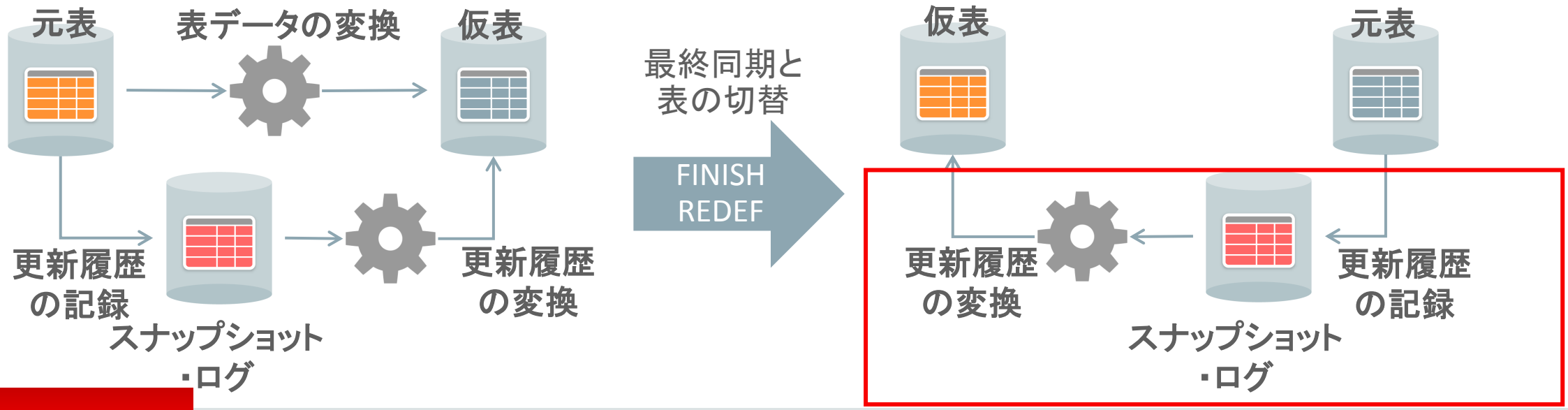
- オンライン再定義完了後の状態から、再定義前の状態に戻すことが可能となった
  - オンライン再定義完了後の表に対する更新データは保存
- 必要なステップは下記2つ
  - 再定義プロセスの中で実行するプロシージャ(*START_REDEF_TABLE, FINISH_REDEF_TABLE*)の引数としてROLLBACKに関するオプションを指定する
  - オンライン再定義完了後に実際にロールバックする場合には、*ROLLBACK*プロシージャを実行する

NEW IN 12.2

2) オンライン再定義のロールバック

ロールバック指定時の動作

- 元表と仮表のデータ同期用に使われる下記のマテリアライズド・ビュー・ログが*FINISH_REDEF_TABLE*完了後にも残り、逆方向のデータ同期のために更新履歴を保持
- 元表へのロールバックをしない判断をした場合は、*ABORT_ROLLBACK*を実行する
  - 上記のマテリアライズド・ビュー・ログが削除される



NEW IN 12.2

2) オンライン再定義のロールバック

実行例(表への列追加をロールバック)

実行例

```
-- 元表と仮表の定義確認
SQL> desc ユーザ名.元表名;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COL1                                      NOT NULL NUMBER
 COL2                                               CHAR(10)

SQL> desc ユーザ名.仮表名;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COL1                                      NOT NULL NUMBER
 COL2                                               CHAR(10)
 COL3                                               NUMBER
```

NEW IN 12.2

2) オンライン再定義のロールバック

実行例(表への列追加をロールバック)

実行例

```
-- ロールバック機能を有効にしてオンライン表再定義を開始
SQL> BEGIN
  2   DBMS_REDEFINITION.START_REDEF_TABLE(
  3   uname                    => 'ユーザ名',
  4   orig_table               => '元表名',
  5   int_table                => '仮表名',
  6   options_flag             => DBMS_REDEFINITION.CONS_USE_PK,
  7   col_mapping              => 'col1 col1, col2 col2, 0 col3',
  8   enable_rollback          => true);
  9   END;
 10   /
```

NEW IN 12.2

2) オンライン再定義のロールバック

実行例(表への列追加をロールバック)

実行例

```
-- ロールバック機能を有効にした状態でオンライン表再定義を完了
SQL> BEGIN
  2   DBMS_REDEFINITION.FINISH_REDEF_TABLE(
  3   uname                    => 'ユーザ名',
  4   orig_table               => '元表名',
  5   int_table                => '仮表名',
  6   disable_rollback         => false);
  7   END;
  8   /
```

NEW IN 12.2

2) オンライン再定義のロールバック

実行例(表への列追加をロールバック)

実行例

```
-- 再定義が完了したことを確認

SQL> desc ユーザ名.元表名;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COL1                                      NOT NULL NUMBER
 COL2                                               CHAR(10)
 COL3                                               NUMBER    ← ★ 列が追加されている

-- 再定義完了後の新定義の表に対する更新を行う
SQL> insert into ユーザ名.元表名  values (1,'AAA',1);
SQL> commit;
```

NEW IN 12.2

2) オンライン再定義のロールバック

実行例(表への列追加をロールバック)

実行例

```
-- 元表のレコード確認
SQL> select * from ユーザ名.元表名;

                    COL1 COL2       COL3
-------------------------- ---------- ----
                         1 AAA           1

-- ロールバックを実施
SQL> exec DBMS_REDEFINITION.ROLLBACK('ユーザ名','元表名','仮表名');

-- 再定義完了後に更新したデータが反映されているか
SQL> select * from ユーザ名.元表名;

                    COL1 COL2
-------------------------- ----------
                         1 AAA
```

← ★ ロールバックしても再定義完了後に追加したレコードは残る
(但しCOL3の値は失われる)

NEW IN 12.2

3) オンライン再定義の進捗確認

機能概要

- 12.1まではオンライン再定義の各プロシージャの進捗状況を確認する手段がなかった
- 12.2からは、v$online_redefビューにより、オンライン再定義プロセスの各プロシージャの実行状態を確認できる機能が提供された
  - 実行中プロシージャ
  - 実行中のプロシージャの進捗状況(何ステップ中の何処)
  - 実行されているSQL文

ORACLE

NEW IN 12.2

3) オンライン再定義の進捗確認

実行例(START_REDEF_TABLE実行中)

実行例

```
SQL> exec print_table('select * from gv$online_redef');
<省略>
TABLE_OWNER                    : ユーザ名
ORIGINAL_TABLE_NAME            : 元表名
INTERIM_TABLE_NAME             : 仮表名
PARTITION_NAME                 :
OPERATION                      : START_REDEF_TABLE
SUBOPERATION                   : complete refresh MVs
DETAILED_MESSAGE               :
PROGRESS                       : step 6 out of 7
REFRESH_STATEMENT_SQL_ID       : 28a75hgzdhwbm
REFRESH_STATEMENT              : /* MV_REFRESH (INS) */INSERT /*+
BYPASS_RECURSIVE_CHECK APPEND SKIP_UNQ_UNUSABLE_IDX */ INTO "ユーザ名"."仮表名
"("COL1","COL2") SELECT "元表名"."COL1","元表名"."COL2" FROM "ユーザ名"."元表名"
```

例えば、START_REDEF_TABLEプロシージャでは、元表と仮表の完全リフレッシュを行っていることが分かる

参考:print_table https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:1035431863958

NEW IN 12.2

4) 大量バッチ更新の最適化

機能概要

- DBMS_REDEFINITION.EXECUTE_UPDATEプロシージャによって、テーブルへの大量更新の性能を改善可能
- オンライン再定義の仕組みを使って、指定したUPDATE文の実行を内部的にダイレクト・パス・インサートする INSERT SELECT文に変換してくれる
- NOLOGGING属性のついた仮表を作成しそこにダイレクト・パス・インサートするためREDO生成量が小さくなる
- これまでのように、大量行を更新するようなUPDATE文を、INSERT SELECT文に書き換えるというチューニング[*]を行う必要がなくなる

参考：http://www.oracle.com/technetwork/jp/database/articles/pickup/index-1622021-ja.html

NEW IN 12.2

4) 大量バッチ更新の最適化

- 制限事項
  - オンライン再定義の全ての制限は適用される
  - 1つの表に対して複数のEXECUTE_UPDATEプロシージャを使った更新処理を実行することはできない (EXECUTE_UPDATEと同時に DML文を実行することはできる)
  - UPDATE文で起動するトリガーを作成できない
  - 利用できない型がある
  - その他(全件は「管理者ガイド」マニュアルを参照下さい)
- REDOが生成されないので、更新処理が終わったらバックアップを取得する必要がある
  - 前回取得のバックアップからリカバリできないため

NEW IN 12.2

4) 大量バッチ更新の最適化

実行例(全行UPDATE)

実行例

```
-- DBMS_REDEFINITION.EXECUTE_UPDATEを使った大量データ更新の例
SQL> DECLARE
 update_stmt VARCHAR2(300) := 'update test1.tab1 set col2=1';
BEGIN
 DBMS_REDEFINITION.EXECUTE_UPDATE(update_stmt);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT');
    DBMS_REDEFINITION.ABORT_UPDATE(update_stmt);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('Reason for failure is'|| SQLERRM);
       IF (SQLCODE = 100) THEN DBMS_REDEFINITION.ABORT_UPDATE(update_stmt);
       END IF;
END;
/
```

実行したいUPDATE文

NEW IN 12.2

4) 大量バッチ更新の最適化

REDO生成量の比較

☐ execute_update文

```
Statistic                              Total     per Second     per Trans
-------------------------------- ------------------ -------------- -------------
redo size                        134,404,208    2,739,309.2   3,054,641.1
```

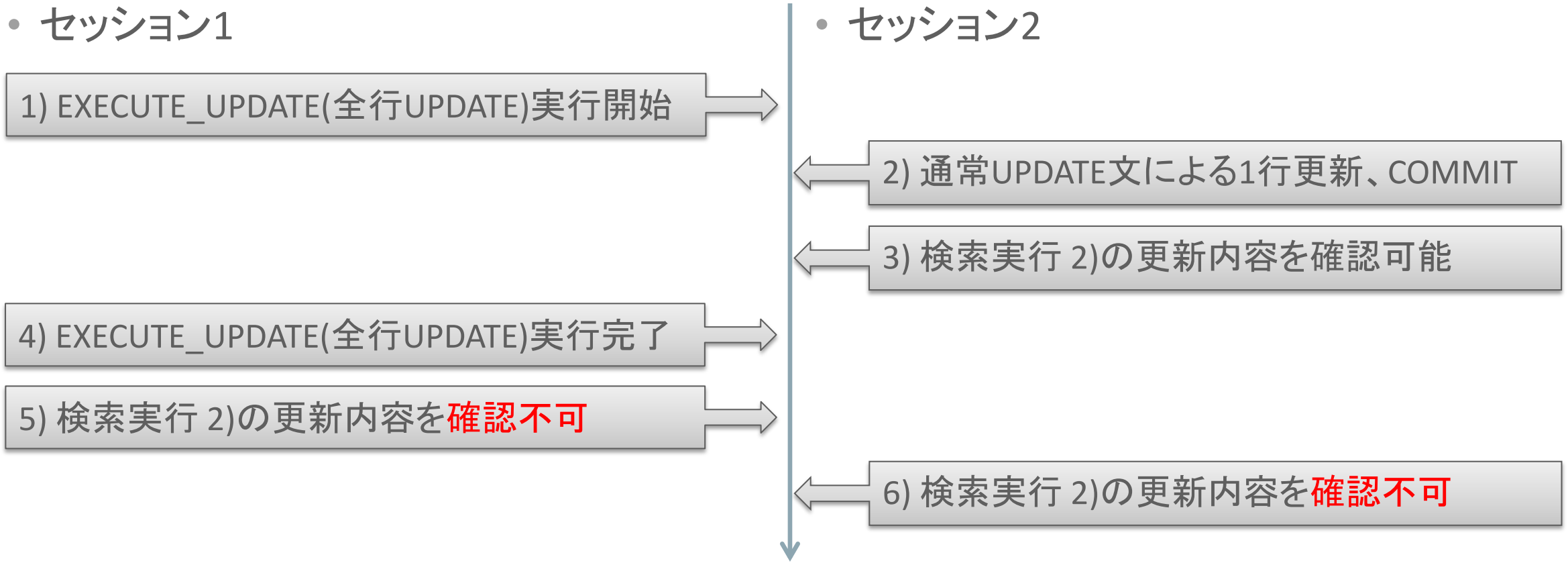
☐ 通常のupdate文

```
Statistic                              Total     per Second     per Trans
-------------------------------- ------------------ -------------- -------------
redo size                        652,925,188    6,061,318.1 6.5292519E+08
```

NEW IN 12.2

4) 大量バッチ更新の最適化

注意点



5) 依存するマテリアライズド・ビューを差分リフレッシュ

NEW IN 12.2

- 12.2よりオンライン再定義中に元表と依存関係のあるマテリアライズド・ビューを差分リフレッシュ可能
- SYNC_INTERIM_TABLEプロシージャ実行の一部としてリフレッシュを実行する
- これまでは表のオンライン再定義完了後に完全リフレッシュする必要があった

```
SQL> BEGIN
  2  DBMS_REDEFINITION.START_REDEF_TABLE(
  3     uname                  => 'ユーザ名',
  4     orig_table             => '元表名',
  5     int_table              => '仮表名',
  6     options_flag           => DBMS_REDEFINITION.CONS_USE_PK,
  7     REFRESH_DEP_MVIEWS     => 'Y');
  8  END;
  9  /
```

自動的にリフレッシュさせたい場合は、
START_REDEF_TABLE実行時に
REFRESH_DEP_MVIEWS に ‘Y’ を指定しておく

NEW IN 12.2

6) オンラインテーブル移動

- 12.1まではALTER TABLE … MOVE文を実行中のDML操作は不可
  - MOVE PARTITION, MOVE SUBPARTITIONに関しては例外としてONLINEキーワードがサポートされており、DMLをブロックせず実行可能だった
- 12.2では非パーティション表に関してもONLINEキーワードに対応し、DMLをブロックすることなく実行することが可能
- テーブル移動時に自動的に索引メンテナンスも行われる

```
SQL> alter table <元表名> move online compress;
```

オンライン・オペレーション 12.2 新機能一覧

NEW IN 12.2

1. オンライン再定義のエラーからの再開
2. 再定義のロールバック
3. 再定義の進捗確認
4. 再定義中のバッチ更新の最適化
5. 依存関係のあるマテリアライズド・ビューをリフレッシュ
6. オンラインテーブル移動
7. 非パーティション表をオンラインでパーティション化
8. パーティション/サブパーティションのオンライン分割
9. パーティション交換のためのテーブル作成
10. Filteredパーティションメンテナンス

上記7-10の機能については、
パーティション機能のセッションで紹介

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®