
ORACLE®

ORACLE

Oracle Database 12c Release 1

アプリケーションの継続性

日本オラクル株式会社

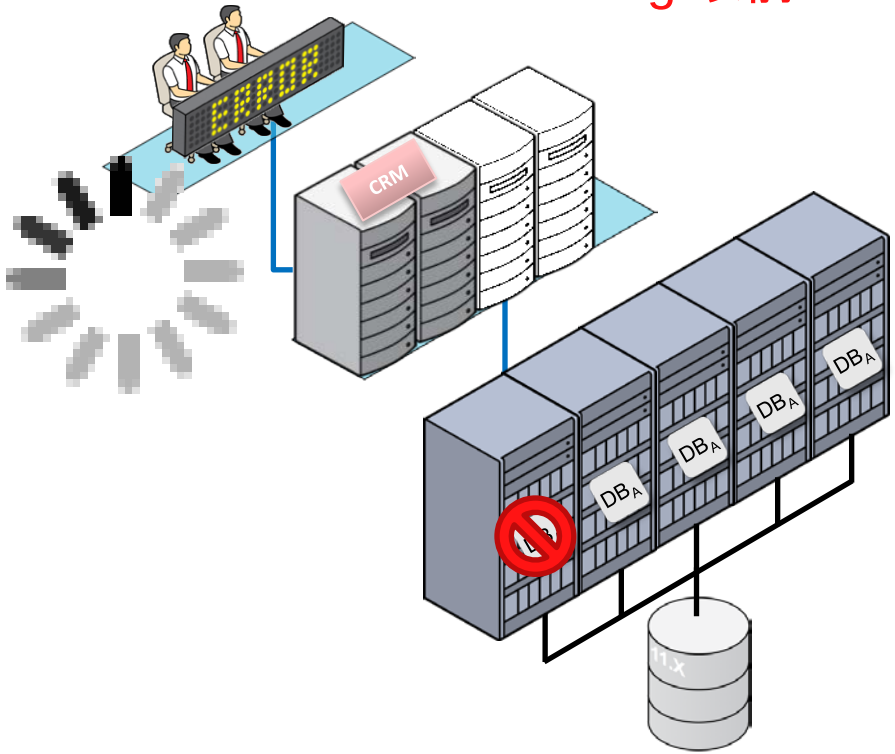
以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

想定外（計画外）の停止、クラッシュ

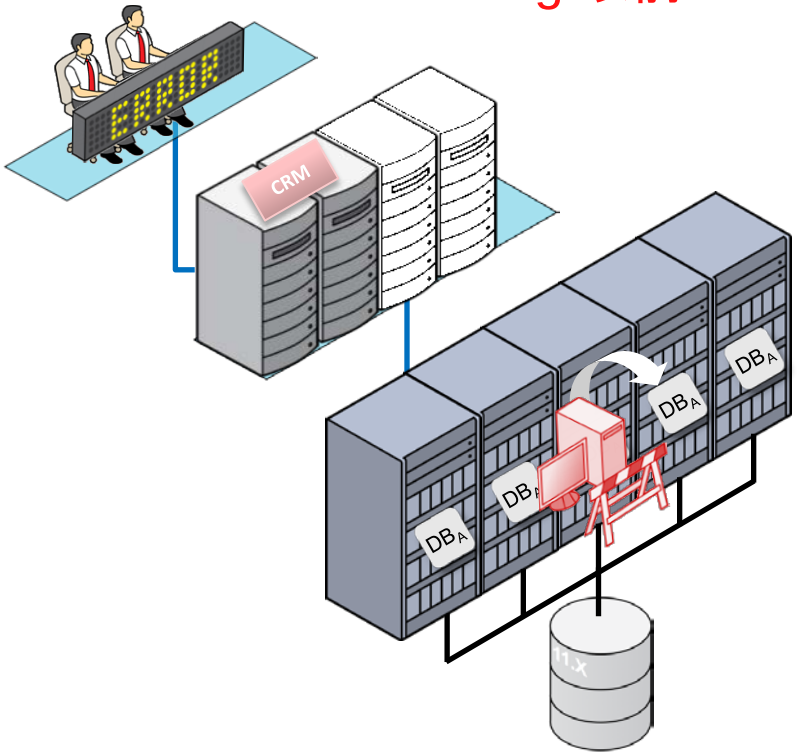
Oracle Database 11g 以前



- データベースの異常停止により実行中だったトランザクションのどこまでをコミットされたのか、どこから再入力する必要があるのか判断できないケースがありました。
- 2重コミット
- 不明トランザクションをクリーンアップするための中間層の初期化
- DB異常に対応するためのアプリケーションコードの複雑化

計画停止時にトランザクションが失われるケース

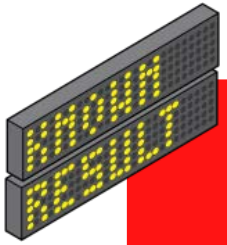
Oracle Database 11g 以前



- 計画停止の方が非計画停止より頻度が多い傾向があります
- FAN と Oracle Connection Pool(Weblogic,UCP,OCI,..) の組み合わせによりアイドル（使われていない）接続のクリーンアップは 10gR2 から可能
- FAN と Oracle Connection Pools でも長時間接続を離さないアプリには対応できていなかった

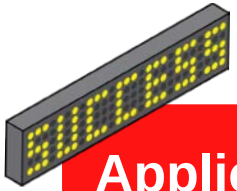
アプリケーション開発で実装が難しい点を解決

Oracle Database 12c 新機能



Transaction Guard

最後のトランザクション結果を確実に取り出すためのプロトコルとAPI



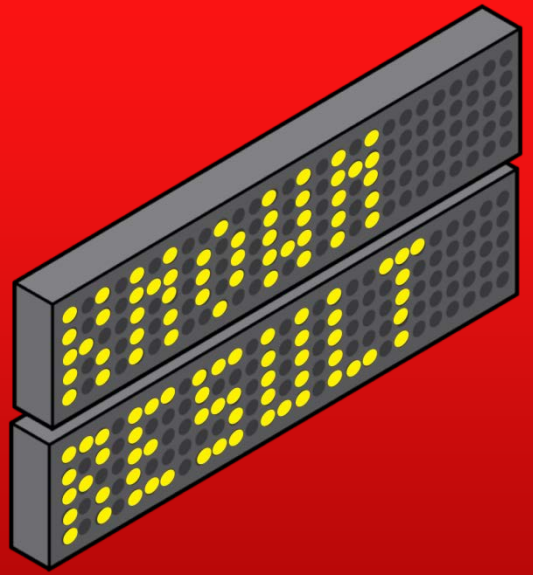
Application Continuity for Java

想定外の停止、計画停止時に実行中だったセッションを安全に再実行する

アジェンダ

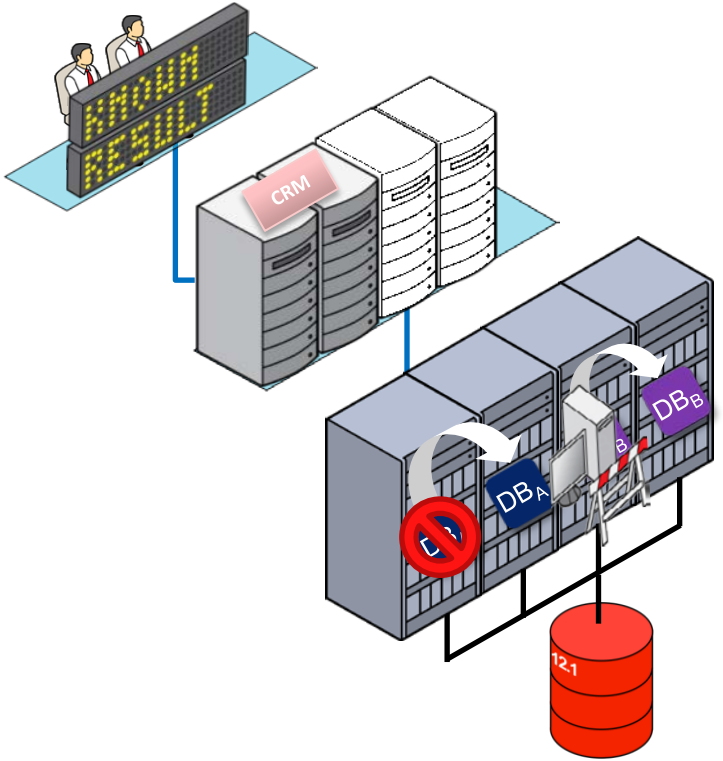
1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

Transaction Guard



Transaction Guard

コミット結果を保存。後から取り出し可能



- トランザクションのコミット結果を取り出すAPI
- このAPIを使うことにより2重発注などのよくある問題を未然に防ぐことができる
- Transaction Guard APIによりアプリケーションは自分で再実行するのか、または、エンドユーザーに再入力を促すかなどの判断が可能に
- “Application Continuity for Java” 機能でも使われる
- JDBC-Thin, C/C++ (OCI/OCCI), ODP.Net クライアント用 API

Transaction Guard の対応、非対応

- DB 12c 以上 が必要
- さまざまなタイプの COMMIT に対応
  - ローカル・トランザクション
  - 自動コミット(auto commit), Commit on Success
  - PL/SQL ブロックの中での COMMIT
  - DDL, Parallel DDL, Remote
- 未対応
  - XA
  - Active Data Guard でスタンバイに接続しデータベース・リンク経由でプライマリを更新

アジェンダ

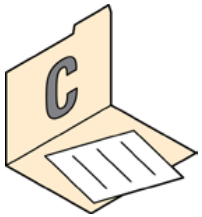
1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

回復可能、リトライ可能　なエラー

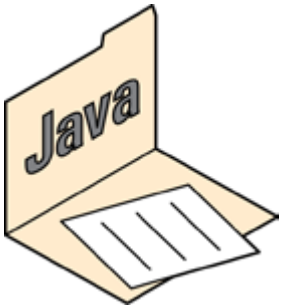
ハードウェア、ネットワーク、OS …

- Transaction Guard, Application Continuity の両方共 “Recoverable” なエラーに対応するための機能
- Recoverable(回復可能)、Transient(一過性の), temporary
- sqlplus , OCI などでは
  - ORA-3113: 通信チャネルでend- of- file が検出されました
  - ORA-3135: connection lost contact
- Java(JDBC)では
  - java.sql.SQLRecoverableException: No more data to read from socket

エラーのクラス

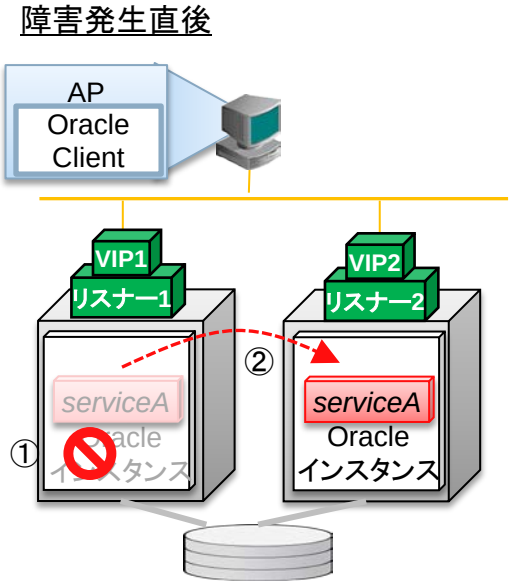


- DB12c の OCI で判別するためのAPIが追加されました
  - エラー番号の列挙は不要に
- JDBC では以前から定義されていて Oracle JDBC ドライバー も実装していました。
- 多くのISV（ソフトウェア・ベンダー）さんはこの例外のときは再試行するようにアプリを作られていると推測しています



RAC, Data Guard 固有の Recoverable エラー

- RAC で右図のような状態
  - ORA-12514: TNS:listener does not currently know of service requested in connect descriptor
- Data Guard でスイッチオーバー中
  - ORA-16456: スタンバイへのスイッチオーバーが進行中であるか完了しています

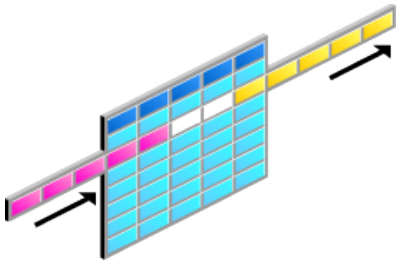


アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
    i. Recoverable Exception(回復可能な例外)
    ii. 動作
    iii. 設定
3. Application Continuity for Java
    i. 動作概要
    ii. Weblogic Server 12.1.2 での使用
    iii. 動作詳細
    iv. 設定
4. Application Continuity for Java デモ
5. Q&A

DB12c サーバー側でコミット履歴を保持するテーブル

- LTXID: Logical Transaction(ロジカル・トランザクション) ID
- SYSAUX 表領域にデフォルトで作成されます
- 別表領域への移動のみサポートされます
- パーティション表でRACのノード数分作られます

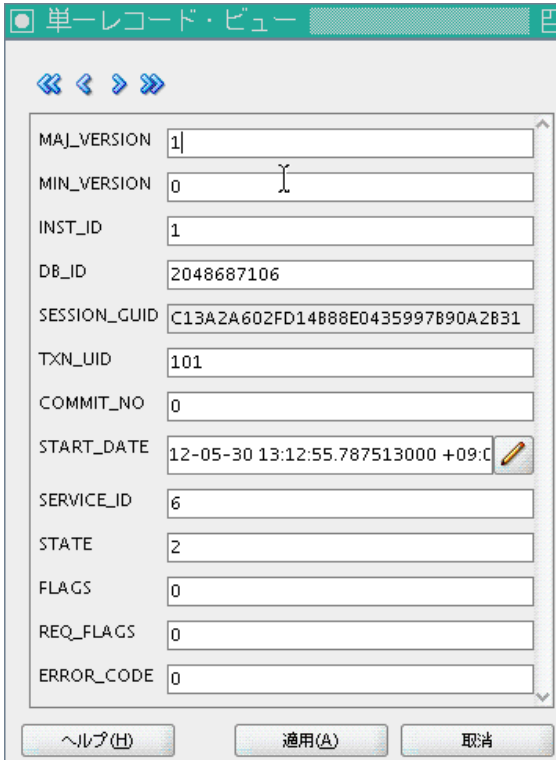


LTXID_TRANS | cdemocp.sql | DBMS_APP_CONT | セッション

列 | データ | 制約 | 権限付与 | 統計 | トリガー | フラッシュバック | 依存性 | 詳細 | パーティション | 索引 | SQL

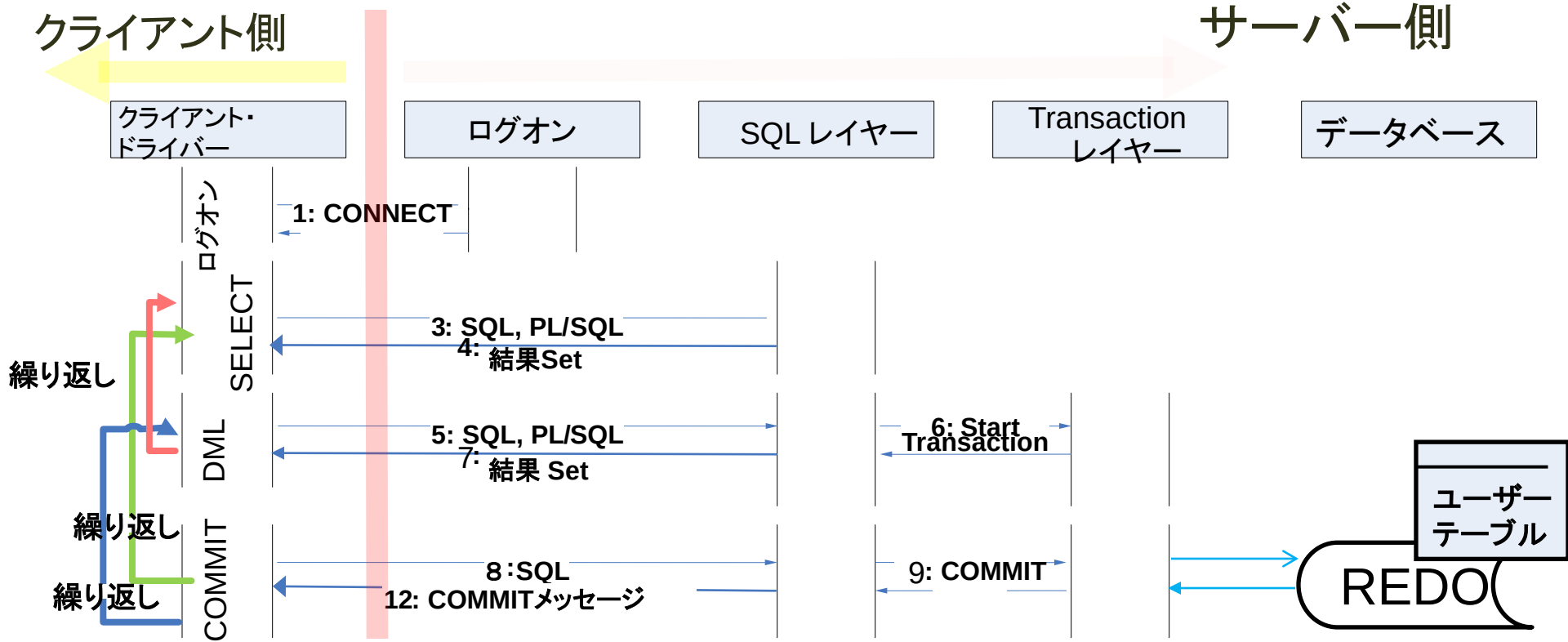
アクション...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1	MAJ_VERSION	NUMBER	No	(null)
2	MIN_VERSION	NUMBER	No	(null)
3	INST_ID	NUMBER	No	(null)
4	DB_ID	NUMBER	No	(null)
5	SESSION_GUID	RAW	No	(null)
6	TXN_UID	NUMBER	No	(null)
7	COMMIT_NO	NUMBER	No	(null)
8	START_DATE	TIMESTAMP(6) WITH TIME ZONE	No	(null)
9	SERVICE_ID	NUMBER	No	(null)
10	STATE	NUMBER	No	(null)
11	FLAGS	NUMBER	No	(null)
12	REQ_FLAGS	NUMBER	No	(null)
13	ERROR_CODE	NUMBER	No	(null)

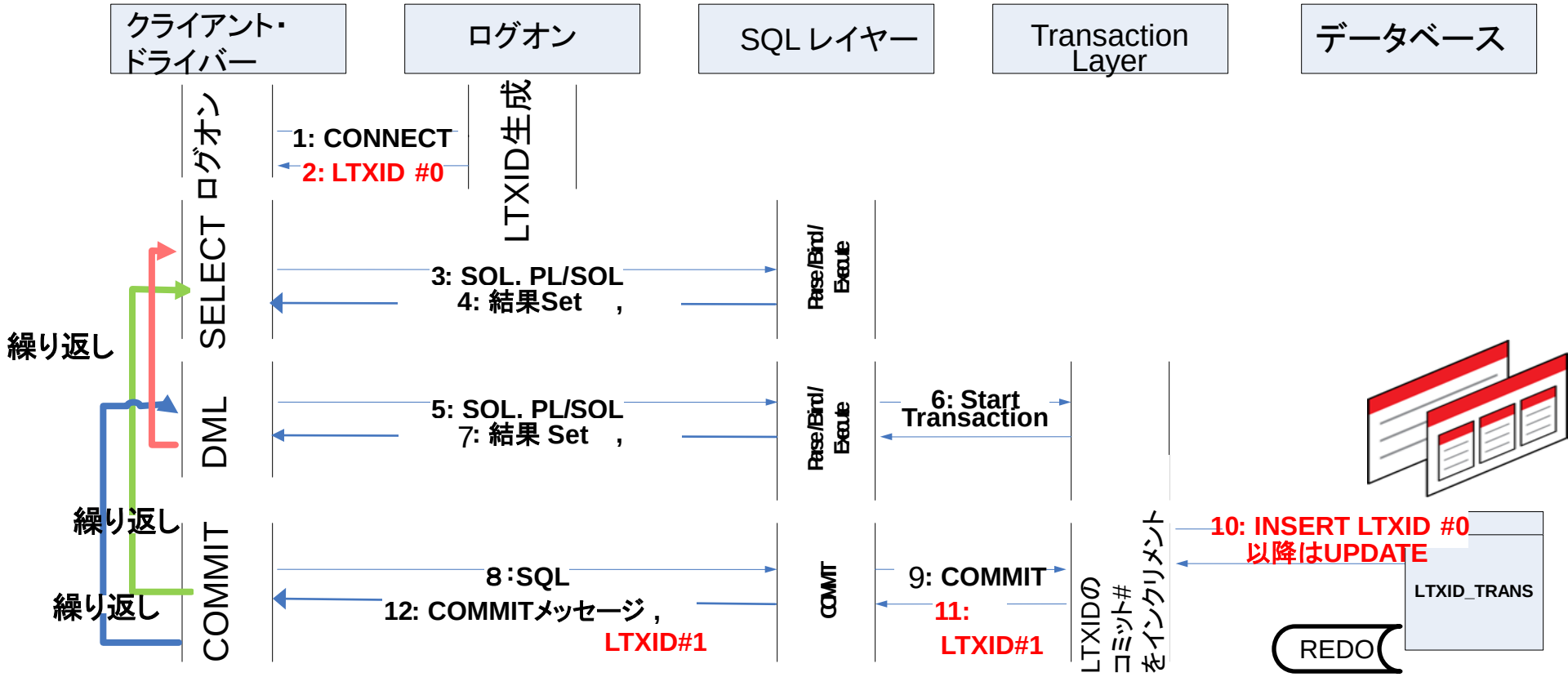


一般的なセッションの 12c でのデフォルトの動作

以前のバージョンと同じ

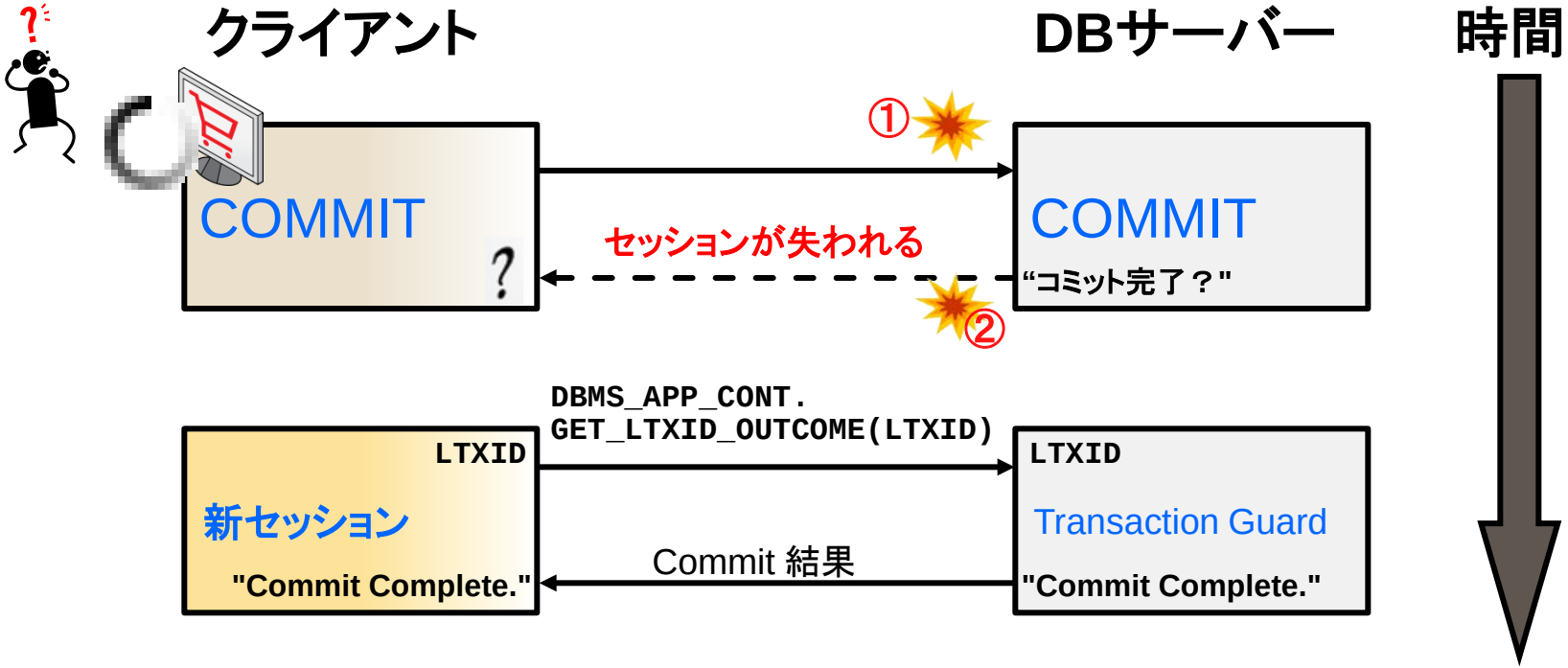


12c で Transaction Guard 有効時の動作



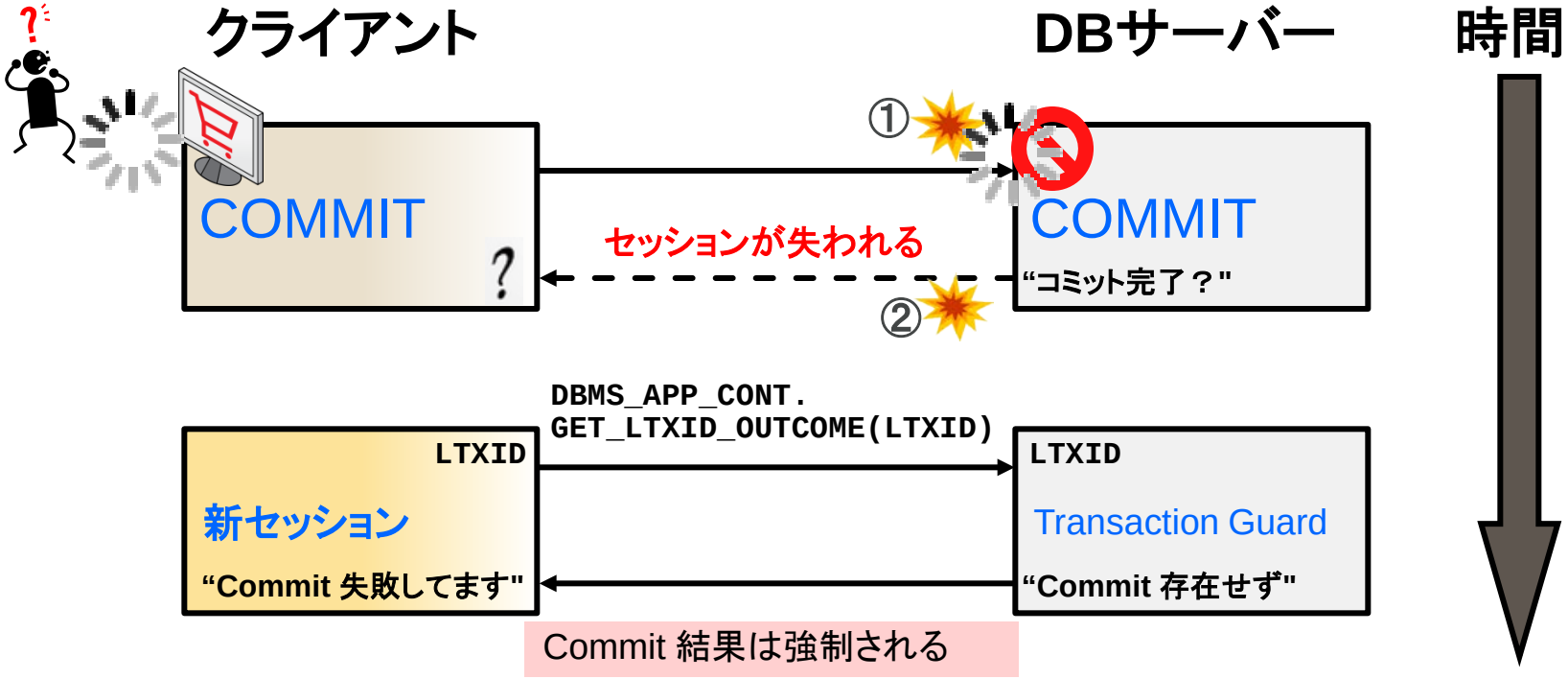
API使用 および 動作 パターン#1

コミット発行後に返事がなかったが実際には成功していた場合



API使用 および 動作 パターン#2

コミット発行後に返事がなく実際には 失敗 していた場合



アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

アプリケーションのためのDB側の設定

- 権限付与

```
GRANT execute on DBMS_APP_CONT to [アプリケーションユーザー]
```

- 12c 新規PL/SQL パッケージ
  - ラッパーAPI から呼び出される場合もあります

```
SQL> desc dbms_app_cont

PROCEDURE GET_LTXID_OUTCOME

 Argument Name                  Type                    In/Out
 ------------------------------ ----------------------- ------ --------
 CLIENT_LTXID                   RAW                     IN
 COMMITTED                      BOOLEAN                 OUT
 USER_CALL_COMPLETED            BOOLEAN                 OUT
```

サーバー側の設定: データベース・サービス属性

RAC の管理者管理サービスでの例

srvctl add service -d codedb -s GOLD -r serv1 -a serv2 -**commit_outcome** TRUE -retention 604800 -notification TRUE

- 12c 新規属性
- Retention
  - LTXID 保持期間
  - デフォルト: 24時間
  - 最大: 30日

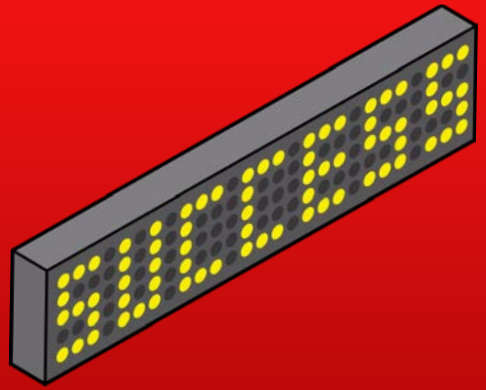
シングル・インスタンスでの例

```
declare
  params dbms_service.svc_parameter_array;
begin
  params('COMMIT_OUTCOME'):='true';
  params('RETENTION_TIMEOUT'):=604800;
  params('AQ_HA_NOTIFICATIONS'):='true';
  dbms_service.modify_service('[サービス名]',params);
end;
/
```

アジェンダ

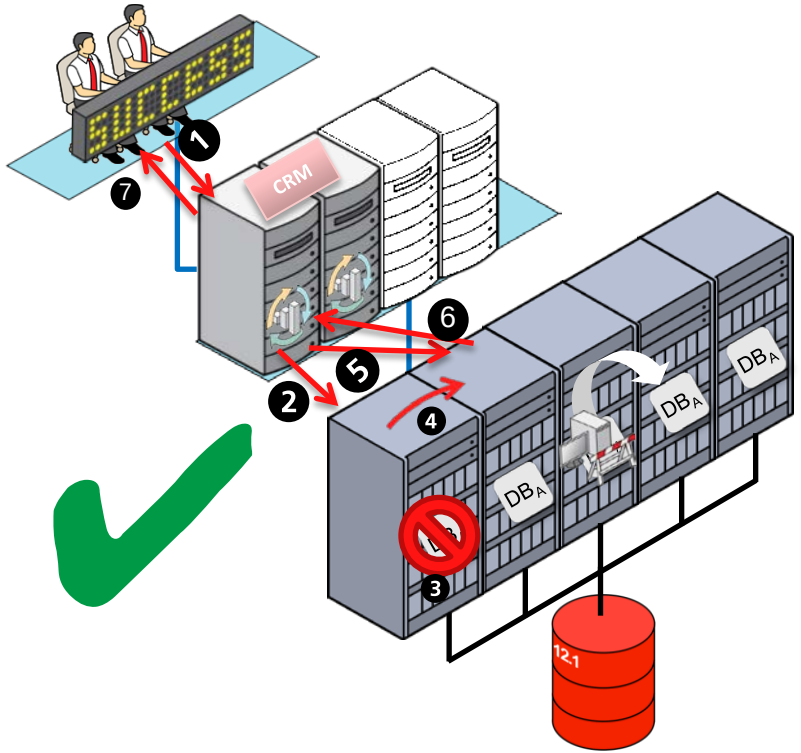
1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

Application Continuity for Java



Application Continuity for Java

計画外停止、および、計画停止をエンドユーザーに気づかせない



- Recoverable(回復可能)エラー発生時にDML(トランザクション)を再実行し再度データ投入
- 多くのハードウェア、ネットワーク、OS, ストレージのエラーや停止をエンドユーザーに気づかせずにトランザクション実行

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
    i. Recoverable Exception(回復可能な例外)
    ii. 動作
    iii. 設定
3. Application Continuity for Java
    i. 動作概要
    ii. Weblogic Server 12.1.2 での使用
    iii. 動作詳細
    iv. 設定
4. Application Continuity for Java デモ
5. Q&A

Application Continuityとは

- Application Continuityは、データベース障害時にアプリケーションのデータベース処理のフェイルオーバー(Replay)を透過的に実行する機能です。
- これは、Oracle Database12cとOracle JDBC Driver12cにより提供される機能で、JDBCを使用するアプリケーションやWebLogic12(12.1.2)上で動作するアプリケーションに適用できます。

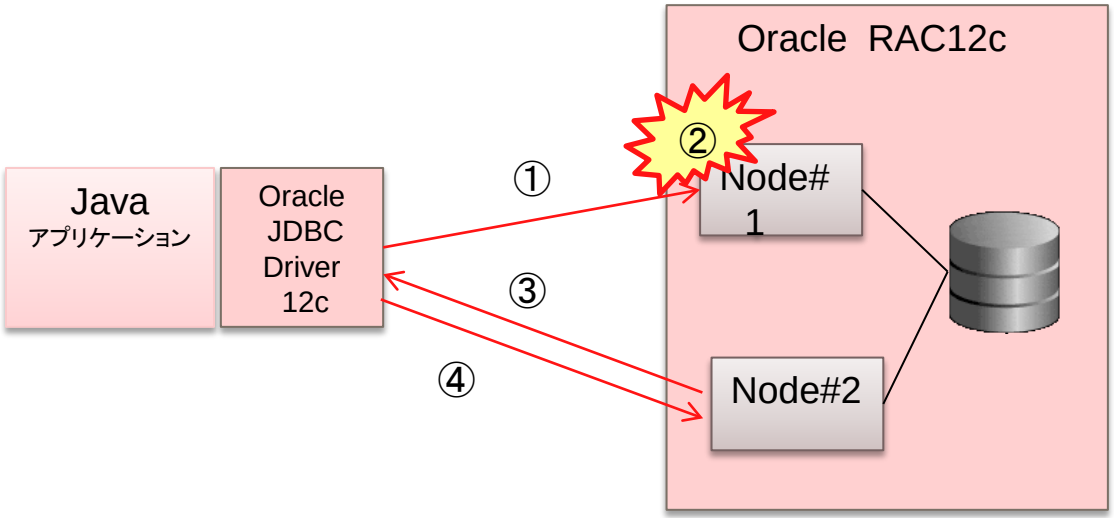
メリット	説明
データベース処理の可用性向上	アプリケーションがデータベースに対して処理を実行している最中にデータベースに障害が発生した場合でも、その復旧後や、RACであれば残存インスタンスにフェイルオーバーして処理を継続することができます。
アプリケーションでの対応不要	上述の機能を利用するために、特にアプリケーション側で特別なコードを記述する必要はないため、既存のアプリケーションのデータベース処理の可用性を容易に向上する事ができます。

Application Continuityの仕組み

Oracle RACの場合

- 下図は、クライアントがFANによりRAC側の障害を検知し、データベース処理を再実行（Replay)するまでの順序を説明したものです。

順序	処理
①	クライアントがRACサービスに対しアクセスし、データベース処理を実行
②	クライアントが接続しているノードで障害が発生
③	FANによりRAC側からノード障害を検知
④	設定した内容に基づきReplay（下記）が実行される。 1. クライアントが対象のサービスが動作す るノードへの新規セッションを確立。（この時、必要であれば任意の初期化処理をCallback実行させることも可能） 2. データベースへの処理を再実行。

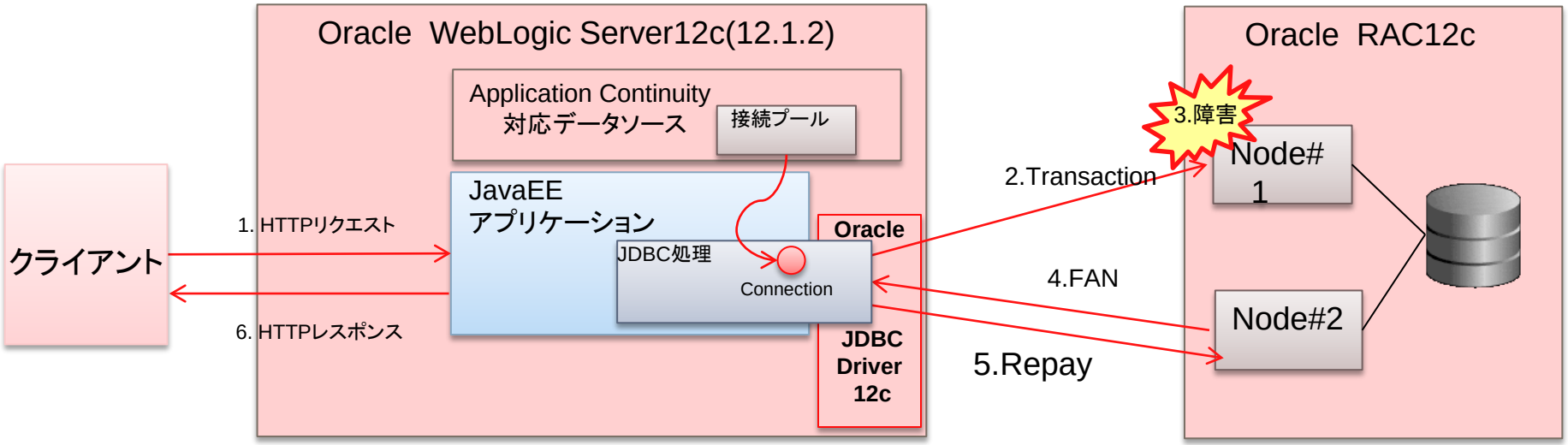


アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
    i. Recoverable Exception(回復可能な例外)
    ii. 動作
    iii. 設定
3. Application Continuity for Java
    i. 動作概要
    ii. Weblogic Server 12.1.2 での使用
    iii. 動作詳細
    iv. 設定
4. Application Continuity for Java デモ
5. Q&A

WebLogicデータソースのApplication Continuity対応

- 2013年7月10日にリリースされたWebLogic12c(12.1.2)では Application Continuityに対応したデータソースを構成できます。
- これにより、WebLogicで動作するアプリケーションがDB障害発生に透過的にJDBC処理をフェイルオーバー(Replay)させることが可能になります。



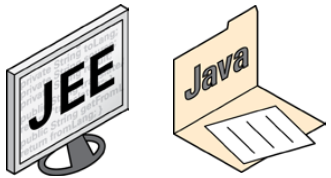
データソース作成ウィザード最終画面での確認

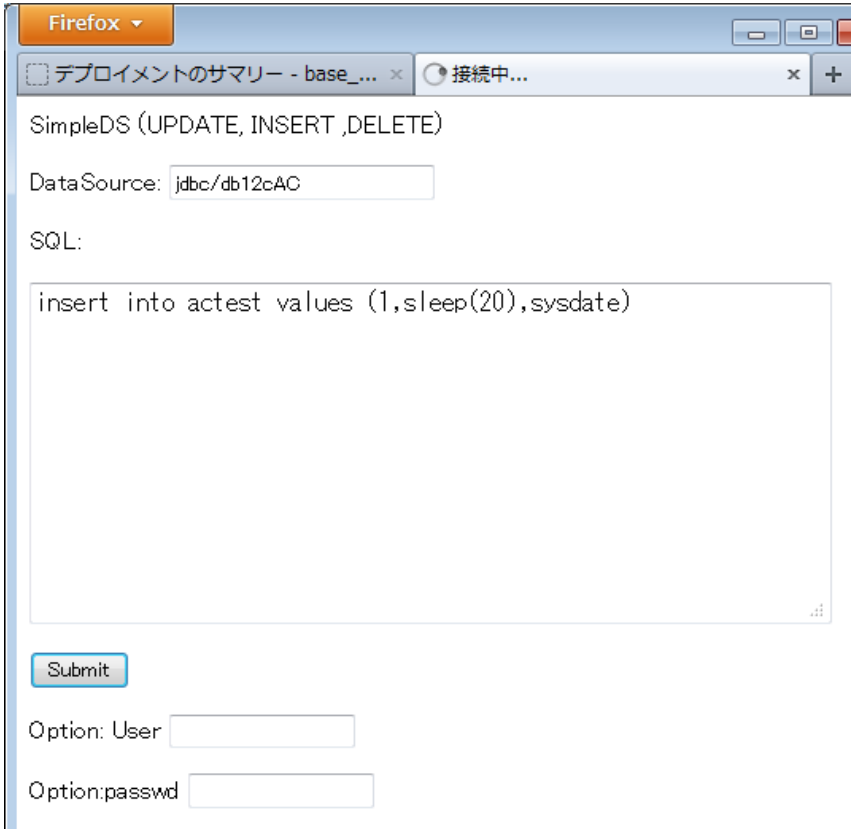
新規接続ファクトリクラス

- 「ドライバ・クラス名」が「oracle.jdbc.replay.OracleDataSourceImpl 」になっていることを確認
- DB12c に含まれるJDBCドライバーに含まれる
  - ojdbc6.jar
  - ojdbc7.jar
- UCP(Universal Connection Pool) でもこのクラスを使い Application Continuity 機能を利用することができます

使用と実行例①: アプリからのINSERT文発行

- このアプリケーションは比較的単純なサーブレットです。
- 何らかのフレームワーク(JPA等)を使うアプリケーションでも同様の動作をします
- Application Continuity データソースを利用し、INSERT文を発行するテスト用のJavaEEアプリケーションをWebLogicにデプロイします。
- INSERT文の対象となる表は、行が入っていない状態にしておきます。
- アプリケーションから（敢えて時間のかかる）INSERT文を発行します。

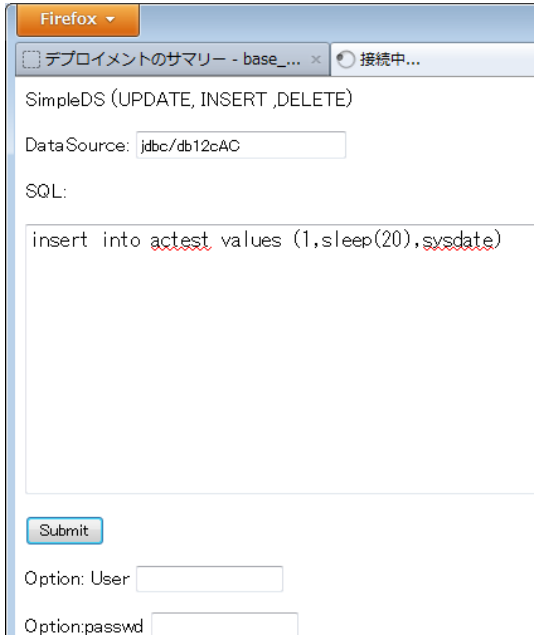




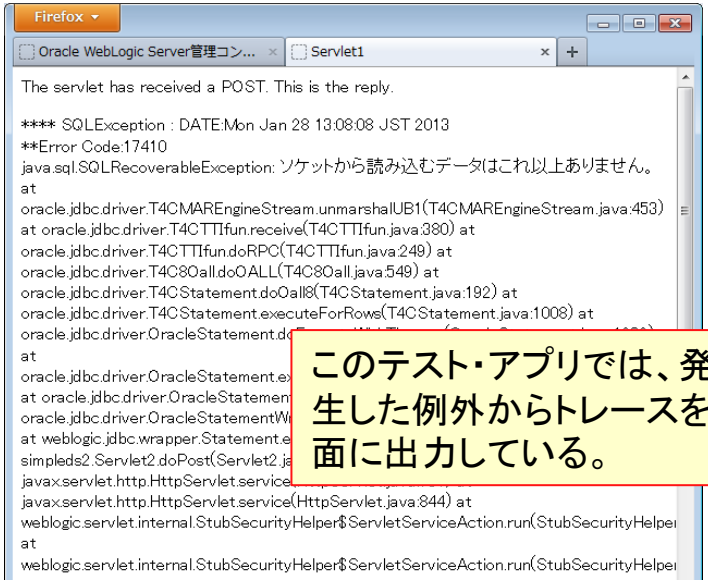
使用と実行例②: Recoverable エラー発生

- データベースを強制停止(shutdown abort) しても、アプリはまだ実行状態であることを確認します。

アプリはエラーにならず、実行中の状態



【参考】Application Continuity対応データソースを使用していない場合、同じ処理でも下図のようにエラーになる。



このテスト・アプリでは、発生した例外からトレースを画面に出力している。

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

JDBC Java アプリケーションのソースコード

“リクエスト” の定義など

- JDBC 処理に入ったところ、
  または、実行中に
  エラーを検知し再接続、再生
  - 例えば Thread.sleep() 中は
    ACはキックされない
- JDBC ドライバー自身が自分が
  受け取った処理だけを”再生”
  - Thread.sleep()は再生されない

```
import javax.sql.*;
...(中略）
DataSource ds = (DataSource)ic.lookup(“jdbc/db12cAC”);
                                    //データソースオブジェクト取得
Connection conn = ds.getConnection();          //コネクション取得
String sql = “select * from emp”               //実行するSQL文
Statement stmt = conn.createStatement();       //Statementの作成
ResultSet rset = stmt.executeQuery(sql);       //結果セットの取得
... ..(中略）
Thread.sleep(180000);
while(rset.next) { fetch…}
rset.close();  //結果セットクローズ処理
stmt.close();  //Statementクローズ処理
conn.close();  //コネクションクローズ処理
```

“リクエスト“
==
再生対象範囲

- WebLogic データソース、Universal Connection
  Pool(UCP) を使わない場合、
  ”リクエスト”範囲を定義するメソッド呼び出しが必要

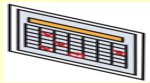
Application Continuity 動作: 3 つのフェーズ

Runtime(初回実行,障害発生前)	再接続	再実行(Replay,再生)
•JDBCの出すDB “コール”を全てJVM内のメモリーのキュー(配列)エリアに保存 •バインド変数 •実行後にDBから返されるリプレイ・コンテキストも保存	•タイムアウトチェック •新しい接続を得る •新接続先DBの正当性チェック •Transaction Guard 機能を呼び出し最後のトランザクションがCOMMITしたかどうかをチェック	•キュー中のコールを再実行 • コール実行毎に結果がRuntime時と一致しているか確認 •元のJavaアプリに処理が戻る

再生時に結果が変わる場合、再生は中断される

UPDATE 買い物カゴ SET OWNER='U12345' WHERE …

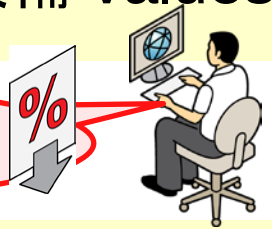
SELECT メーカー、値段 FROM 商品 WHERE .. 値段 < 3000



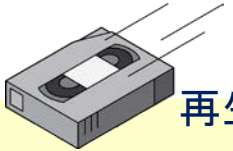
5 件ヒットしました

INSERT INTO 買い物候補 values(2480,'BrandCo','USB加湿器

USB加湿器値下げしよう
3480円 -> 2980円



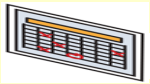
RACノードがクラッシュ



再生

UPDATE 買い物カゴ SET OWNER='U12345' WHERE …

SELECT メーカー、値段 FROM 商品 WHERE .. 値段 < 3000



6 件ヒットしました

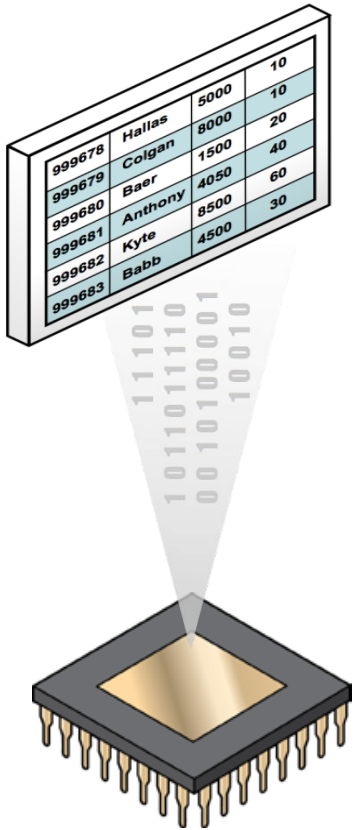
INSERT INTO 買い物候補 values(2480,'BrandCo','USB加湿器')

SQLException: ORA-41412: results changed during replay; failover cannot continue

結果が同じかどうかのチェック

高速に行う実装

- 初回の結果を全部保持するのではなくチェックサム(Hash)を計算して保持
- サーバー側で計算
- SQL毎にチェックサムをReplay　Contextに入れクライアントへ返す
- チェックサムの計算はSPARC チップ、Intelチップであれば専用演算命令を呼びだすのでオーバーヘッドは小さい
- LOB(BLOB, CLOB)は対象外

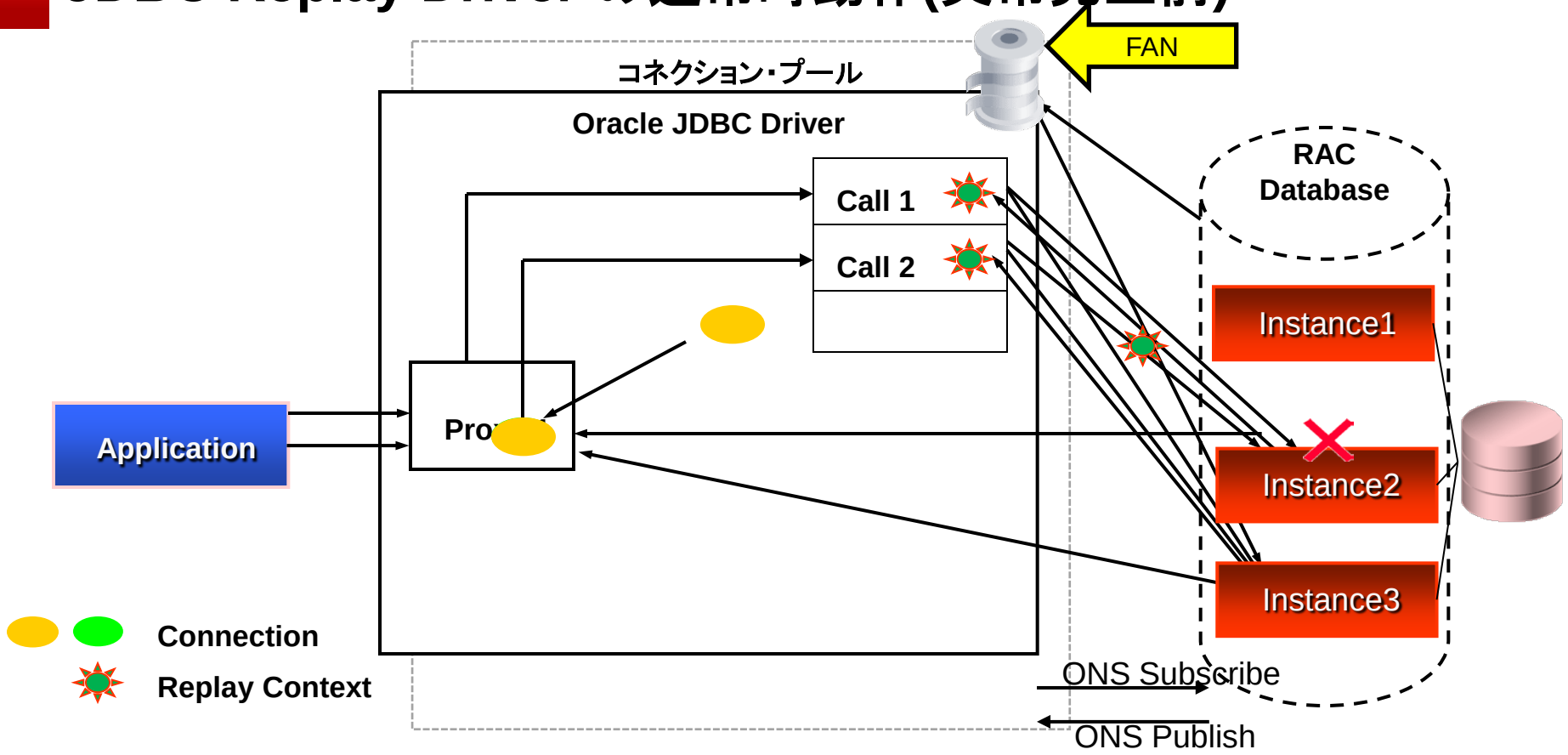


トランザクション自動再実行のための新実装

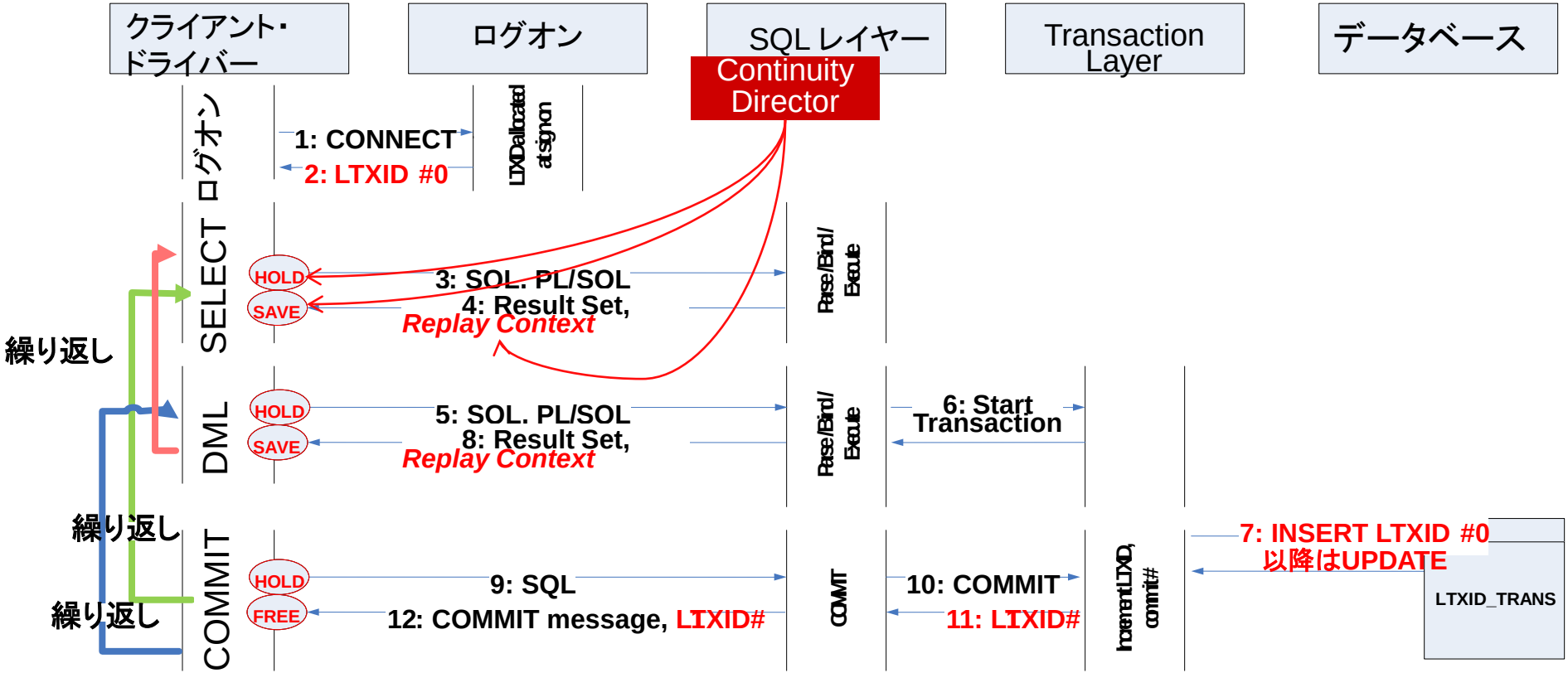
Transaction Guard 以外のコンポーネント

JDBC Replay Driver	Continuity Director	Replay Context
•クライアント側	•サーバー側 •Replay Driverに対しCallの保持またはパージの指示	•サーバーで生成されドライバーへ返される

JDBC Replay Driver の通常時動作(異常発生前)



12c で AC 有効時の通常時動作(異常発生前)

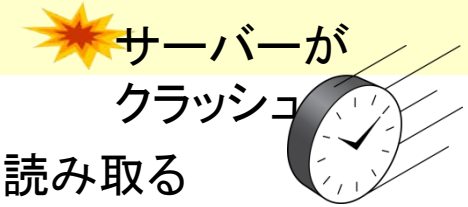


“リクエスト” 開始から初回のDMLまでの読み取りは再生時も同じ SCN で読み取る

```
// リクエスト開始
SELECT 日時　FROM ログイン履歴
SELECT 値段、品目 FROM 購入履歴　WHERE …
UPDATE 買い物カゴ SET USER=‘U1234’ …
```

SCN#1

SCN#2



- 初回のSELECT文発行時の SCN　時のスナップショットを読み取る

```
SELECT 日時　FROM ログイン履歴
SELECT 値段、品目 FROM 購入履歴　WHERE …
UPDATE 買い物カゴ SET USER=‘U1234’ …
```

現在のSCN

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

潜在的な副産物(Side-Effects)

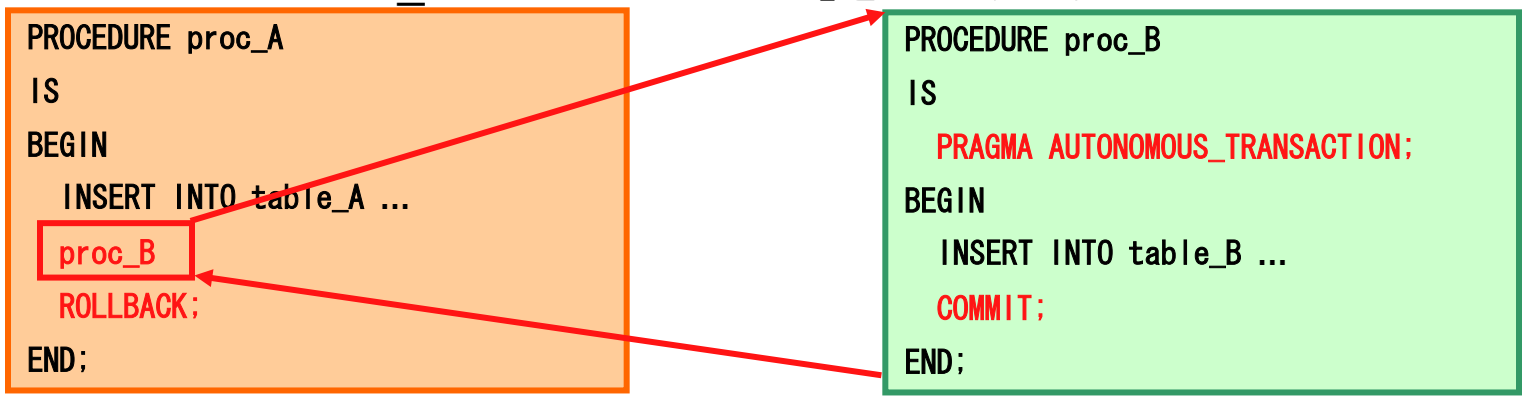
リクエスト中の以下は初回実行時と再生時に2度実行される
2度実行されてはいけない場合、再生無効化(disableReplay() 呼び出しを使う)の必要あり

- Autonomous transaction(自律型トランザクション)
- 以下は データベース の外に対する操作
  - UTL_HTTP - making HTTP callouts
  - UTL_URL - accessing URLs
  - UTL_FILE, UTL_FILE_TRANSFER – ファイルシステムアクセス
  - UTL_SMTP, UTL_TCP, UTL_MAIL – ネットワーク
  - DBMS_PIPE - to external sources
  - DBMS_ALERT - email or other notifications

自律型トランザクション

実行中のトランザクションとは別トランザクションのサブプログラムを呼び出すことができる

呼び出されるサブプログラムには自律型トランザクションの設定「PRAGMA AUTONOMOUS_TRANSACTION句」を記述する。

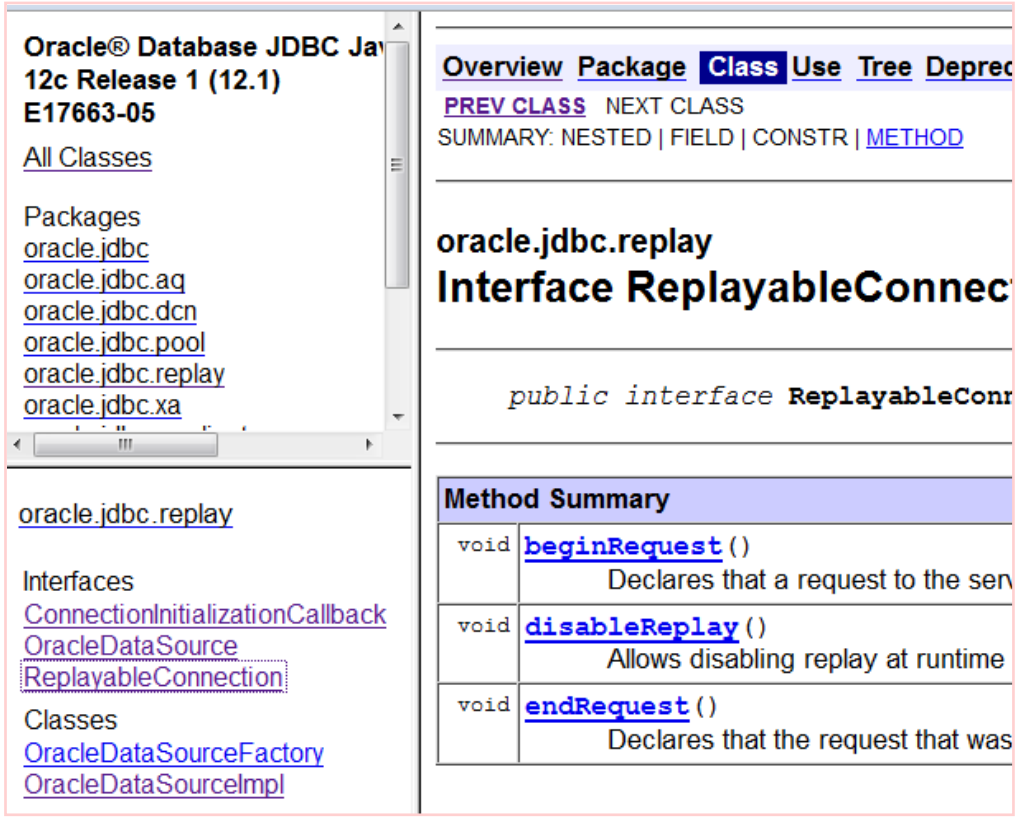


※ table_A へデータは登録されないが、table_Bへはデータが登録される

監査情報を登録するなどの用途で利用

47

JDBC新規パッケージ、インターフェース、メソッド



- beginRequest()
  - リクエスト境界の開始を宣言
  - Oracle製コネクションプール(UCP, Weblogic)を使わない場合に呼び出す
- disableReplay()
  - 記録(キャプチャー)と再生をキャンセル
- endRequest()
  - リクエスト境界の終了を宣言
  - 上記 beginRequest() に対応

サービス属性

srvctl 例: ポリシー管理RAC上

```
$ srvctl add service -db orcl -service acservice
      -serverpool ora.orcldb
      -cardinality singleton
      -failovertype TRANSACTION
      -commit_outcome TRUE
      -session_state dynamic
      -failoverretry 50
      -failoverdelay 5
      -retention 86400
      -replay_init_time 300          // デフォルト 5 分
      -notification TRUE
```

Application Continuity に必要な設定

Application Continuity でのオプショナルな設定

セッション・ステート session_state {dynamic|static}

- アプリケーションで/リクエスト中での Session State (セッション･ステート) の使用
  - Non Transactional(トランザクショナルではない) Session State(NTSS)
    - セッション状態の変化はDBに コミット/永続化 されない
    - 例) ALTER SESSION SET …..=‘…….’
    - 例) PL/SQL グローバル変数
- 開発者/DBAが判断し以下のどちらかをデータベース･サービス属性に設定
  - DYNAMIC モード (デフォルト)
    - non-transaction server-side session state(NTSS) が ”リクエスト” 境界の中で変更される可能性がある
      - 再生/再実行はリクエスト開始から最初の COMMIT まで。そのCOMMIT以降は再生不可能
  - STATIC モード
    - NTSSが変更されないことが分かっている
      - リクエスト境界中に複数回のCOMMIT(複数トランザクション)があっても全てのトランザクションを再生可能
      - 設定する際には慎重なアプリケーションのアセスメントが必要です。

“リクエスト” 中の複数回 COMMIT

- DYMAMIC モードでは最初のCOMMITまでが再生対象です
  - 2つ目のトランザクションは再生されません

```
select   catid, name, descn from category where catid = ?
select   productid, name, descn, category from product where(lower(name) like ?)
select   i.itemid, listprice, unitcost, supplier, i.productid, name, descn, category, status, attr1, attr2, attr3, attr4, attr5, qty
          from item i, inventory v, pr…….
select   qty as value from inventory where itemid = upper(?)
select   ordernum.nextval from dual
insert   into orders (orderid, userid, orderdate, shipaddr1, shipaddr2, shipcity, shipstate, shipzip, shipcountry, billaddr1, billadd
          billcity, billstate, bill……
insert   into orderstatus (orderid, linenum, timestamp, status) values (?, ?, sysdate, ‘P’)
conn.commit();                                                    // コミット
// 2 つ目のトランザクション開始
insert   into lineitem (orderid, linenum, itemid, quantity, unitprice) values (?, ?, ?, 1, 16.5)
update  inventory set qty = qty - 1 where itemid = upper(?)
conn.commit();                                                    // コミット
```

再生可能範囲

初回実行時と同じ値を使うよう設定が可能なMutable

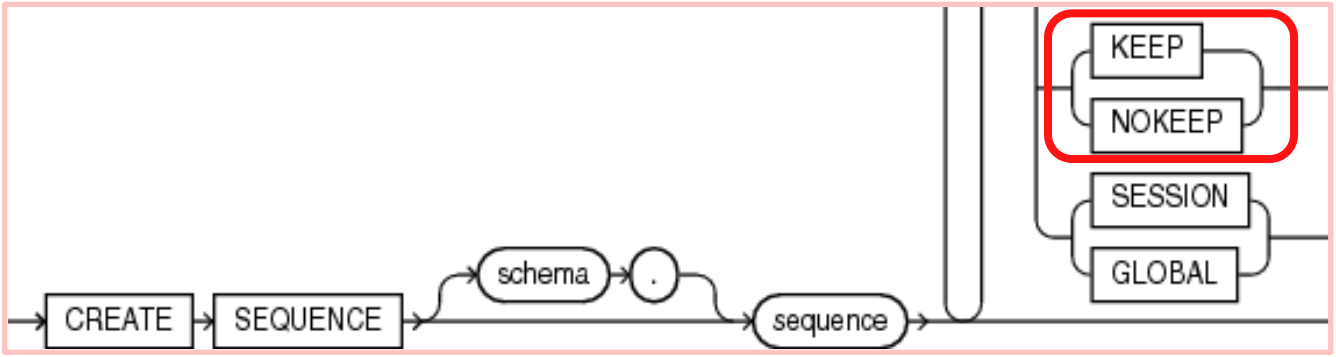
Mutable、変化物: 呼び出す毎に違う値を返す関数

シーケンス	タイムスタンプ	ユニークID生成
mySeq.NEXTVAL	SYSDATE SYSTIMESTAMP	SYS_GUID *再生時に Parallel Query で実行される場合同じ値は使われません

補足：Database Replay(RAT,Real Application Testing) にある機能と似た機能

シーケンス（順序）.NEXTVAL で過去と同じ値を使う

- アプリケーション実行ユーザーへのオブジェクト権限付与、取り消し
  - GRANT　KEEP SEQUENCE [to ユーザー] on [sequence オブジェクト]
  - REVOKE　KEEP SEQUENCE [from ユーザー] on [sequence オブジェクト]
  - GRANT ALL に上記は含まれませんので個別に実行する必要あり
  - SYS,AUDSYS などの組み込みユーザーはサポートされない
- シーケンス（順序）所有者の場合
  - {CREATE | ALTER} SEQUENCE [sequence オブジェクト]　{KEEP | NOKEEP}



タイムスタンプ、SYSGUID に対する設定

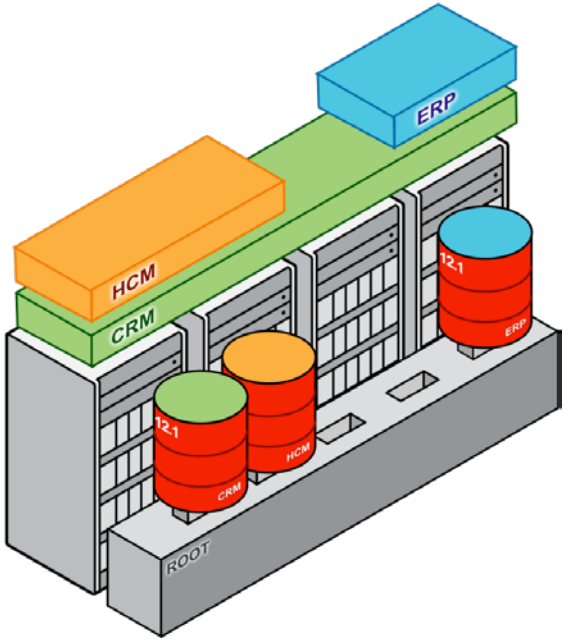
- システム権限
  - seq.NEXTVAL はオブジェクト権限(前スライド参照)
  - DBAロールに含まれる
- GRANT　[KEEP DATE TIME | KEEP SYSGUID] [to ユーザー]
- REVOKE [KEEP DATE TIME | KEEP SYSGUID] [from ユーザー]
  - DATE TIME = { SYSDATE および SYSTIMESTAMP }

Pluggable DB のサポート

- RAC, Data Guard 共サポート
- ALTER SESSION SET CONTAINER=…
  により PDB を切り替える動作は non-CDB での logoff/logon に相当し Transaction Guard および Application Continuity はサポートされません。

```
ords.setURL(jdbcURL);
ords.setUser(user);
ords.setPassword(passwd);

Connection con = ords.getConnection();
doSQL(con, “select ename from emp where empno=3253");
doSQL(con, " ALTER SESSION SET CONTAINER=CDB1_PDB2");
doSQL(con, “INSERT INTO …");
con.close();
```



ALTER SYSTEM KILL SESSION

12c での変更点

- オプションのない KILL SESSION では ‘Recoverable’ エラーになるので Application Continuity 機能により再接続、再実行される可能性があります。
- 再接続させないためには以下のオプションで実行します。

```
alter system  kill session            'sid, serial#, @inst' noreplay
alter system  disconnect session 'sid, serial#, @inst' noreplay

DBMS_SERVICE.DISCONNECT_SESSION(
‘[service name]’,
DBMS_SERVICE.NOREPLAY
)
```

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

UCP での新旧接続ファクトリクラスによる動作の違い

接続ファクトリクラス名をプロパティファイルで設定

```
$ cat runnoreplay
java -classpath
./actest.jar:$ORACLE_HOME/ucp/lib/ucp.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar
actest.ACTest actest_noreplay.properties

$ grep datasource actest_noreplay.properties
datasource=oracle.jdbc.pool.OracleDataSour
ce
```

```
$ cat runreplay
java -classpath
./actest.jar:$ORACLE_HOME/ucp/lib/ucp.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar
actest.ACTest actest_replay.properties

$ grep datasource actest_replay.properties
datasource=oracle.jdbc.replay.OracleDataSo
urceImpl
```

アジェンダ

1. アプリケーションのフェイルオーバー: 現状
2. Transaction Guard
   i. Recoverable Exception(回復可能な例外)
   ii. 動作
   iii. 設定
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細
   iv. 設定
4. Application Continuity for Java デモ
5. Q&A

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®